Long and Diverse Text Generation with Planning-based Hierarchical Variational Model

Zhihong Shao[1], Minlie Huang[1]*, Jiangtao Wen[1], Wenfei Xu[2], Xiaoyan Zhu[1]
[1] Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems
[1] Beijing National Research Center for Information Science and Technology
[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2] Baozun, Shanghai, China
szh19@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn
jtwen@tsinghua.edu.cn, xuwenfeilittle@gmail.com
zxy-dcs@tsinghua.edu.cn

Abstract

Existing neural methods for data-to-text generation are still struggling to produce long and diverse texts: they are insufficient to model input data dynamically during generation, to capture inter-sentence coherence, or to generate diversified expressions. To address these issues, we propose a Planning-based Hierarchical Variational Model (PHVM). Our model first plans a sequence of groups (each group is a subset of input items to be covered by a sentence) and then realizes each sentence conditioned on the planning result and the previously generated context, thereby decomposing long text generation into dependent sentence generation sub-tasks. To capture expression diversity, we devise a hierarchical latent structure where a global planning latent variable models the diversity of reasonable planning and a sequence of local latent variables controls sentence realization. Experiments show that our model outperforms state-of-the-art baselines in long and diverse text generation.

1 Introduction

Data-to-text generation is to generate natural language texts from structured data (Gatt and Krahmer, 2018), which has a wide range of applications (for weather forecast, game report, product description, advertising document, etc.). Most neural methods focus on devising encoding scheme and attention mechanism, namely, (1) exploiting input structure to learn better representation of input data (Lebret et al., 2016; Liu et al., 2018), and (2) devising attention mechanisms to better employ input data (Mei et al., 2016; Liu et al., 2018; Nema et al., 2018) or to dynamically trace which part of input has been covered in generation (Kiddon et al., 2016). These models are able to produce fluent and coherent short texts in some applications.

However, to generate long and diverse texts such as product descriptions, existing methods are still unable to capture the complex semantic structures and diversified surface forms of long texts. **First**, existing methods are not good at modeling input data dynamically during generation. Some neural methods (Kiddon et al., 2016; Feng et al., 2018) propose to record the accumulated attention devoted to each input item. However, these records may accumulate errors in representing the state of already generated prefix, thus leading to wrong new attention weights. **Second**, inter-sentence coherence in long text generation is not well captured (Wiseman et al., 2017) due to the lack of high-level planning. Recent studies propose to model planning but still have much space for improvement. For instance, in (Puduppully et al., 2019) and (Sha et al., 2018), planning is merely designed for ordering input items, which is limited to aligning input data with the text to be generated. **Third**, most methods fail to generate diversified expressions. Existing data-to-text methods inject variations at the conditional output distribution, which is proved to capture only low-level variations of expressions (Serban et al., 2017).

To address the above issues, we propose a novel Planning-based Hierarchical Variational Model (PHVM). To better model input data and alleviate the inter-sentence incoherence problem, we design a novel planning mechanism and adopt a compatible hierarchical generation process, which mimics the process of human writing. Generally speaking, to write a long text, a human writer first arranges contents and discourse structure (i.e., *high-level planning*) and then realizes the surface form of each individual part (*low-level realization*). Motivated by this, our proposed model first performs

*Corresponding author: Minlie Huang.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3257–3268, Hong Kong, China, November 3–7, 2019.

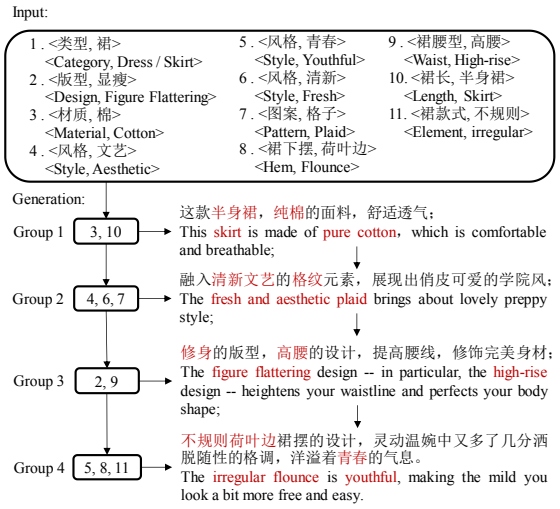

Figure 1: Generation process of PHVM. After encoding a list of input attribute-value pairs, PHVM first conducts planning by generating a sequence of groups, each of which is a subset of input items. Each sentence is then realized conditioned on the corresponding group and its previous generated sentences.

planning by segmenting input data into a sequence of groups, and then generates a sentence conditioned on the corresponding group and preceding generated sentences. In this way, we decompose long text generation into a sequence of dependent sentence generation sub-tasks where each sub-task depends specifically on an individual group and the previous context. By this means, the input data can be well modeled and inter-sentence coherence can be captured. Figure 1 depicts the process.

