Chapter Ordering in Novels

Allen Kim, Steven Skiena
Department of Computer Science,
Stony Brook University, NY, USA
{allekim,skiena}@cs.stonybrook.edu

Abstract

Understanding narrative flow and text coherence in long-form documents (novels) remains an open problem in NLP. To gain insight, we explore the task of *chapter ordering*, reconstructing the original order of chapters in novel given a random permutation of the text. This can be seen as extending the well-known *sentence ordering* task to vastly larger documents: our task deals with over 9,000 novels with an average of twenty chapters each, versus standard sentence ordering datasets averaging only 5-8 sentences. We formulate the task of reconstructing order as a constraint solving problem, using *minimum feedback arc set* and *traveling salesman problem* optimization criteria, where the weights of the graph are generated based on models for character occurrences and chapter boundary detection, using relational chapter scores derived from RoBERTa. Our best methods yield a Spearman correlation of 0.59 on this novel and challenging task, substantially above baseline.

1 Introduction

Novels are a fundamental long-form medium for storytelling, requiring understanding of narrative flow for full comprehension of the text. Although neural network-based language models demonstrate amazing performance in comprehending short text windows, the problems of understanding long-form narrative texts such as summarization remain largely open.

A major challenge in research on long-form narration is the cost of annotation: the mere act of reading a novel generally requires a 10-15 hour commitment on the part of the annotator, making it clearly cost prohibitive to obtain full annotations of a significant corpus of novels. Although published summaries are available for up to a few hundred popular books, annotation cost is the rate limiting step for work in this area (Wu et al., 2021).

In this paper, we propose a new task that gets to the heart of narrative flow analysis for novels, without the need for human annotation. Most novels are partitioned into chapters as a means of providing thematic breakpoints for the reader. Randomly permuting the order of these chapters obviously ruins the coherence of the narrative flow of the underlying story. Here we propose the new task of *chapter ordering*: reconstructing the original order of the book given an input random chapter permutation of the text.

This task is clearly related to the task of sentence ordering (Lapata, 2003; Barzilay and Lapata, 2005, 2008), reconstructing a set of sentences in a tiny story as to maximize their coherence. While there is a large volume of prior work on sentence ordering, to our knowledge, this is the first paper to address chapter ordering. The natural approaches for the sentence task cannot be readily applied to chapter order. First, chapters in novels usually contain several hundred sentences, which cannot easily be encoded with the neural models applied to sentence ordering. Second, the number of chapters in a novel (an average of 21.6 chapters/novel in our corpus) versus 5 to 8 sentences in popular datasets. This matters, because the impressive performances of the best sentence ordering models show severe degradation with longer sequences. Finally, ordering sentences in a tiny story is a simpler perceptual task for humans, whereas correctly ordering the chapters of even a single novel requires hours of reading and analysis. The combination of these factors makes chapter ordering a difficult task that cannot be tackled with standard methods. The overview of our approach can be found in Figure 1.

In this paper, we define the chapter ordering problem and propose a variety of neural-based models to score the relative order of chapter-pairs. Our main contributions[1] include:

[1]Code and link to dataset found at https://github.com/allenkim/chapter-ordering-in-novels

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 3838 - 3848

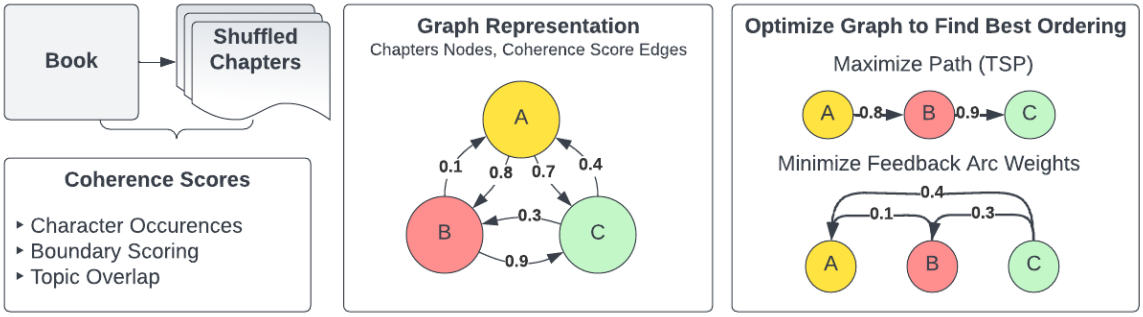

Figure 1: High-level overview of the chapter ordering task: after partitioning and shuffling a novel's chapters, we define a network of coherence scores between chapter pairs, with the permutation reconstructed using an optimization criteria combining the traveling salesman problem (TSP) and minimum arc feedback set.

