Plan ahead: Self-Supervised Text Planning for Paragraph Completion Task

Dongyeop Kang[*]
dongyeopk@berkeley.edu
UC Berkeley

Eduard Hovy
hovy@cs.cmu.edu
Carnegie Mellon University

Abstract

Despite the recent success of contextualized language models on various NLP tasks, language model itself cannot capture textual coherence of a long, multi-sentence document (e.g., a paragraph). Humans often make structural decisions on what and how to say about before making utterances. Guiding surface realization with such high-level decisions and structuring text in a coherent way is essentially called a planning process. Where can the model learn such high-level coherence? A paragraph itself contains various forms of inductive coherence signals called self-supervision in this work, such as sentence orders, topical keywords, rhetorical structures, and so on. Motivated by that, this work proposes a new paragraph completion task PARCOM; predicting masked sentences in a paragraph. However, the task suffers from predicting and selecting appropriate topical content with respect to the given context. To address that, we propose a self-supervised text planner SSPlanner that predicts what to say first (content prediction), then guides the pretrained language model (surface realization) using the predicted content. SSPlanner outperforms the baseline generation models on the paragraph completion task in both automatic and human evaluation. We also find that a combination of noun and verb types of keywords is the most effective for content selection. As more number of content keywords are provided, overall generation quality also increases.

1 Introduction

One may think textual coherence can be achieved from a gigantic language model trained on massive data. This might be true in simple cases, such as generating short replies (Kannan et al., 2016), but not in a long, multi-sentence generation. This is mainly because per-word predictions from the autoregressive models can not capture the long-term flow of text, while humans often make structural decisions on what and how to say about before they speak (Byrne, 1979; McKeown, 1985; Hovy, 1990; Swan, 2002; Kang, 2020). Guiding the surface-level realization with such high-level decisions and coherently structuring output text is called a *planning* process.

Where can the model learn such high-level decisions related to long-term coherence? A written paragraph itself can be a pot of golden resources, containing various forms of inductive coherence signals. Different types of coherence signals in a paragraph have been studied and used in many different ways: a sequence of words or sentences (Devlin et al., 2019; Radford et al., 2019), a discourse structure of a text (Appelt, 1982; Hovy, 1991; Kang et al., 2019), an order of sentences (Chambers and Jurafsky, 2008; Barzilay and Lapata, 2008), topic introduction, co-reference, a sequence of events (Tomkins, 1978; Schank and Abelson, 2013), and more. In this work, we primarily focus on the effect of topical content in text planning.

Despite the recent advances of contextualized language models (Devlin et al., 2019; Radford et al., 2019), the lack of appropriate tasks makes it difficult to evaluate generation models' long-term coherence. Prior tasks fall into classification or ranking problems, such as narrative close task (Chambers and Jurafsky, 2008; Mostafazadeh et al., 2016), sentence ordering (Barzilay and Lapata, 2008), and next sentence prediction (Devlin et al., 2019). Some recent works focused on designing generation tasks: story generation (Fan et al., 2019), text infilling (Huang et al., 2019; Fedus et al., 2018; Hua and Wang, 2019), or paragraph bridging (Kang et al., 2019). However, most of them suffer from predicting appropriate topical content given limited context, due to the limited usage of self-supervision

[*]This work was done while DK was at CMU.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 6533–6543,
November 16–20, 2020.

signals from the paragraph.

This work proposes a new open-ended paragraph completion task; PARCOM; predicting the masked sentences in a paragraph. Unlike the prior works, our task uses two effective ways of self-supervision learnt from a written paragraph itself: (1) we augment more training instances via permutation masking and (2) resolve the context sparsity problem by providing a set of ground-truth content keywords and predicting them directly from context at testing time.

For the task, we propose a self-supervised text planner (SSPlanner) that explicitly predicts content keywords (content prediction) from context and guides the pretrained language model (surface-realization) using the predicted content. The distribution of predicted keywords is then combined with the distribution of words in the language model using copy mechanism (See et al., 2017). The predicted content keywords are an approximation of topical intents by the generator, providing a hint to guide the surface realizer to bridge the coherency gap between the given context and text to generate. Overall, SSPlanner combines two advantages; micro-level language fluency from the pre-trained language model (bottom-up) and macro-level content choice controlled by the macro-level planning (top-down). Our experiment shows that SSPlanner achieves significant improvements over the baselines in both automatic and human evaluation.

2 Related Work

We first categorize a wide range of long-term coherent generation tasks (Table 1), based on their inclusion relationship (**C**-**T**) between the given context (**C**) and target to predict (**T**).

C-T	Tasks	Content Selection	Content Planning	Content Ordering	Surface Realization
$\subset$	Data-to-text	×	×	✓	✓
$\supset$	Summarization	✓	×	△	✓
$\approx$	Paraphrasing	△	×	×	✓
$\perp\!\!\!\perp$	StoryGen, Text Infilling, Bridging, PARCOM (ours)	✓	✓	✓	✓

Table 1: Comparison of generation tasks by different inclusion relationships between **C**ontext and **T**arget.