To deal with expression diversity, this model also enables us to inject variations at both high-level planning and low-level realization with a hierarchical latent structure. At high level, we introduce a global planning latent variable to model the diversity of reasonable planning. At low level, we introduce local latent variables for sentence realization. Since our model is based on Conditional Variational Auto-Encoder (CVAE) (Sohn et al., 2015), expression diversity can be captured by the global and local latent variables.

We evaluate the model on a new advertising text[1] generation task which requires the system to generate a long and diverse advertising text that covers a given set of attribute-value pairs describing a product (see Figure 1). We also evaluate our model on the recipe text generation task from (Kiddon et al., 2016) which requires the system to correctly use the given ingredients and maintain coherence among cooking steps. Experiments on advertising text generation show that our model outperforms state-of-the-art baselines in automatic and manual evaluation. Our model also generalizes well to long recipe text generation and outperforms the baselines. Our contributions are twofold:

- We design a novel Planning-based Hierarchical Variational Model (PHVM) which integrates planning into a hierarchical latent structure. Experiments show its effectiveness in coverage, coherence, and diversity.
- We propose a novel planning mechanism which segments the input data into a sequence of groups, thereby decomposing long text generation into dependent sentence generation sub-tasks. Thus, input data can be better modeled and inter-sentence coherence can be better captured. To capture expression diversity, we devise a hierarchical latent structure which injects variations at both high-level planning and low-level realization.

2 Related Work

Traditional methods (Reiter and Dale, 1997; Stent et al., 2004) for data-to-text generation consist of three components: content planning, sentence planning, and surface realization. Content planning and sentence planning are responsible for what to say and how to say respectively; they are typically based on hand-crafted (Kukich, 1983; Dalianis and Hovy, 1993; Hovy, 1993) or automatically-learnt rules (Duboue and McKeown, 2003). Surface realization generates natural language by carrying out the plan, which is template-based (McRoy et al., 2003; van Deemter et al., 2005) or grammar-based (Bateman, 1997; Espinosa et al., 2008). As these models are shallow and the two stages (planning and realization) often function separately, traditional methods are unable to capture rich variations of texts.

Recently, neural methods have become the mainstream models for data-to-text generation due to their strong ability of representation learning and scalability. These methods perform well in generating weather forecasts (Mei et al., 2016) or very short biographies (Lebret et al., 2016; Liu

[1] An advertising text describes a product with attractive wording. The goal of writing such texts is to advertise a product and attract users to buy it.

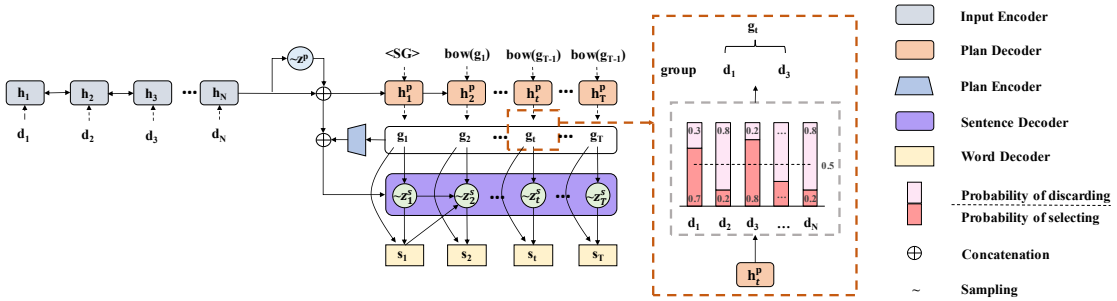

Figure 2: Architecture of PHVM. The model controls planning with a global latent variable z^p. The plan decoder conducts planning by generating a sequence of groups $g = g_1 g_2 ... g_T$ where g_t is a subset of input items and specifies the content of sentence s_t to be generated. The sentence decoder controls the realization of s_t with a local latent variable z_t^s; dependencies among z_t^s are explicitly modeled to better capture inter-sentence coherence.

et al., 2018; Sha et al., 2018; Nema et al., 2018) using well-designed data encoder and attention mechanisms. However, as demonstrated in Wiseman et al. (2017) (a game report generation task), existing neural methods are still problematic for long text generation: they often generate incoherent texts. In fact, these methods also lack the ability to model diversity of expressions.