- *Definition of the Chapter Ordering Task, with Associated Data Set* – We define an accessible new task relevant to study narrative understanding on large texts, building on our dataset of 9,007 text-cleaned, chapter-permuted Project Gutenberg novels, each with between 5 and 50 chapters per book and between five and two hundred paragraphs per chapter. This is large enough to facilitate interesting machine learning-based approaches, and provides a standardized resource for future studies of chapter ordering to start with.

- *Neural-based Coherent Models of Narrative Flow* – We model the task of chapter ordering using weighted directed graphs, where edge (x, y) receives a coherent score predicting the likelihood that chapter x appears (immediately) before y in the published work. We propose several natural coherence notions implicit in long-form narratives, including:
  - *Character entry/exit recognition:* Tracking the life progression of characters through a novel is essential to understanding of the flow of the story. We train effective classifiers to recognize each character's introduction
  - *Identifying initial/concluding chapters:* There are subtle characteristics of the start and end of a story that distinguishes it from the middle of the narrative.
  - *Neighboring chapter detection:* Narrative flow implies that the events at the end of one chapter are often connected to the start of the next chapter. This direct correlation provides strong reinforcement in keeping certain chapters paired.
  - *Topic correlation:* Generalizing beyond chapter boundaries, the topical content in nearby chapters is generally more closely related than those further away.

- *Optimization Techniques for Chapter Ordering* – We present a method for finding the chapter order that maximizes the coherence score based on two classical problems on directed weighted graphs: minimum arc feedback and traveling salesman problem. We demonstrate that combining both objectives simultaneously best exploits the variety of coherence constraints inherent in narrative analysis.

The paper is organized as follows. Section 2 surveys related work in sentence ordering and whole book-oriented NLP. Section 3 formalizes our task. We present our methods for coherent scoring in Section 4, and in Section 5 our optimization techniques. We present our experimental results in Section 6 before concluding with future directions for research.

2 Related Work

The sentence ordering task was first proposed to test coherence of sentences (Lapata, 2003; Barzilay and Lapata, 2005, 2008). Traditionally, this was tackled using human designed features and classical machine learning techniques. This included heuristics with Markov models (Barzilay and Lee, 2004; Bollegala et al., 2005; Ji and Pulman, 2006), K-means clustering (Ji and Nie, 2008; Zhang, 2011), support vector machines (Bollegala

et al., 2006; Nahnsen, 2009; Peng et al., 2009; Yanase et al., 2015) and others like latent semantic analysis (Zhang et al., 2010) and conditional random fields (Gella and Duong Thanh, 2012). We also note the representation of sentence ordering as a graph in many past works as well (Elsner and Charniak, 2011; Li et al., 2011; Guinaudeau and Strube, 2013). However, with advances in neural modeling, other approaches have developed.

One category are pairwise ordering models, which deal with ordering at a pairwise level and combines the results to get a global ordering. Traditionally, the models used in these approaches have been GRUs and LSTMs (Chen et al., 2016; Agrawal et al., 2016; Moon et al., 2019), but recently have moved towards Transformer-based models like BERT (Kumar et al., 2020; Shen and Baldwin, 2021). One common approach to sentence ordering considers the sentences as a graph and optimizes the weights using algorithms such as topological sort (Prabhumoye et al., 2020) or the asymmetric traveling salesman problem (Keswani and Jhamtani, 2021). The main disadvantage is that a pair-wise ordering of two sentences is determined in isolation from the other sentences. It may be the case that the other sentences provide context that make the ordering sensible.

Thus, the other main category of sentence ordering deal with set-to-sequence models, which deal with end-to-end encoder-decoder frameworks. Most common are pointer networks for sequence prediction (Gong et al., 2016; Cui et al., 2018; Yin et al., 2020). This is often combined with other methods such as recurrent neural networks (Logeswaran et al., 2018; Oh et al., 2019). Recently, works have been incorporating graph-related networks (Cui et al., 2020; Yin et al., 2021; Lai et al., 2021) as well. Additionally, the task has been considered as a text-to-marker generation problem using BART (Basu Roy Chowdhury et al., 2021). These methods have the advantage that all the sentences are taken in context, but lack the interpretability of the previous models discussed.