C $\subset$ **T**: Data-to-text produces text from structured data (e.g., table). Moryossef et al. (2019); Puduppully et al. (2019); Shen et al. (2019); Miculicich et al. (2019) combine content planning with surface realization. However, since content is explicitly provided as a data form, the planner mostly orders and structures, not prediction.

C $\supset$ **T**: In abstractive summarization, all context information is entirely given in the source document, as a superset of target summaries to predict. Thus, generation only pays attention to abstracting the context into a shorter form instead of content prediction or ordering.

C $\approx$ **T**: Paraphrasing is transforming surface patterns of text while preserving its semantics. Fu et al. (2019) used variational autoencoders for surface realization with a latent bag of words model for differentiable content planning, where content to generate itself is given in context, not requiring any content planning.

C $\perp\!\!\!\perp$ **T**: Story generation (Fan et al., 2019), text infilling (Fedus et al., 2018; Huang et al., 2019), paragraph bridging (Kang et al., 2019), and our proposed PARCOM are very challenging tasks where context and target have no overlap (open-ended), but they should be coherently connected. Table 2 categorize various generation models applied on the open-ended tasks (C$\perp\!\!\!\perp$T), based on its self-supervision types:

Models ($\perp\!\!\!\perp$)	Bidirect. Flow	Permutation Masking	Content Guidance	Content Prediction
Keskar et al. (2019)	✓	✓	✓	×
Fan et al. (2019)	×	×	✓	×
Huang et al. (2019)	✓	✓	×	×
Hua and Wang (2019)	×	×	✓	×
Kang et al. (2019)	✓	×	×	×
SSPlanner (ours)	✓	✓	✓	✓

Table 2: Comparison of generation models in C$\perp\!\!\!\perp$T tasks by different self-supervision types.

Keskar et al. (2019) conditioned language models with topical words to control the target text. Fan et al. (2019) developed a surface realizer on anonymized entities using semantic role labeling. Hua and Wang (2019) used pre-extracted topics to guide a generator to produce stylized argumentation text. However, they are given the topical content as input (content guidance), while our SSPlanner directly predicts plan words from context (content prediction).

Fedus et al. (2018); Huang et al. (2019) developed various methods for text infilling task. Very similar to our task, Kang et al. (2019) developed language models informed by discourse relations on the bridging task; given the first and last sentences, predicting the intermediate sentences (bidirectional flow). However, they did not explicitly

predict content words given context nor use them as a self-supervision signal in training. Unlike random masking in Keskar et al. (2019); Huang et al. (2019), we propose a better data augmentation training method via permutation masking.

3 PARCOM: Paragraph Completion Task from Self-Supervision Signals

Our task is motivated by the recently proposed task; paragraph bridging (Kang et al., 2019), predicting intermediate sentences of a paragraph, given the first and the last sentences. To prevent generation becoming too divergent from the context in story or prompt generation (Fan et al., 2019), the bridging task restricts generation to end with the last sentence given, provided as an ending goal for generation.

However, in the bridging task, the objective is to generate text by coherently linking the two extreme sentences, making the task itself too challenging even for human[1]. For instance, the first and last sentences are too sparse to generate multiple (from 2 to 5) target sentences, increasing divergence of generation exponentially. Also, data usage in (Kang et al., 2019) is very inefficient; training a single instance per paragraph.

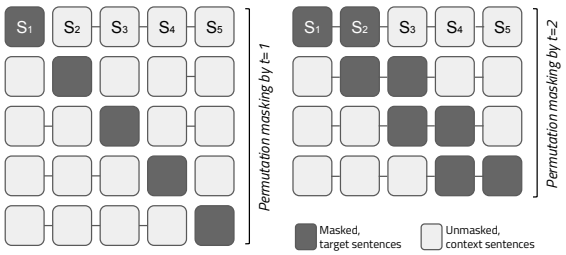
To address those issues, we propose a new paragraph completion task PARCOM by maximizing self-supervision presented in a paragraph itself (Figure 1). We describe two types of self-supervisions: (1) masking a fixed-length of consecutive sentences in any position over a paragraph to maximize usage of a paragraph and (2) extracting partial keywords of the masked text as plan keywords to resolve the content sparsity problem. Mainly, we learn the patterns between the context and the plan keywords in training and at testing time predict the plan keywords, and guide the surface generator (§4).

3.1 Data Augmentation via Permutation Masking

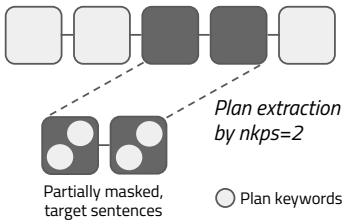
Our work is motivated by word masking, in training contextualized language models (Devlin et al., 2019), but extending it to *sentence-level* for learning longer coherence.

Let t be the number of targets, masked sentences to predict and c be the number of unmasked, context sentences given, where l=t+c is the total length

[1]METEOR score from human generation on the task is only about 4.5 (Kang et al., 2019)



(a) Sentence masking via permutation: t=1 (left) or t=2 (right): One paragraph has a total of 5+4=9 training instances.