As for long text generation, recent studies tackle the incoherence problem from different perspectives. To keep the decoder aware of the crucial information in the already generated prefix, Shao et al. (2017) appended the generated prefix to the encoder, and Guo et al. (2018) leaked the extracted features of the generated prefix from the discriminator to the generator in a Generative Adversarial Nets (Goodfellow et al., 2014). To model dependencies among sentences, Li et al. (2015) utilized a hierarchical recurrent neural network (RNN) decoder. Konstas and Lapata (2013) proposed to plan content organization with grammar rules while Puduppully et al. (2019) planned by reordering input data. Most recently, Moryossef et al. (2019) proposed to select plans from all possible ones, which is infeasible for large inputs.

As for diverse text generation, existing methods can be divided into three categories: enriching conditions (Xing et al., 2017), post-processing with beam search and rerank (Li et al., 2016), and designing effective models (Xu et al., 2018). Some text-to-text generation models (Serban et al., 2017; Zhao et al., 2017) inject high-level variations with latent variables. Variational Hierarchical Conversation RNN (VHCR) (Park et al., 2018) is a most similar model to ours, which also adopts a hierarchical latent structure. Our method differs from VHCR in two aspects: (1) VHCR has no planning mechanism, and the global latent variable is mainly designed to address the KL collapse problem, while our global latent variable captures the diversity of reasonable planning; (2) VHCR injects distinct local latent variables without direct dependencies, while our method explicitly models the dependencies among local latent variables to better capture inter-sentence connections. Shen et al. (2019) proposed ml-VAE-D with multi-level latent variables. However, the latent structure of ml-VAE-D consists of two global latent variables: the top-level latent variable is introduced to learn a more flexible prior of the bottom-level latent variable which is then used to decode a whole paragraph. By contrast, our hierarchical latent structure is tailored to our planning mechanism: the top level latent variable controls planning results and a sequence of local latent variables is introduced to obtain fine-grained control of sentence generation sub-tasks.

We evaluated our model on a new advertising text generation task which is to generate a long and diverse text that covers all given specifications about a product. Different from our task, the advertising text generation task in (Chen et al., 2019) is to generate personalized product description based on product title, product aspect (e.g., "appearance"), and user category.

3 Task Definition

Given input data $x = \{d_1, d_2, ..., d_N\}$ where each d_i can be an attribute-value pair or a keyword, our task is to generate a long and diverse text $y = s_1 s_2 ... s_T$ (s_t is the t^{th} sentence) that covers x as much as possible. For the advertising

text generation task, x consists of specifications about a product where each d_i is an attribute-value pair $< a_i, v_i >$. For the recipe text generation task, x is an ingredient list where each d_i is an ingredient. Since the recipe title r is also used for generation, we abuse the symbol x to represent $< \{d_1, d_2, ..., d_N\}, r >$ for simplification.

4 Approach

4.1 Overview

Figure 2 shows the architecture of PHVM. PHVM first samples a global planning latent variable z^p based on the encoded input data; z^p serves as a condition variable in both planning and hierarchical generation process. The plan decoder takes z^p as initial input. At time step t, it decodes a group g_t which is a subset of input items (d_i) and specifies the content of the t^{th} sentence s_t. When the plan decoder finishes planning, the hierarchical generation process starts, which involves the high-level sentence decoder and the low-level word decoder. The sentence decoder models inter-sentence coherence in semantic space by computing a sentence representation h_t^s and sampling a local latent variable z_t^s for each group. h_t^s and z_t^s, along with g_t, guide the word decoder to realize the corresponding sentence s_t.

The planning process decomposes the long text generation task into a sequence of dependent sentence generation sub-tasks, thus facilitating the hierarchical generation process. With the hierarchical latent structure, PHVM is able to capture multi-level variations of texts.

4.2 Input Encoding

We first embed each input item d_i into vector $\boldsymbol{e}(d_i)$. The recipe title r is also embedded as $\boldsymbol{e}(r)$. We then encode x[2] with a bidirectional Gated Recurrent Unit (GRU) (Cho et al., 2014). For advertising text generation, x is represented as the concatenation of the last hidden states of the forward and backward GRU $\boldsymbol{enc}(x) = [\overrightarrow{h_N}; \overleftarrow{h_1}]$; for recipe text generation, $\boldsymbol{enc}(x) = [\overrightarrow{h_N}; \overleftarrow{h_1}; \boldsymbol{e}(r)]$. $h_i = [\overrightarrow{h_i}; \overleftarrow{h_i}]$ is the context-aware representation of d_i. Note that input encoder is not necessarily an RNN; other neural encoders or even other encoding schemes are also feasible, such as multi-layer perceptron (MLP) and bag of words.