We also rely heavily on literature in natural language processing on books. Analysis of books have been streamlined through pipelines like BookNLP (Bamman et al., 2014) as well as datasets of entities (Bamman et al., 2019). We also find other book-related works that improve the quality of books as well as understanding aspects such as time (Kim et al., 2020, 2021). In particular, we focus on chapters and require proper chapter segmentation (Pethe et al., 2020).

3 Problem Formulation

The chapter ordering task can be formulated as follows. Given a book of N chapters, let $C = [c_1, \ldots, c_N]$ be the sequence of chapters where each c_i represents the text content of chapter i for $1 \leq i \leq N$. Let $o = [o_1, \ldots, o_N]$ be a random permutation of indices from 1 to N. Given a sequence of random chapters $r = [r_{o_1}, \ldots, r_{o_N}]$, the task is to find the correct order of chapter indices $o' = [o'_1, \ldots, o'_N]$ such that $[r_{o'_1}, \ldots, r_{o'_N}] = C$.

3.1 Dataset

Our dataset comes from Project Gutenberg (Gutenberg, n.d.) filtered down to English fiction books of which we collected 19,437. We follow Pethe, Kim, and Skiena (2020) to clean and annotate chapters using regular expressions on common chapter headings. To filter out anomalies, we mandate that the number of chapters have to be between 5 and 50. Additionally, we restrict each chapter to be at least 5 and at most 200 paragraphs. As a final restriction, we also require a numbering of chapters that start from one, either in words or numeral form. This is to avoid books that may be sequels or other parts of a longer series. We note that the chapter title as well as the chapter numbers are not included as part of the input. Any books that do not follow these requirements are filtered out of the dataset.

In total, we have 9,007 books with a mean chapter length of 21.64 and standard deviation of 9.96. The median chapter length is 21 with the min and max being 5 and 50 respectively due to the filtering. These were then randomly train-test split in an 8:2 ratio.

4 Coherence Scoring

In this section, we describe our observations of books that help us quantify relations between chapters. Since the goal is to provide a benchmark on the feasibility of the chapter ordering task and not an exhaustive comparison on the effectiveness of different models, we primarily use RoBERTa (Liu et al., 2019) as our main pretrained language model for all relevant tasks.

Implementation Details. We use the Transformers (Wolf et al., 2019) library for fine-tuning RoBERTa. All models were run on a compute server with 2.30 GHz CPU and TeslaV100 GPU.

	Intro (class 0)			Middle (class 1)			End (class 2)		
	F1	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**
Only Intro	15.67	8.50	**100**	-	-	-	-	-	-
Only Middle	-	-	-	90.71	82.99	**100**	-	-	-
Only End	-	-	-	-	-	-	15.67	8.50	**100**
Guessing	8.56	8.54	8.54	82.97	83.00	82.93	8.55	8.51	8.60
RoBERTa	**52.45**	**43.71**	65.58	87.20	**91.21**	83.52	**41.33**	36.30	47.99
GPT2	33.29	57.96	23.35	**91.28**	85.58	97.80	24.31	**69.17**	14.75

Table 1: Evaluation of Character Introduction / End Model with F1 score (F1), Precision (P), Recall (R) - guessing was done proportional to the occurrence ratio

No hyperparameter tuning was done on any models; default values were run for all models.

4.1 Character Entry/Exit Recognition

Characters are a fundamental aspect of any story. By tracking the life of a character through a book, we can greatly improve our understanding of the underlying story. In particular, characters are typically introduced into the story with physical descriptions and also end up leaving the story with details of their departure.

We define the task as follows: given an occurrence of a character and the context of text it appears in, we want to predict whether this is the first time a character is introduced, the last time it is mentioned, or some time in the middle. This is treated as a three-class (intro, middle, end) classification problem.

Data Preparation. For each book, we apply named entity recognition with coreference resolution using Stanza (Qi et al., 2020) to identify characters. Afterwards, we cluster the same characters together using naming convention heuristics e.g. Sherlock Holmes and Mr. Holmes. For each character occurrence in the text, we extract a 512 sub-token window around it. The names of the character in question were replaced with a special `<main>` token and every other character in the context was replaced with a special `<other>` token. These context windows were then used for fine-tuning RoBERTa with a token classification head with three classes. Only the first occurrence was marked as class 0 (intro) and only the last occurrence was marked as class 2 (end); the majority of the instances were marked as class 1 (middle).