(b) Plan extraction on target sentence. The maximum number of keywords per sentence ($nkps$=2) is given.

Figure 1: Paragraph completion (PARCOM) task: (a) predicting the masked, target sentences given the unmasked, context sentences and (b) each masked, target sentence is given a small number of keywords extracted from the original target sentence.

of a paragraph. For instance, in Figure 1, we have a l=5 length paragraph. We restrict the number of context sentences to be larger than the number of target sentences ($c > t$), to avoid context become too sparse. Also, we produce a total of 5+4=9 training instances, making use of data more efficient.

3.2 Denser Context by Plan Extraction

We provide extra partial information as a set of keywords to guide the surface generator. This is motivated by data-to-text tasks, but our plans are topical content instead of structured data.

We then question what types of plan keywords are the most effective for completing the paragraph. We extract keywords using various keyword extraction systems:

- `Off-the-shelf` systems extract keywords for each sentence using the three off-the-shelf systems: YAKE (Campos et al., 2020) using statistical features (e.g., TF, IDF), RAKE (Rose et al., 2010) using graph-based features (e.g., word degree), and PositionRank (Florescu and Caragea, 2017) using position-based PageRank. Then we choose duplicate keywords by majority voting.
- `Syntactic` features (e.g., part-of-speech tags, named entities (Fan et al., 2019), events

(Tomkins, 1978)) are often regarded as the most salient topical content in generation. Using off-the-shelf Part-of-speech (PoS) tagger[2], we extract three types of syntactic features: *nouns*, *verbs*, and *nouns+verbs*.

- `Attention` weights are used to capture context-aware keywords. We use the pre-trained BERT (Devlin et al., 2019) to encode context and target text, then average the attention weights of context words with respect to each target word. We only use the first head's attentions, then average them over all 12 layers[3]. We finally choose words with the maximum weight except for the special tokens (e.g., [CLS]) and punctuation marks.

We set the maximum number of keywords per sentence (*nkps*) to 5. Some extractors output an empty keyword list, so the number of keywords across the systems is different. Our keywords are always uni-grams. In case they are not uni-grams, we split them by whitespaces and use individual unigrams as unique keywords. If the target text has multiple sentences, we combine all keywords from the sentences and randomly shuffle them. The plan keywords extracted are only provided while training our plan predictor, but not at test time. At testing time, we explicitly predict the keywords given context.

4 Self-supervised Text Planning (SSPlanner)

SSPlanner has various self-supervision modules that learn coherence signals from a paragraph itself: **surface realizer** (language model) by learning from a sequence of words, **next sentence predictor** by learning from a sequence of two consecutive sentences, **sentence position embeddings** by learning from an order of context sentences, **plan predictor** by learning from the relationship between the given context and important keywords used in the generation of the target text, and **content guidance** by learning from whether the predicted plan keywords are used or not in the target (See Figure 2).

Our planner is motivated by the two-stage generation framework (Moryossef et al., 2019; Miculicich et al., 2019; Fu et al., 2019; Hua and Wang, 2019). While in prior works, the content is explicitly given from the dataset or task itself, our plan predictor in SSPlanner predicts the plan keywords only from the given context, by learning the topical relationship between context and content in target from training data.

Given l length of a paragraph $s_1..s_l$ where each sentence s consists of a n number of words $s = w_1..w_n$, PARCOM splits it into the context sentences $\mathtt{x}{=}s_1..s_{j-1}, s_{j+t}..s_n$ and t target sentences to predict $\mathtt{y}{=}s_j..s_{j+t-1}$. For each target sentence, p number of plan keywords $k_{j,1}..k_{j,p}$ for arbitrary target sentence s_j are given only at training time. The plan keywords are chosen from the entire vocabulary $\mathbb{V}^W$ and later combined with word distribution from the language model. We describe each self-supervision module in SSPlanner as follows:

Surface realization with pre-trained language models. We use two different types of transformer-based language models: BERT (Devlin et al., 2019) and GPT2 (Radford et al., 2019). While GPT2 is trained on bidirectionally tied language modeling, BERT is trained on masked language modeling. For BERT, we use the sequential sampling method (Wang and Cho, 2019). Using them, we encode context $\mathtt{x}$ and output the hidden representation $h_{j,i} = \mathtt{f}(h_{j-1,i}, \mathtt{x}_{k<(j,i)})$ for j^{th} word in i^{th} sentence, where $\mathtt{f} \in \{\text{BERT, GPT2}\}$ is the transformer language model. We then output the sentence vector h_i by averaging all word vectors in a sentence.

Sentence position embedding. We concatenate the encoded sentence representation with its sentence position embedding. By adding the sentence position embeddings into context encoding, the model is aware of where the context sentence came from (e.g., from the first or last). Compared to the simple concatenation of them (Kang et al., 2019), our sentence position embedding helps better learn the bi-directional coherence. The context vector's final representation is then $h^c = \frac{1}{n}\sum_i h_i; \mathtt{pos}^c_i$ where n is the number of sentences in a text and $\mathtt{pos}^c_i$ is the position embedding of i^{th} sentence in the context paragraph.