[2]For advertising text generation, x is first ordered by attributes so that general attributes are ahead of specific ones; for recipe text generation, we retain the order in the dataset

4.3 Planning Process

The planning process generates a subset of input items to be covered for each sentence, thus decomposing long text generation into easier dependent sentence generation sub-tasks. Due to the flexibility of language, there may exist more than one reasonable text that covers the same input but in different order. To capture such variety, we model the diversity of reasonable planning with a global planning latent variable z^p. Different samples of z^p may lead to different planning results which control the order of content. This process can be formulated as follows:

$$g = argmax_g P(g|x, z^p) \quad (1)$$

where $g = g_1 g_2 ... g_T$ is a sequence of groups, and each group g_t is a subset of input items which is a main condition when realizing the sentence s_t.

The global latent variable z^p is assumed to follow the isotropic Gaussian distribution, and is sampled from its prior distribution $p_\theta(z^p|x) = \mathcal{N}(\mu^p, \sigma^{p2}\mathbf{I})$ during inference and from its approximate posterior distribution $q_{\theta'}(z^p|x, y) = \mathcal{N}(\mu^{p'}, \sigma^{p'2}\mathbf{I})$ during training:

$$[\mu^p; \log \sigma^{p2}] = \boldsymbol{MLP}_\theta(x) \quad (2)$$

$$[\mu^{p'}; \log \sigma^{p'2}] = \boldsymbol{MLP}_{\theta'}(x, y) \quad (3)$$

We solve Eq. 1 greedily by computing $g_t = argmax_{g_t} P(g_t|g_{<t}, x, z^p)$ with the plan decoder (a GRU). Specifically, at time step t, the plan decoder makes a binary prediction for each input item by estimating $P(d_i \in g_t|g_{<t}, x, z^p)$:

$$P(d_i \in g_t) = \boldsymbol{\sigma}(v_p^T \boldsymbol{tanh}(W_p[h_i; h_t^p] + b_p)) \quad (4)$$

where $\boldsymbol{\sigma}$ denotes the sigmoid function, h_i is the vector of input item d_i, and h_t^p is the hidden state of the plan decoder. Each group is therefore formed as $g_t = \{d_i|P(d_i \in g_t) > 0.5\}$ (If this is empty, we set g_t as $\{argmax_{d_i} P(d_i \in g_t)\}$.).

We feed $bow(g_t)$ (the average pooling of $\{h_i|d_i \in g_t\}$) to the plan decoder at the next time step, so that h_{t+1}^p is aware of what data has been selected and what has not. The planning process proceeds until the probability of stopping at the next time step is over 0.5:

$$P_t^{stop} = \boldsymbol{\sigma}(W_c h_t^p + b_c) \quad (5)$$

The hidden state is initialized with $\boldsymbol{enc}(x)$ and z^p. The plan decoder is trained with full supervision, which is applicable to those tasks where

reference plans are available or can be approximated. For both tasks we evaluate in this paper, we approximate the reference plans by recognizing the subset of input items covered by each sentence with string match heuristics. The loss function at time step t is given by:

$$\begin{aligned} -\log P(g_t = \widetilde{g}_t | \widetilde{g_{<t}}, x, z^p) \\ = -\sum_{d_i \in \widetilde{g}_t} \log P(d_i \in g_t) \\ -\sum_{d_i \notin \widetilde{g}_t} \log(1 - P(d_i \in g_t)) \end{aligned} \quad (6)$$

where $\widetilde{g}_t$ is the reference group. As a result, z^p is forced to capture features of reasonable planning.

4.4 Hierarchical Generation Process

The generation process produces a long text $y = s_1 s_2 ... s_T$ in alignment with the planning result $g = g_1 g_2 ... g_T$, which is formulated as follows:

$$c = \{x, z^p\} \quad (7)$$

$$y = argmax_y P(y|g, c) \quad (8)$$

We perform sentence-by-sentence generation and solve Eq. 8 greedily by computing $s_t = argmax_{s_t} P(s_t|s_{<t}, g, c)$. s_t focuses more on g_t than on the entire plan g. The generation process is conducted hierarchically, which consists of sentence-level generation and word-level generation. Sentence-level generation models inter-sentence dependencies at high level, and interactively controls word-level generation which conducts low-level sentence realization.

Sentence-level Generation The sentence decoder (a GRU) performs sentence-level generation; for each sentence s_t to be generated, it produces a sentence representation h^s_t and introduces a local latent variable z^s_t to control sentence realization.