Evaluation. Table 1 shows the results of the model on the test set for each of the predicted classes - Intro (first occurrence of character), Middle (some middle occurrence of a character), End (last occurrence of a character). Each class is evaluated using standard metrics, where $F1$ represents the F1 score, P represents precision, and R represents recall. In general, we see that RoBERTa performs well in predicting introductions of characters and decently for ends as well. This is sensible as when characters are introduced, there are more descriptive adjectives and related signals in the text. Typically, characters also leave the story at the end of some event, which has its own signals as well.

4.2 Initial/Concluding Chapters

As in the case of sentence ordering, there are characteristics of the start and end of a book that make it more discernible as compared to the middle of the book. Generally, there are "introducing" words related to setting the scene in the beginning while there are "concluding" words related to closing up the story in the end, similarly to the introduction and ends of characters.

We consider two tasks: first chapter prediction and last chapter prediction. For first chapter prediction, given text at the start of a chapter, the task is to determine if it is the first chapter or not. Similarly, for last chapter prediction, given text at the end of a chapter, the task is to determine if it is the last chapter or not. These are treated as two separate binary classification models.

Data Preparation. For the first chapter prediction task, for each book, we take the *first* 512 subtokens of each chapter and use the first chapter as a positive example and the others as negative. Analogously, for the last chapter prediction task, for each

	Acc	F1	P	R
RoBERTa First	96.1	52.8	58.0	48.4
RoBERTa Last	97.2	65.7	74.6	58.7
GPT2 First	96.3	52.3	56.3	48.8
GPT2 Last	96.8	56.0	74.0	45.0

Table 2: First and last chapter prediction metrics, Accuracy, F1 score (F1), Precision (P), Recall (R), across *all chapters*

	First Chapter	Last Chapter
P@1	0.577	0.675
P@3	0.737	0.817
P@5	0.802	0.875

Table 3: First and last chapter precisions *across all 1,802 books* in test set - P@k represents the presence of the target in the top k

book, we take the *last* 512 sub-tokens of each chapter and use the last chapter as the positive example. Both these tasks were fine-tuned on RoBERTa with a token classification head with two classes.

Evaluation. Tables 2 and 3 show results for the first and last chapter predictions. In general, we see that the model is able to predict the correct first chapter for more than 57% of books and predict the correct last chapter for more than 67% of books. It is interesting to note the relatively better performance of last chapter prediction to first chapter, which indicates more common signals among book endings as compared to introductions.

4.3 Boundary Scoring

The end of one chapter is typically connected to the start of the next. This sequential correlation provides strong reinforcement in keeping certain chapters paired. We consider the following binary classification task for a book: given the end of chapter X and the start of a different chapter Y, does chapter Y directly follow X?

Data Preparation. As positive examples, we concatenate the text at the end of a chapter with the text at the start of the next chapter with a separator token. Every other combination is considered to be a negative example. Thus, a book with N chapters contain $N-1$ positive examples and $N(N-1)-(N-1)$ negative examples. Given an input size of 512 sub-tokens, we concatenate the last 256 tokens of one chapter and the first 256 tokens of another chapter as input to the model (accounting for special tokens). We again fine-tune RoBERTa with a token classification head for two classes.

Evaluation. Table 4 shows the results for boundary detection. In general, we find that this is a difficult task in isolation with an F1 of up to 0.32 in the macro scale, but nevertheless, this provides valuable signal chaining chapters together when the model is confident.

4.4 Topic Overlap

Similar to chapter boundaries, the discussion of content in one chapter is generally more closely related to the content of a neighboring chapter as opposed to one further away. We capture this with a simple lemma overlap score between chapters. We do so by counting the number of lemmas in one chapter that is in common with another and normalizing by the number of lemmas in the smaller chapter. Lemmatization was done using Stanza. Stop words were filtered out using NLTK (Loper and Bird, 2002).

Figure 2 shows the average topic score for varying distances between chapter pairs. As expected, we see that neighboring chapters share the most overlap score while chapters further away decrease in value.