Plan prediction. This work assumes that high-level plan words consist of bag-of-words (Fu et al., 2019), so that the model directly predicts the plan keywords from the vocabulary used in surface realization. We calculate the plan probabilities over the entire vocabularies $\mathbb{V}$ given the context vector h^c and choose the p number of keywords with maximum probability estimates over vocabulary: $\hat{p}_{k\in\mathbb{V}} = \mathtt{softmax}(h^c W^{cv} \mathbb{V})$ where $\mathbb{V}$ is the vocabu-

[2] https://spacy.io/

[3] Vig (2019) observed that which layers or heads are important for syntactic and semantic tasks.

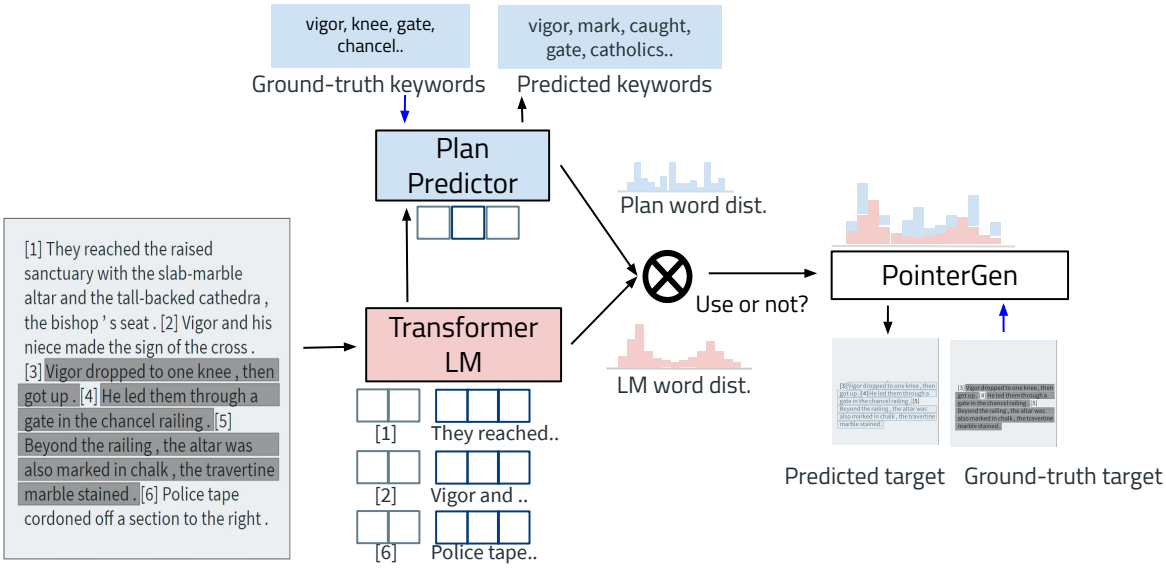


Figure 2: SSPlanner: first predicts high-level plan keywords (Plan Predictor) then guides the surface generation (Transformer LM) using the predicted plan keywords. The ground-truth plan keywords and target sentences (blue arrows) are only given in training time, whereas not in testing time. The predicted and ground-truth target can be seen in Table 7. Best viewed in color.

lary from the training data and W^{cv} is the trainable model parameter. We do not control any explicit cut-off in the $p_{k\in\mathbb{V}}$ in order to make the distribution differentiable. The objective is then:

$$\mathbb{L}_{\texttt{plan}} = -\sum_{k\in\mathbb{V}} \log p^*_k \log \hat{p}_k \quad (1)$$

where the loss is calculated by cross-entropy, $\hat{p}$ is the estimated probability distribution over vocabulary and $p*$ is the true one-hot distribution over plan keywords extracted from the extraction algorithms (i.e., [0,1..0,1] over $\mathbb{V}$).

Next sentence prediction. Motivated by Devlin et al. (2019), we also add an auxiliary task of predicting whether the target sentence is related to context or not. For negative samples, PARCOM assigns 50% of random target sentences. We optimize $\hat{p}_{next} = softmax(W^c h^c)$ where W^c is the trainable parameter for the binary classification. Next sentence prediction's objective is then:

$$\mathbb{L}_{\texttt{next}} = -\sum_{j} p^*_{next} \log \hat{p}_{next} \quad (2)$$

where the loss is calculated by binary cross-entropy, p^*_{next} is the true label for next sentences and $\hat{p}_{next}$ is the predicted label.

Content guidance. We combine two distributions between plan predictions and language modeling through copy mechanism following the pointer-generator (See et al., 2017). For j^{th} sentence, we learn the probability of choosing the plan keyword or the word from language modeling based on context vectors, plan keyword distributions, and sentence position embedding of target sentences: $\texttt{P}_{plan}(v_k) = \sigma(\texttt{W}^{ck}[h^c; \hat{p}_k; \texttt{pos}^t_j])$, where σ is a sigmoid function, $\texttt{W}^{ck}$ is the trainable parameter, and $\texttt{v} \in [0,1]$ is a probability of whether choosing the plan keyword or not.