The latent variable z^s_t is assumed to follow the isotropic Gaussian distribution. At time step t, the sentence decoder samples z^s_t from the prior distribution $p_\phi(z^s_t|s_{<t}, g, c) = \mathcal{N}(\mu^s_t, \sigma^{s2}_t \mathbf{I})$ during inference and from the approximate posterior distribution $q_{\phi'}(z^s_t|s_{\leq t}, g, c) = \mathcal{N}(\mu^{s'}_t, \sigma^{s'2}_t \mathbf{I})$ during training. h^s_t and the distribution of z^s_t are given by:

$$h^s_t = \boldsymbol{GRU}_s([z^s_{t-1}; h^w_{t-1}], h^s_{t-1}) \quad (9)$$

$$[\mu^s_t; \log \sigma^{s2}_t] = \boldsymbol{MLP}_\phi(h^s_t, bow(g_t)) \quad (10)$$

$$[\mu^{s'}_t; \log \sigma^{s'2}_t] = \boldsymbol{MLP}_{\phi'}(h^s_t, bow(g_t), s_t) \quad (11)$$

where h^w_{t-1} is the last hidden state of the word decoder after decoding sentence s_{t-1}, and $\boldsymbol{GRU}_s$ denotes the GRU unit of the sentence decoder. By this means, we constrain the distribution of z^s_t in two aspects. First, to strengthen the connection from the planning result g, we additionally condition z^s_t on g_t to keep z^s_t focused on g_t. Second, to capture the dependencies on $s_{<t}$, we explicitly model the dependencies among local latent variables by inputting z^s_{t-1} to the sentence decoder, so that z^s_t is conditioned on $z^s_{<t}$ and is expected to model smooth transitions in a long text.

We initialize the hidden state h^s_0 by encoding the input x, the global planning latent variable z^p and the planning result g:

$$h^g_t = \boldsymbol{GRU}_g(bow(g_t), h^g_{t-1}) \quad (12)$$

$$h^s_0 = W_s[\boldsymbol{enc}(x); z^p; h^g_T] + b_s \quad (13)$$

where h^g_T is the last hidden state of $\boldsymbol{GRU}_g$ that encodes the planning result g.

Word-level Generation The word decoder (a GRU) conducts word-level generation; it decodes a sentence $s_t = argmax_{s_t} P(s_t|s_{<t}, z^s_t, g, c)$ conditioned on $\{h^s_t, z^s_t, g_t\}$. Specifically, we sample word w^t_k of s_t as follows:

$$w^t_k \sim P(w^t_k | w^t_{<k}, s_{<t}, z^s_t, g, c) \quad (14)$$

4.5 Loss Function

We train our model end-to-end. The loss function has three terms: the negative evidence lower bound (ELBO) of $\log P(y|x)$ ($\mathcal{L}_1$), the loss of predicting the stop signal ($\mathcal{L}_2$) and the bag-of-word loss (Zhao et al., 2017) ($\mathcal{L}_3$).

We first derive the ELBO:

$$\begin{aligned} \log P(y|x) \geq E_{q_{\theta'}(z^p|x,y)}[\log P(y|x, z^p)] \\ - D_{KL}(q_{\theta'}(z^p|x, y)||p_\theta(z^p|x)) \end{aligned} \quad (15)$$

$$\begin{aligned} \log P(y|x, z^p) = \log P(g, y|x, z^p) \\ = \sum_{t=1}^{T} \log P(g_t|g_{<t}, x, z^p) \\ + \log P(s_t|s_{<t}, g, x, z^p) \end{aligned} \quad (16)$$

$$\begin{aligned} \log P(s_t|s_{<t}, g, x, z^p) \\ \geq E_{q_{\phi'}(z^s_t|s_{\leq t}, g, x, z^p)}[\log P(s_t|s_{<t}, z^s_t, g, x, z^p)] \\ - D_{KL}(q_{\phi'}(z^s_t|s_{\leq t}, g, x, z^p)||p_\phi(z^s_t|s_{<t}, g, x, z^p)) \end{aligned} \quad (17)$$

We can obtain the ELBO by unfolding the right hand side of Eq. 15 with Eq. 16 and 17. During

training, we use linear KL annealing technique to alleviate the KL collapse problem(Bowman et al., 2016).

$\mathcal{L}_2$ is given by:

$$\mathcal{L}_2 = \sum_{t=1}^{T-1} \log P_t^{stop} + \log(1 - P_T^{stop}) \quad (18)$$

$\mathcal{L}_3$ is the sum of bag-of-word loss (Zhao et al., 2017) applied to each sentence, which is another technique to tackle the KL collapse problem.

5 Experiments

5.1 Dataset

We evaluated PHVM on two generation tasks. The first task is the new advertising text generation task which is to generate a long advertising text that covers all given attribute-value pairs for a piece of clothing. The second task is the recipe generation task from (Kiddon et al., 2016) which is to generate a correct recipe for the given recipe title and ingredient list.