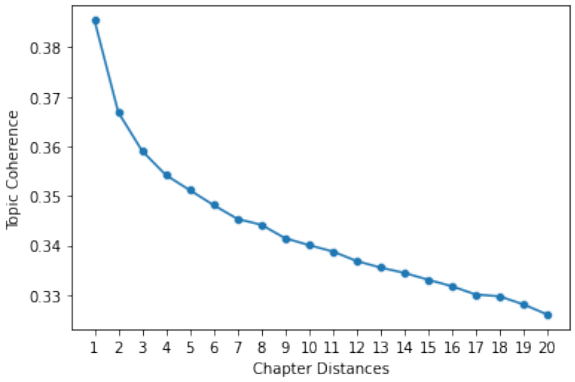

Figure 2: Average Topic Overlap Scores as a function of chapter distance - chapters closer together tend to overlap in topic more than those further away

5 Constraint Solving Problem

Section 4 discussed different methods of measuring coherence between chapters. For each method, we construct a weighted matrix that provides chapter pairwise scores. These matrices can then be used as weights of a directed graph that we can optimize to find an optimal ordering.

	Micro (Support: 1,019,744)				Macro (Support: 1,802)			
	Accuracy	**F1**	**P**	**R**	**Accuracy**	**F1**	**P**	**R**
Random	92.75	3.69	3.69	3.70	90.89	4.03	5.23	3.69
RoBERTa	**96.574**	27.86	**66.74**	17.61	**94.81**	23.78	**57.17**	17.03
GPT2	96.36	**30.50**	54.08	21.24	94.55	**26.09**	50.13	**20.60**

Table 4: Evaluation of Chapter Boundary Model with Accuracy, F1 score (F1), Precision (P), Recall (R) - Micro shows results across all chapter order pairs while Macro shows averaged metrics across each book

5.1 Computing Coherence Matrices

For each method, we describe the details in how the weights are constructed.

Character Introductions and Exits. Given a model that can predict the first and last occurrences of a character in a book, for each character, we identify the chapter where the character was most likely introduced as well as the chapter where the character was most likely last seen. Given the best guess of the first chapter and last chapter a character appears in, we add a vote to each chapter pair from first to others and others to the last chapter. We then count the votes among all the characters in the book, and normalize between disagreeing votes. For example, if there are three characters that seem to start at chapter X before chapter Y, but one character that seems to start at chapter Y before chapter X. The directed edge weight from node X to Y would be 0.75, while the weight from Y to X would be 0.25.

Boundary Scoring with First and Last Chapter Prediction. For each chapter, we can directly compute a score to every other chapter by directly extracting the probabilities from the boundary scoring model. To factor in first and last chapters, we also consider sentinel nodes that act as the first and last nodes and add weights from the first sentinel node to every other non-sentinel node based on the output from the first chapter model. Analogously, we add weights from every non-sentinel node to the last sentinel node based on the output from the last chapter model.

Topic Overlap. For each pair of chapters, we compute a lemma intersection score as described in Section 4.4. As an example, a filtered chapter with the words "walk happy park bench" would have a score of 0.75 with a filtered chapter of "walk happy park ball play". This score directly defines the weight edges between every pair of chapters.

5.2 Optimization Methods

We first consider optimal ways to find orderings for each method individually, and then consider approaches that combine them.

- *Character introductions and ends* - This can be optimized by finding the order that minimizes the sum of the weights of back edges, i.e. edges that go from a node to an earlier one in the given order.
- *Chapter boundaries with first and last chapters* - This can be treated as finding a directed path that visits each node exactly once and maximizes the coherence scores in a weighted, asymmetric graph.
- *Topic overlap* - This is the same as above, but for a weighted, symmetric graph.

For these optimization tasks, we consider two classical problems, both known to be NP-hard.

Minimum feedback arc set. A feedback arc set is a subset of edges in the graph that contains at least one edge out of every cycle in the graph. The edge constraints from character entries/exit scores imply entries before exits; the topological order implicit after removing the feedback set minimizes violated constraints.

Traveling salesman problem. Given a graph, what is the shortest possible route that visits each node exactly once? The weights from boundary, first/ last chapter, and topic overlap scores naturally define appropriate edge weights for TSP.

Solutions. Although these problems are NP-complete, heuristic solutions are effective. For minimum feedback arc set, we consider a heuristic approach of local minimization using random swaps. For the traveling salesman problem, we

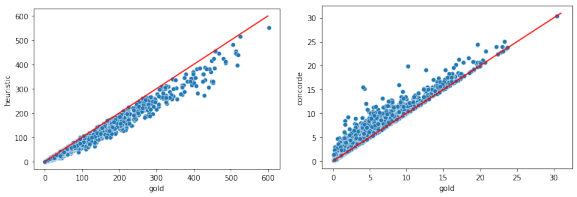

Figure 3: The left figure shows the feedback arc set weights of the correct ordering (gold) versus the feedback arc set weights of our optimized ordering (heuristic) while the right figure shows TSP scores of the correct ordering (gold) versus the TSP scores of our optimized ordering (concorde).

employ an off-the-shelf exact solver, Concorde[2] (Applegate et al., 2009), which employs branch and bound techniques to make our task feasible within a short time.