We then decode each target sentence using the same language model decoder: $s_j = \texttt{g}(s_{j-1}, \hat{y}_{j-1})$, where $\texttt{g} \in \{\text{BERT, GPT2}\}$ is the language model decoder and s is its output hidden state. We can obtain the attention over plan keywords k:

$$\alpha^{plan}_k = \texttt{softmax}(\hat{p}_k \texttt{W}^{kj}[s_j; \texttt{pos}^t_j]) \quad (3)$$

where $\texttt{W}^{kj}$ is the trainable parameter. Lastly, we combine the distribution of plan probabilities $\texttt{P}_{plan}$ and word probabilities in decoding $\texttt{P}_{lm}$.

$$P(y) = \texttt{P}_{plan} \sum_k (\alpha^{plan}_k) + (1 - \texttt{P}_{plan}) P_{lm}(y) \quad (4)$$

The objective of the pointer-generator is then:

$$\mathbb{L}_{\texttt{gen}} = -\sum_{i\in\texttt{t}, j=1..n} P(\hat{y}_{i,j}) \log P(y^*_{i,j}) \quad (5)$$

Final objective. The final objective of our training is to minimize the three objectives; plan prediction, next sentence prediction, and pointer generation, together:

$$\mathbb{L}_{SPP} = \lambda_{plan}\mathbb{L}_{\texttt{plan}} + \lambda_{next}\mathbb{L}_{\texttt{next}} + \mathbb{L}_{\texttt{gen}} \quad (6)$$

where the weighting terms; λ_{plan} and λ_{next}, are obtained through the cross-validation.

5 Experiment

We answer three questions in our experiments: Q1. Does SSPlanner help produce a more coherent generation in PARCOM? If so, which of the self-supervision modules are the most helpful? Q2. What types of plan keywords (e.g., noun, verb, attention) are most effective in terms of generation quality? How many keywords given are the most helpful? Q3. Is PARCOM a valid generation task to measure text coherence?

Dataset	Domain	#Sent.	#Para.	Length	#Inst.	#Keyw.
Fantasy	book	1.6M	352K	4.7	21M	3.2-19M
Romance	book	5.3M	1.1M	4.6	67M	27-62M
WikiText	wiki	510K	3.3M	6.5	82M	14M-78M
CNNDM	news	12M	311K	39.3	246M	63-315M

Table 3: Data statistics: domain of text, the number of sentences, the number of paragraphs, the averaged length (number of sentences) of paragraph, the number of training instances permuted from the paragraphs, and minimum to maximum number of keywords extracted.

Paragraph datasets. Table 3 shows the paragraph datasets collected for our experiment. We collect paragraphs from various domains: the two most frequent sub-genres extracted from BookCorpus (Zhu et al., 2015) dataset; `Fantasy` and `SciFi`, Wikipedia text from `wikiText`-103 (Merity et al., 2016), and news articles from CNN/DailyMail (`CNNDM`) dataset (See et al., 2017). `CNNDM` and `WikiText` contain factual knowledge about events or things, whereas `Fantasy` and `Romance` are more narrative.

For a fair comparison, we restrict the number of sentences in a paragraph from 4 to 7, the same as the setup in Kang et al. (2019). Since `CNNDM` has no specific line breakers in the document, each document is regarded as a single paragraph (39.3 lengths on average). Each dataset is randomly split by 0.9/0.05/0.05 for the train, valid, and test set, respectively.

Models. As baselines, we compare non-pretrained sequence-to-sequence models: **BiLSTM** (Hochreiter and Schmidhuber, 1997) and hierarchical seq2seq **HRED** (Serban et al., 2017; Sordoni et al., 2015) by encoding the concatenation of context sentences and then decoding the target sentences. We also compare two strong paragraph generation models: **FlowNet**$_{disc}$ using discourse relations and **FlowNet**$_{latent}$ using latent delta relations (Kang et al., 2019), following the same setups (e.g., discourse parser, hyper-parameters) of the original paper.

Also, we use the pre-trained language model baselines fine-tuned on our paragraph datasets: the fine-trained bert-base-uncased (**BERT**$_{finetune}$) and gpt2-base (**GPT2**$_{finetune}$) models (Wolf et al., 2019). For BERT, we use the sequential sampling method (Wang and Cho, 2019) with Nucleus sampling strategies for producing more diverse text (Holtzman et al., 2019).

Our proposed method **SSPlanner** is trained using either bert-base-uncased or gpt2-base. As an upper-bound of our method, we predict masked, target text using the ground-truth plan keywords $\hat{p}$.

We find the best hyper-parameters on the validation set using a grid search on the learning rate, the number of training epochs, sampling parameters, and so on. We follow the default parameters used in the HuggingFace's transformer models (Wolf et al., 2019). For a pointer-generator, we follow the default parameters in (See et al., 2017). The maximum number of plan keywords per sentence is 3. For more details, see the Appendix.

Metrics. We evaluate our models using both automatic metrics and human evaluation: For automatic metrics, we use two hard metrics: BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005), as well as an embedding similarity metric to capture the semantic similarity: Vector Extrema (VE) (Liu et al., 2016).