Advertising Text Generation We constructed our dataset from a Chinese e-commerce platform. The dataset consists of 119K pairs of advertising text and clothing specification table. Each table is a set of attribute-value pairs describing a piece of clothing. We made some modifications to the original specification tables. Specifically, if some attribute-value pairs from a table do not occur in the corresponding text, the pairs are removed from the table. We also recognized attribute values by string matching with a dictionary of attribute values. If a pair occurs in the text but not in the table, the pair is added to the table.

The statistics are shown in Table 1 and Table 2.

Category	Tops	Dress / Skirt	Pants
# Type	22	23	9
# Attr.	13	16	11
# Val.	264	284	203
Avg. # Input Pairs	7.7	7.7	6.6
Avg. Len.	110	111	108
# Instances	48K	47K	24K

Table 1: Detailed statistics of our dataset. *# Attr.* / *# Val.*: the total number of attributes / attribute values.

Our dataset consists of three categories of clothing: tops, dress / skirt, and pants, which are further divided into 22, 23, and 9 types respectively (E.g., shirt, sweater are two types of tops). Other categories (e.g., hats and socks) are discarded because these categories have insufficient data for training. The average length of advertising text is about 110 words. To evaluate the expression diversity of our dataset, we computed distinct-4 (see Section 5.3) on 3,000 randomly sampled texts from our dataset. The distinct-4 score is 85.35%, much higher than those of WIKIBIO (Lebret et al., 2016) and ROTOWIRE (Wiseman et al., 2017) (two popular data-to-text datasets). Therefore, our dataset is suitable for evaluating long and diverse text generation[3].

# Attr.	# Val.	Vocab	Avg. # Input Pairs	Avg. # Len.
28	633	54.9K	7.5	110.2

Table 2: General statistics of our dataset. We counted the size of vocabulary after removing brand names.

We left 1,070 / 3,127 instances for validation / test, and used the remainder for training.

Recipe Text Generation We used the same train-validation-test split (82,590 / 1,000 / 1,000) and pre-processing from (Kiddon et al., 2016). In the training set, the average recipe length is 102 tokens, and the vocabulary size of recipe title / text is 3,793 / 14,103 respectively. The recipe dataset covers a wide variety of recipe types indicated by the vocabulary size of recipe title.

5.2 Baselines

We compared our model with four strong baselines where the former two do not perform planning and the latter two do:

Checklist: This model utilizes an attention-based checklist mechanism to record what input data has been mentioned, and focuses more on what has not during generation (Kiddon et al., 2016).

CVAE: The CVAE model proposed by Zhao et al. (2017) uses a latent variable to capture the diversity of responses in dialogue generation. We adapted it to our task by replacing the hierarchical encoder with a one-layer bidirectional GRU.

Pointer-S2S: A two-stage method (Puduppully et al., 2019) that decides the order of input data with Pointer Network (Vinyals et al., 2015) before generation with Sequence-to-Sequence (Seq2Seq) (Bahdanau et al., 2015).

Link-S2S Link-S2S (Sha et al., 2018) is a Seq2Seq with link-based attention mechanism

[3]We presented a detailed comparison with other benchmark corpora in Appendix A.2.

Models	BLEU (%)	Coverage (%)	Length	Distinct-4 (%)	Repetition-4 (%)
Checklist	**4.17**	84.52**	83.61**	21.95**	46.40**
CVAE	4.02	77.65**	80.96**	41.69**	36.58**
Pointer-S2S	3.88**	85.97**	74.88**	18.16**	36.78**
Link-S2S	3.90**	70.49**	**95.65**	16.64**	59.83**
PHVM (ours)	2.85**	**87.05**	89.20**	**72.87**	**3.90**
w/o z^p	3.07**	84.74**	91.97**	70.51**	4.19
w/o z_t^s	3.38**	84.89**	75.28**	42.32**	20.88**

Table 3: Automatic evaluation for advertising text generation. We applied bootstrap resampling (Koehn, 2004) for significance test. Scores that are significantly worse than the best results (in bold) are marked with ** for p-value < 0.01.

where a link matrix parameterizes the probability of describing one type of input item after another.

5.3 Implementation Details

For both advertising text generation and recipe text generation, the settings of our model have many in common. The dimension of word embedding is 300. All embeddings were randomly initialized. We utilized GRU for all RNNs. All RNNs, except the input encoder, the plan decoder, and the plan encoder, have a hidden size of 300. The global planning latent variable and local latent variables have 200 dimensions. We set batch size to 32 and trained our model using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 and gradient clipping threshold at 5. We selected the best model in terms of $\mathcal{L}_1 + \mathcal{L}_2$ on the validation set.

As we need to train the plan decoder with full supervision, we extracted plans from the texts by recognizing attribute values (or ingredients) in each sentence with string match heuristics. Some sentences do not mention any input items; we associated these sentences with a special tag, which is treated as a special input item for Pointer-S2S, Link-S2S, and our model. Although our extraction method can introduce errors, the extracted plans are sufficient to train a good plan decoder[4].