To show the effectiveness of our optimization, we compare the scores we obtain using our methods to the scores obtained applying the actual chapter orders. Figure 3 shows the feedback arc set weights as well as the TSP weights using our optimization and the ones found through the gold standard. In both cases, we see that our ordering generally has lower feedback arc weight than that of the correct order and has higher TSP scores than that of the correct order, showing the optimality of Concorde and our local search optimization.

The scale of TSP scores and feedback arc set scores are quite different, so we normalize them into z-scores before mixing them. This was done by sampling 100 random orders and computing their metrics to get sample mean and standard deviations. The experiments reported in Table 5 show that employing both optimization criteria in a 50-50 mix typically identifies the best order.

6 Experimental Results

6.1 Evaluation Metrics

We follow the same metrics used in works to evaluate sentence ordering. For all of these metrics, let N be the number of books in our test set.

Perfect Match Ratio (PMR). PMR measures the fraction of books that were predicted to be *exactly* the same as the original order. This is the strictest metric.

$$\text{PMR} = \frac{1}{N}\sum_{i=1}^{N} 1(o_i = o'_i)$$

where o_i and o'_i represents the correct order and predicted order respectively.

Chapter Accuracy (Acc). Acc measures the average percentage of chapters for which the predicted absolute position was correct within each book. This is also a strict metric.

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{k_i}\sum_{j=1}^{k_i} 1(o^i_j = o'^i_j)\right]$$

where k_i represents the number of chapters in book i, and o^i_j and o'^i_j represents the correct and predicted chapter index for the j-th chapter in book i respectively.

Kendall Tau (Tau). Tau quantifies the distance between the predicted order and the correct order in terms of the number of inversions (Lapata, 2006).

$$\tau = 1 - \frac{2(\#\text{ inversions})}{\binom{n}{2}}$$

Spearman. Spearman measures the rank correlation of the predicted order with the standard increasing chapter order. The indices of the correct order are remapped such that the indices are strictly increasing before computing. Equation omitted as this is a well-known metric.

Rouge-S. Rouge-S calculates the percentage of chapter pairs for which the relative order is predicted correctly (Chen et al., 2016). There is no penalty for gaps between the chapter pairs.

$$\text{Rouge-S} = \frac{1}{\binom{k_i}{2}}\left[\text{Pairs}(o_i) \cap \text{Pairs}(o'_i)\right]$$

where k_i represents the number of chapters in book i, and Pairs(o) represents all $\binom{k_i}{2}$ relative ordering pairs for order o_i.

Longest Common Subsequence (LCS). LCS measures the ratio of the longest common subsequence between the predicted order and the correct order, computed using dynamic programming.

[2]We note that Concorde only solves the symmetric TSP problem, so we convert our asymmetric instances into symmetric ones by creating dummy nodes, doubling the number of nodes (Jonker and Volgenant, 1983).

Method	Param	PMR	Acc	Tau	Spearman	Rouge-S	LCS
Random	-	0.000	5.710	0.000	-0.192	50.142	33.590
Character	-	0.777	18.012	10.446	55.251	72.022	49.699
Boundary	Default	1.664	10.207	3.856	10.503	56.589	47.437
	+First/Last chapter	3.163	18.150	11.517	28.500	63.160	52.236
$c\cdot$ **Boundary with** $(1-c)\cdot$ **Topic**	$c = 0.0$	0.277	6.799	0.783	0.025	50.138	36.896
	$c = 0.25$	3.219	19.994	12.990	32.452	64.662	53.102
	$c = 0.5$	3.718	20.541	13.881	33.488	65.247	54.267
	$c = 0.75$	**3.940**	20.614	13.426	32.900	65.136	54.437
$c\cdot$ **Character with** $(1-c)\cdot$ **[best Boundary + Topic]**	$c = 0.0$	0.333	11.927	5.084	17.870	58.124	46.205
	$c = 0.25$	1.332	19.909	13.041	57.290	73.505	54.155
	$c = 0.5$	1.332	19.846	11.780	57.375	73.496	54.151
	$c = 0.75$	1.48	19.916	12.648	57.758	73.703	54.237
$c\cdot$ **Character with** $(1-c)\cdot$ **[best Boundary + Topic] initialized with best TSP**	$c = 0.0$	3.885	20.581	13.402	32.844	65.105	54.402
	$c = 0.25$	1.387	22.010	14.071	59.204	74.469	**55.361**
	$c = 0.5$	1.554	**22.048**	**14.537**	59.174	74.416	55.210
	$c = 0.75$	1.498	21.684	14.270	**59.493**	**74.518**	55.157
	$c = 1.0$	1.276	21.413	14.159	58.499	73.699	51.993