For human evaluation, we measure fluency, coherence with respect to context, and overall quality with 1-5 Likert scale. We randomly select 100 samples from the test set in each `Romance`, `WikiText`, and `CNNDM` (total 300 paragraphs). Each sample is annotated by three crowd-workers then averaged. We also measure how human performs on the task by asking workers to predict the masked text in these 300 paragraphs.

5.1 Automatic and Human Evaluation

Table 4 and 5 show automatic and human evaluation result on PARCOM task. The fine-tuned models ({BERT,GPT2}$_{finetune}$)[4] and FlowNet models show significant improvements over the seq2seq

[4]In our experiment, no fine-tuned models (original pre-trained models) show very poor performance on our task.

Models	Fantasy			Romance			WikiText			CNNDM		
	B	**M**	**VE**	**B**	**M**	**VE**	**B**	**M**	**VE**	**B**	**M**	**VE**
BiLSTM (Hochreiter and Schmidhuber, 1997)	2.6	2.9	26.5	2.2	2.4	25.6	2.7	2.9	31.2	2.5	2.5	30.0
HRED (Sordoni et al., 2015)	2.8	2.9	25.2	2.4	2.5	28.2	2.8	2.7	32.4	2.8	2.9	31.9
FlowNet$_{disc}$ (Kang et al., 2019)	3.6	4.5	41.4	3.2	3.7	38.6	3.2	3.6	38.9	3.4	3.8	40.1
FlowNet$_{latent}$ (Kang et al., 2019)	3.7	4.5	42.8	3.1	3.6	35.2	3.1	3.5	37.5	3.3	3.7	38.7
BERT$_{finetune}$ (Devlin et al., 2019)	3.7	4.6	38.5	4.1	4.4	42.8	4.2	4.7	48.9	4.4	4.8	47.5
GPT2$_{finetune}$ (Radford et al., 2019)	3.9	5.0	42.8	4.3	4.7	48.5	4.6	4.8	50.1	4.5	5.0	50.2
SSPlanner (BERT)	5.7	6.7	57.0	5.9	6.8	54.0	6.1	6.4	54.3	6.4	6.9	57.0
SSPlanner (GPT2)	**7.1**	**9.2**	**69.5**	**7.2**	**8.1**	**73.9**	**7.6**	**7.7**	**66.8**	**6.9**	**7.8**	**59.9**
SSPlanner (GPT2) \w $\hat{p}$	11.1	11.8	79.6	12.7	13.3	84.4	12.5	13.0	87.8	12.1	12.9	84.9

Table 4: Automatic evaluation. **B** is BLEU, **M** is METEOR, and **VE** is vector extrema. For all metrics, the higher the better. SSPlanner used keywords from the off-the-shelf system for training. $\hat{p}$ is the ground-truth plan keywords extracted from the off-the-shelf system.

Models	Romance			WikiText			CNNDM		
	F	**C**	**Q**	**F**	**C**	**Q**	**F**	**C**	**Q**
GPT2$_{finetune}$	4.4	2.1	3.6	3.9	1.9	1.9	3.8	1.6	1.8
SSPlanner	4.2	3.8	3.9	3.8	3.6	3.8	3.6	3.1	3.2
SSPlanner \w $\hat{p}$	4.1	4.6	4.3	3.8	4.1	4.0	4.0	4.4	4.1
Human	4.8	4.9	4.9	4.6	4.5	4.5	4.5	4.4	4.4

Table 5: Human evaluation. **F** is fluency, **C** is coherence with context, and **Q** is overall quality. Each metric is scaled out of 5.

Models	Romance		WikiText		CNNDM	
	NSP	**PP**	**NSP**	**PP**	**NSP**	**PP**
SSPlanner	91.6	48.1	92.7	50.2	90.7	49.4

Table 6: Accuracies of each self-supervision module in SSPlanner. **NSP** is next sentence prediction, and **PP** is plan prediction.

baselines (BiLSTM and HRED) by large margins (∼1.5 METEOR), showing the importance of fine-tuning on target text and modeling inter-sentential relation, respectively.

In all datasets, SSPlanner shows significant improvements in both hard and soft metrics. This indicates that explicitly predicting content words before surface realization helps generate more coherence text on target-oriented generation in PARCOM. SSPlanner with GPT2 outperforms SSPlanner with BERT, because such autoregressive models like GPT2 are more appropriate for our task, whereas BERT is not. Finally, the performance of SSPlanner with the ground-truth keywords ($\hat{p}$) achieves the dramatic gain, which can be seen as an upper bound of our planning framework. Among domains, `Fantasy` and `Romance` seem to be better predicted compared to `WikiText` and `CNNDM` that require additional factual knowledge as well as narrative coherence.

Using the best model; **SSPlanner** (GPT2), we conduct a human evaluation on various system outputs and human-generated text (Table 5). The fine-tuned GPT2 model shows high fluency as itself but very low coherence with context, because PARCOM requires not only fluent and natural text but also context-aware text. SSPlanner achieves much higher coherence and overall quality than the baselines, but still is far behind the upper-bound model (SSPlanner with $\hat{p}$) and human generation.