Advertising Text Generation We embedded an attribute-value pair by concatenating the embedding of attribute and the embedding of attribute value. Embedding dimensions for attribute and attribute value are 30 and 100 respectively. The input encoder, the plan decoder, and the plan encoder all have a hidden size of 100.

Recipe Text Generation We embedded a multi-word title (ingredient) by taking average pooling of the embeddings of its constituent words. Embedding dimensions for title word and ingredient word are 100 and 200 respectively. The input encoder, the plan decoder, and the plan encoder all have a hidden size of 200.

[4]Our corpus and code are available at /https://github.com/ZhihongShao/Planning-based-Hierarchical-Variational-Model.

5.4 Automatic Evaluation Metrics

We adopted the following automatic metrics. (1) **Corpus BLEU**: BLEU-4 (Papineni et al., 2002). (2) **Coverage**: This metric measures the average proportion of input items that are covered by a generated text. We recognized attribute values (ingredients) with string match heuristics. For the advertising text generation task, synonyms were also considered. (3) **Length**: The average length of the generated texts. (4) **Distinct-4**: Distinct-n (Li et al., 2016) is a common metric for diversity which measures the ratio of distinct n-grams in generated tokens. We adopted distinct-4. (5) **Repetition-4**: This metric measures redundancy with the percentage of generated texts that repeat at least one 4-gram.

5.5 Advertising Text Generation

5.5.1 Automatic Evaluation

Table 3 shows the experimental results. As our dataset possesses high expression diversity, there are many potential expressions for the same content, which leads to the low BLEU scores of all models. Our model outperforms the baselines in terms of coverage, indicating that it learns to arrange more input items in a long text. With content ordering, Pointer-S2S outperforms both Checklist and CVAE in coverage. By contrast, our planning mechanism is even more effective in controlling generation: each sentence generation subtask is specific and focused, and manages to cover 95.16% of the corresponding group on average. Noticeably, Link-S2S also models planning but

Models	Grammaticality			κ	Coherence			κ
	Win (%)	Lose (%)	Tie (%)		Win (%)	Lose (%)	Tie (%)	
PHVM vs. Checklist	59.0**	23.5	17.5	0.484	54.5*	42.5	3.0	0.425
PHVM vs. CVAE	69.5**	13.5	17.0	0.534	60.0**	37.0	3.0	0.426
PHVM vs. Pointer-S2S	76.5**	17.0	6.5	0.544	56.5**	39.0	4.5	0.414
PHVM vs. Link-S2S	66.0**	28.5	5.5	0.462	62.5**	31.5	6.0	0.415

Table 4: Manual pair-wise evaluation for advertising text generation. We conducted Sign Test for significance test. Scores marked with * mean p-value < 0.05 and ** for p-value < 0.01. κ denotes Fleiss' kappa, all indicating *moderate agreement*.

has the lowest coverage, possibly because a static link matrix is unable to model flexible content arrangement in long text generation. As for diversity, our model has substantially lower repetition-4 and higher distinct-4, indicating that our generated texts are much less redundant and more diversified. Notably, Link-S2S has the longest texts but with the highest repetition-4, which produces many redundant expressions.

To investigate the influence of each component in the hierarchical latent structure, we conducted ablation tests which removed either global latent variable z^p or local latent variables z_t^s. As observed, removing z^p leads to significantly lower distinct-4, indicating that z^p contributes to expression diversity. The lower coverage is because the percentage of input items covered by a planning result decreases from 98.4% to 94.4% on average, which indicates that z^p encodes useful information for planning completeness. When removing z_t^s, distinct-4 drops substantially, as the model tends to generate shorter and more common sentences. This indicates that z_t^s contributes more to capturing variations of texts. The significantly higher repetition-4 is because removing z_t^s weakens the dependencies among sentences so that the word decoder is less aware of the preceding generated context. The lower coverage is because each generated sentence covers less planned items (from 95.16% to 93.07% on average), indicating that z_t^s keeps sentence s_t more focused on its group.

5.5.2 Manual Evaluation

To better evaluate the quality of the generated texts, we conducted pair-wise comparisons manually. Each model generates texts for 200 randomly sampled inputs from the test set. We hired five annotators to give preference (win, lose or tie) to each pair of texts (ours vs. a baseline, 800 pairs in total).

Metrics Two metrics were independently evaluated during annotation: **grammaticality** which measures whether a text is fluent and grammatical, and **coherence** which measures whether a text is closely relevant to input, logically coherent, and well-organized.