Table 5: Results of chapter ordering - **Random** refers to random ordering, **Character** refers to scores derived from character introduction and ends and is optimized with local search for feedback arc set, **Boundary** tests boundary scores with and without first and last chapter augmentations and optimized with Concorde, **Boundary with Topic** refers to the augmented boundary scores jointly with the topic overlap score and optimized with Concorde. $c = 1$ is omitted as it is equivalent to **Boundary**. **Character with best Boundary and Topic** refers to character scores jointly with the best **Boundary with Topic** scores ($c = 0.75$) and optimized with local search on both TSP and feedback arc set. Again, $c = 1$ is omitted as it is equivalent to **Character**. The last section is the same as before, but with the initial optimization being seeded with the optimal order found with Concorde in **Boundary with Topic**.

6.2 Analysis

Table 5 presents our final results as an ablation study, giving a complete breakdown of different model variations. We note that adding initial/conclusion chapter prediction to the boundary matrix as all the metrics improve considerably. Our strongest results combine all methods of coherence scoring, with top performance when the boundary scores are weighted more heavily than the topic coherence score.

Figure 4 show the decline in our overall metrics as the number of chapters increases. As previously shown in the sentence ordering task, the problem gets harder as the number of text units increases.

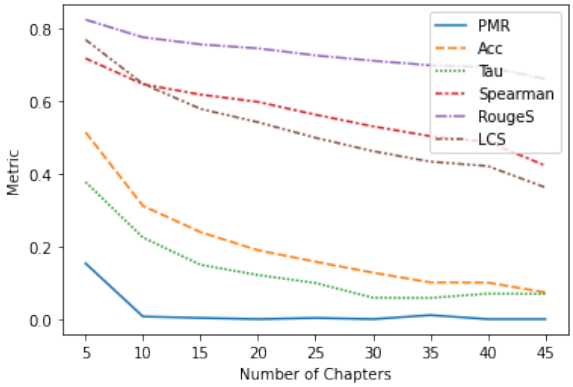

Figure 4: Performance metrics bucketed by chapter count: the ordering task gets harder for longer books.

7 Conclusion

In this paper, we introduce a new task of chapter ordering as an approach to study long-form narrative understanding without cost-prohibitive human annotation. The standard approaches used in the

smaller-scale sentence ordering tasks do not extend directly to chapter ordering, because longer texts exceed the capacity of current neural language models.

Future work may include incorporating efficient transformers that incorporate longer text windows, although we believe that approaches which explicitly incorporate logical reasoning about events may be necessary to achieve substantial progress. We will release our full dataset on publication to encourage additional research on this task.

Limitations

To our knowledge, this is the first paper to introduce *chapter ordering*, and while we present novel methods to tackle it, we discuss limitations of our approach.

First, our dataset is limited to English fiction books that were generally written in the early 1900s and earlier. This is largely due to copyright issues involved with using more modern texts, and because of this, our models are inherently biased to the language used in older texts. It is unclear how well these models will perform on modern texts given changes in writing style over time. Inherently, there will also be model biases concerning gender and race that originate from these older texts as well.

Second, we use the base RoBERTa as our language model of choice to produce coherence scores, but we do not extensively survey other models primarily due to memory and time constraints. As the main bottleneck in our approach were the coherence scores as opposed to the optimization, using larger and more efficient language models should improve performance. In particular, efficient transformers such as Longformer (Beltagy et al., 2020) that are capable of taking in longer windows of text should show improved results for this task.

Finally, we present only one view of tackling this problem, based on optimizing a graph generated from coherence scores suggested by human observations on characters and story connection. All of our relational scores were derived from analyzing chapter pairs in isolation. However, it is possible that certain chapter orderings make sense only with other chapters as context, and this is not completely captured with our approach. Scores may also show improvement with a contrastive learning objective that considers all chapters holistically.

References

Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort story: Sorting jumbled images and captions into stories. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 925–931, Austin, Texas. Association for Computational Linguistics.