5.2 Performance of Self-supervision Modules

We measure performance (i.e., accuracy) of each self-supervision module in SSPlanner: next sentence prediction (NSP) and plan prediction (PP) on the test samples (Table 6). SSPlanner achieves very high accuracy in NSP. In PP, SSPlanner correctly predicts almost half of the keywords from the total vocabulary size, indicating that the plan prediction module in SSPlanner can capture a certain level of coherence between the given context and target text to predict, although it is not perfect.

5.3 Comparison of Self-supervision Modules and Keyword Types in Training

Table 8 shows ablation on self-supervision modules. All scores are macro-averaged on three datasets: `Romance`, `WikiText`, and `CNNDM`. Each module helps improve the overall performance (METEOR): plan prediction (+2.1 M), sentiment positional embedding (+1.4 M), and next sentence prediction (+0.4 M),

Among the different types of keywords used in training (Table 9), the combination of nouns and verbs and the keywords extracted from the off-the-

Task: given context sentences ⓪, ①, ⑤, predict target sentences ②, ③, ④

⓪ "They reached the raised sanctuary with the slab-marble altar and the tall-backed cathedra , the bishop ' s seat ." ① "Vigor and his niece made the sign of the cross ." ② "Vigor dropped to one knee , then got up ." ③ "He led them through a gate in the chancel railing ." ④ "Beyond the railing , the altar was also marked in chalk , the travertine marble stained ." ⑤ "Police tape cordoned off a section to the right ."

Plan keywords extracted from target sentences using different systems:

`Off-the-shelf`	② (vigor, dropped, one), ③ (chancel, railing, led), ④ (travertine, marble, stained)
`Syntactic` (noun)	② (vigor, knee), ③ (gate, chancel), ④ (railing, altar, chalk)
`Syntactic` (verb)	② (dropped, got), ③ (led, railing), ④ (marked, stained)
`Syntactic` (nounverb)	② (vigor, dropped, knee), ③ (led, gate, chancel), ④ (railing, altar, marked)
`Attention`	② (vigor, dropped, got), ③ (led, gate, railing), ④ (altar, chalk, travertine)

	SSPlanner	**Human writer**
Human eval.	F : 4.3, C: 3.9, Q: 3.8	F : 4.8, C: 4.9, Q: 4.8
Predicted plan keywords	② (vigor, mark, caught), ③ (gate, catholics, police), ④ (altar, mark, bishop)	② (vigor, show, sanctuary), ③ (altar, blood, trace), ④ (kill, sacrifice, recently)
Predicted target sentences	② "vigor continuously walked down the road ." ③ "he opened the gate which has a sign of catholics ." ④ "both bishop and vigor met a police officer ."	② "Then vigor showed around the sanctuary to them." ③ "In there, they found a trace of the blood on the altar." ④ "They thought that recently the sacrifice was killed in here."

Table 7: Example paragraph with the plan keywords extracted from different algorithms and output predictions by SSPlanner and human writer. **F** is fluency, **C** is coherence with context, and **Q** is overall quality.

Models	**M**	**VE**
SSPlanner	7.9	66.6
- Sentence Position (SP)	-1.4	-8.1
- Plan Prediction (PP)	-2.1	-13.2
- Next Sentence Prediction (NSP)	-0.5	-3.6

Table 8: Ablation on self-supervision modules.

Models	**M**	**VE**
\w Random	6.1	54.0
\w Syntac(Verb)	7.6	63.7
\w Syntac(Noun)	7.5	62.2
\w Syntac(N+V)	**8.0**	**66.3**
\w Off-the-shelf	7.8	66.8
\w Attention	7.6	63.6

Table 9: Comparison of plan keyword types at training (right). All scores are macro-averaged on three datasets: `Romance`, `WikiText`, and `CNNDM`.

shelf algorithm outperform the other types. We conjecture that since a sentence consists of both entities (i.e., nouns) and events (i.e., verbs) according to the script theory (Schank and Abelson, 2013), the combination of them provides the largest amount of information to complete the sentence. Attention-based keywords are not that helpful because the averaged attention weights themselves may not be a good indicator for topical coherence.

5.4 Comparison of Keyword Types and Ratios in Testing

In Figure 3, at test time, the predicted keywords from SSPlanner (red) shows dramatic improvements in both METEOR and VE against the random keywords (blue), but far behind the ground-truth keywords (yellow). As more predicted keywords are used at testing time, the generation quality increases. We include the full table of our ablation tests over the three datasets in the Appendix.

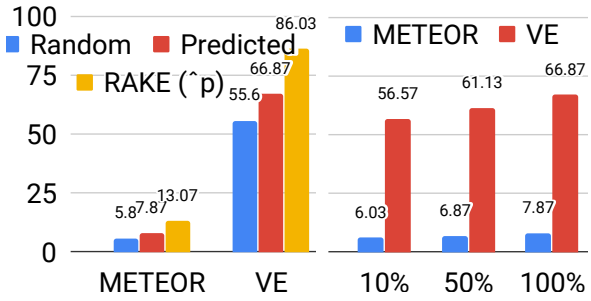


Figure 3: Comparison of plan keyword types (left) and plan keyword ratios used (right) in testing. Best viewed in color.

Table 7 shows an example paragraph with

ground-truth keywords extracted from different algorithms in PARCOM and predicted target sentences with plan keywords by SSPlanner and a human writer. In the prediction by SSPlanner, half of the predicted keywords are used in the generation, making the story more coherent to the first two sentences and the last ending sentence. In the entire test set, we observe that about 43% of predicted keywords are actually used in generation.

6 Conclusion

A written paragraph itself contains various inductive coherence signals to be learned through *self-supervision*. Motivated by this, we propose a paragraph completion task for measuring textual coherence from a long document using different types of self-supervision signals. To solve the task, we propose a text planner SSPlanner that explicitly predicts topical content keywords, and then guides the surface generator using the predicted plan keywords. SSPlanner consists of different kinds of self-supervision modules: sentence positions, a sequence of words or sentences, and the topical relationship between context and target. Our self-supervised planning, in addition to other types of planning (e.g., discourse, goals, coreference, tenses) can be an important step toward modeling a long-term coherence in text generation.

Our results suggest several promising directions: Although our ablation tests show the effect of each self-supervision module, types of plan keywords, and the amount of keywords with respect to generation quality, there are more spaces to explore in self-supervised text planning. For example, one can study the generation quality with respect to the position of the target sentences (beginning, middle, end), the comparison of plan keywords predicted by human and system, the effect of data augmentation by their positions (e.g., masking the only middle), the generation quality with respect to the ratio between masked and unmasked sentences, and more.

Second, we can extend the set of plan keywords to be more structured like a discourse tree. For instance, one can write a simple structure like "(CAUSALITY (ELABORATE (Buy, Coffee)) (Pay, Tip, 12 dollars))" then the system can generate a long, coherent text reflected by the structure. Predicting such structural plans from context and imposing them into the generator would be a potential direction for future work.

Last, text planning is a cognitive function commonly used in human language generation. To generate more human-like utterances, different planning stages should be simultaneously combined together (Kang, 2020), such as abstractive planning, strategic planning, coherence planning, and diversity planning. Combining the heterogeneous planning systems will be a crucial step towards developing a human-like language generation.

Acknowledgements

We thank Dery Lucio, Hiroaki Hayashi, Taehee Jung, Edvisees members at CMU, Hearst lab members at UC Berkeley, and anonymous reviewers at EMNLP 2020 for their helpful comments.

References

Douglas E Appelt. 1982. Planning natural-language utterances to satisfy multiple goals. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

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

Donn Byrne. 1979. *Teaching writing skills*. Longman.

Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257–289.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

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

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Corina Florescu and Cornelia Caragea. 2017. Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115.

Yao Fu, Yansong Feng, and John P Cunningham. 2019. Paraphrase generation with latent bag of words. In *Advances in Neural Information Processing Systems*, pages 13623–13634.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

Eduard H Hovy. 1990. Pragmatics and natural language generation. *Artificial Intelligence*, 43(2):153–197.

Eduard H Hovy. 1991. Approaches to the planning of coherent text. In *Natural language generation in artificial intelligence and computational linguistics*, pages 83–102. Springer.

Xinyu Hua and Lu Wang. 2019. Sentence-level content planning and style specification for neural text generation. 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 591–602, Hong Kong, China. Association for Computational Linguistics.

Yichen Huang, Yizhe Zhang, Oussama Elachqar, and Yu Cheng. 2019. Inset: Sentence infilling with intersentential generative pre-training. *arXiv preprint arXiv:1911.03892*.

Dongyeop Kang. 2020. *Linguistically Informed Language Generation: A Multifaceted Approach*. PhD dissertation, Carnegie Mellon University.

Dongyeop Kang, Hiroaki Hayashi, Alan W Black, and Eduard Hovy. 2019. Linguistic versus latent relations for modeling coherent flow in paragraphs. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Hong Kong.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, volume 36, pages 495–503.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Kathleen R McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1–41.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Lesly Miculicich, Marc Marone, and Hany Hassan. 2019. Selecting, planning, and rewriting: A modular approach for data-to-document generation and translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 289–296, Hong Kong. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. 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 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6908–6915.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Roger C Schank and Robert P Abelson. 2013. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow, and Satoshi Sekine. 2019. Select and attend: Towards controllable content selection in text generation. *arXiv preprint arXiv:1909.04453*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Judith A. Swan. 2002. The science of scientific writing. In *Book*.

Silvan S Tomkins. 1978. Script theory: Differential magnification of affects. In *Nebraska symposium on motivation*. University of Nebraska Press.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. *arXiv preprint arXiv:1906.05714*.

Alex Wang and Kyunghyun Cho. 2019. Bert has a mouth, and it must speak: Bert as a markov random field language model. *arXiv preprint arXiv:1902.04094*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.