David L Applegate, Robert E Bixby, Vašek Chvátal, William Cook, Daniel G Espinoza, Marcos Goycoolea, and Keld Helsgaun. 2009. Certification of an optimal tsp tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15.

David Bamman, Sejal Popat, and Sheng Shen. 2019. An annotated dataset of literary entities. 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 2138–2144, Minneapolis, Minnesota. Association for Computational Linguistics.

David Bamman, Ted Underwood, and Noah A. Smith. 2014. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

Regina Barzilay and Mirella Lapata. 2005. Modeling local coherence: An entity-based approach. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 141–148, Ann Arbor, Michigan. Association for Computational Linguistics.

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

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 113–120, Boston, Massachusetts, USA. Association for Computational Linguistics.

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

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2005. A machine learning approach to

sentence ordering for multidocument summarization and its evaluation. In *Second International Joint Conference on Natural Language Processing: Full Papers*.

Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. 2006. A bottom-up approach to sentence ordering for multi-document summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 385–392, Sydney, Australia. Association for Computational Linguistics.

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.

Micha Elsner and Eugene Charniak. 2011. Extending the entity grid with entity-specific features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 125–129, Portland, Oregon, USA. Association for Computational Linguistics.

Spandana Gella and Long Duong Thanh. 2012. Automatic sentence classifier using sentence ordering features for event based medicine: Shared task system description. In *Proceedings of the Australasian Language Technology Association Workshop 2012*, pages 130–133, Dunedin, New Zealand.

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

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

Project Gutenberg. n.d. `www.gutenberg.org`. Accessed: June 2022.

Donghong Ji and Yu Nie. 2008. Sentence ordering based on cluster adjacency in multi-document summarization. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Paul D Ji and Stephen Pulman. 2006. Sentence ordering with manifold-based classification in multi-document summarization. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 526–533, Sydney, Australia. Association for Computational Linguistics.

Roy Jonker and Ton Volgenant. 1983. Transforming asymmetric into symmetric traveling salesman problems. *Operations Research Letters*, 2(4):161–163.

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

Allen Kim, Charuta Pethe, Naoya Inoue, and Steve Skiena. 2021. Cleaning dirty books: Post-OCR processing for previously scanned texts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4217–4226, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Allen Kim, Charuta Pethe, and Steve Skiena. 2020. What time is it? temporal analysis of novels. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9076–9086, Online. Association for Computational Linguistics.

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

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

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

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

Peifeng Li, Guangxi Deng, and Qiaoming Zhu. 2011. Using context inference to improve sentence ordering for multi-document summarization. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1055–1061, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

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

Han Cheol Moon, Tasnim Mohiuddin, Shafiq Joty, and Chi Xu. 2019. A unified neural coherence 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 2262–2272, Hong Kong, China. Association for Computational Linguistics.

Thade Nahnsen. 2009. Domain-independent shallow sentence ordering. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 78–83, Boulder, Colorado. Association for Computational Linguistics.

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.

Gongfu Peng, Yanxiang He, Ye Tian, Yingsheng Tian, and Weidong Wen. 2009. A novel method of sentence ordering based on support vector machine. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2*, pages 787–794, Hong Kong. City University of Hong Kong.

Charuta Pethe, Allen Kim, and Steve Skiena. 2020. Chapter Captor: Text Segmentation in Novels. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8373–8383, Online. Association for Computational Linguistics.

Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W Black. 2020. Topological sort for sentence ordering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2783–2792, Online. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Aili Shen and Timothy Baldwin. 2021. A simple yet effective method for sentence ordering. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 154–160, Singapore and Online. Association for Computational Linguistics.

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*.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*.

Toshihiko Yanase, Toshinori Miyoshi, Kohsuke Yanai, Misa Sato, Makoto Iwayama, Yoshiki Niwa, Paul Reisert, and Kentaro Inui. 2015. Learning sentence ordering for opinion generation of debate. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 94–103, Denver, CO. Association for Computational Linguistics.

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

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

Renxian Zhang. 2011. Sentence ordering driven by local and global coherence for summary generation. In *Proceedings of the ACL 2011 Student Session*, pages 6–11, Portland, OR, USA. Association for Computational Linguistics.

Renxian Zhang, Wenjie Li, and Qin Lu. 2010. Sentence ordering with event-enriched semantics and two-layered clustering for multi-document news summarization. In *Coling 2010: Posters*, pages 1489–1497, Beijing, China. Coling 2010 Organizing Committee.