Results The annotation results in Table 4 show that our model significantly outperforms baselines in both metrics. Our model produces more logically coherent and well-organized texts, which indicates the effectiveness of the planning mechanism. It is also worth noting that our model performs better in terms of grammaticality. The reason is that long text generation is decomposed into sentence generation sub-tasks which are easier to control, and our model captures inter-sentence dependencies through modeling the dependencies among local latent variables.

5.5.3 Diversity of Planning

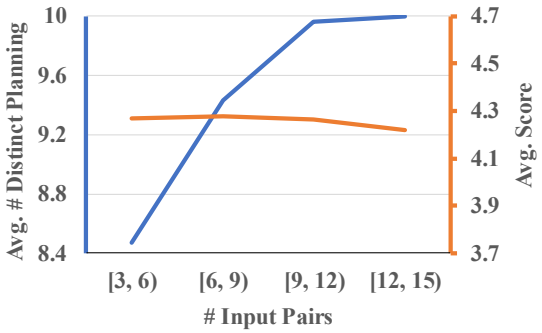

Figure 3: Average number of distinct planning results (left) / average score of generation quality (right) when the number of input pairs varies.

To evaluate how well our model can capture the diversity of planning, we conducted another manual evaluation. We randomly sampled 100 test inputs and generated 10 texts for each input by repeatedly sampling latent variables. Five annotators were hired to score (a Likert scale $\in [1, 5]$)

Models	BLEU (%)	Coverage (%)	Length	Distinct-4 (%)	Repetition-4 (%)
Checklist §	3.0	67.9	N/A	N/A	N/A
Checklist	2.6**	66.9*	67.59	30.67**	39.1**
CVAE	4.6	63.0**	57.49**	52.53**	38.7**
Pointer-S2S	4.3	70.4**	59.18**	30.72**	36.4**
Link-S2S	1.9**	53.8**	40.34**	24.93**	31.6**
PHVM (ours)	**4.6**	**73.2**	**70.92**	**67.86**	**17.3**

Table 5: Automatic evaluation for recipe text generation. Checklist was trained with its own source code. We also re-printed results from (Kiddon et al., 2016) (i.e., Checklist §). We applied bootstrap resampling (Koehn, 2004) for significance test. Scores that are significantly worse than the best results (in bold) are marked with * for p-value $<$ 0.05 or ** for p-value $<$ 0.01.

a text about whether it is a qualified advertising text, which requires comprehensive assessment in terms of fluency, redundancy, content organization, and coherence. We computed the average of five ratings as the final score of a generated text.

Results The average score of a generated text is 4.27. Among the 1,000 generated texts, 79.0% of texts have scores above 4. These results demonstrate that our model is able to generate multiple high-quality advertising texts for the same input.

We further analyzed how our model performs with different numbers of input attribute-value pairs (see Figure 3). A larger number of input items indicates more potential reasonable ways of content arrangement. As the number of input items increases, our model produces more distinct planning results while still obtaining high scores (above 4.2). It indicates that our model captures the diversity of reasonable planning. The average score drops slightly when the number of input pairs is more than 12. This is due to insufficient training data for this range of input length (accounting for 6.5% of the entire training set).

To further verify the planning diversity, we also computed self-BLEU (Zhu et al., 2018) to evaluate how different planning results (or texts) for the same input overlap (by taking one planning result (or text) as hypothesis and the rest 9 for the same input as reference and then computing BLEU-4). The average self-BLEU of the planning results is 43.37% and that of the texts is 16.87%, which demonstrates the much difference among the 10 results for the same input.

Annotation Statistics The Fleiss' kappa is 0.483, indicating *moderate agreement*.

5.6 Recipe Text Generation

Table 5 shows the experimental results. Our model outperforms baselines in terms of coverage and diversity; it manages to use more given ingredients and generates more diversified cooking steps. We also found that Checklist / Link-S2S produces the general phrase "all ingredients" in 14.9% / 24.5% of the generated recipes, while CVAE / Pointer-S2S / PHVM produce the phrase in 7.8% / 6.3% / 5.0% of recipes respectively. These results demonstrate that our model may generalize well to other data-to-text generation tasks.

6 Case Study

We present examples for both tasks in Appendix B.

7 Conclusion and Future Work

We present the Planning-based Hierarchical Variational Model (PHVM) for long and diverse text generation. A novel planning mechanism is proposed to better model input data and address the inter-sentence incoherence problem. PHVM also leverages a hierarchical latent structure to capture the diversity of reasonable planning and sentence realization. Experiments on two data-to-text corpora show that our model is more competitive to generate long and diverse texts than state-of-the-art baselines.

Our planning-based model may be inspiring to other long text generation tasks such as long text machine translation and story generation.

Acknowledgements

This work was supported by the National Science Foundation of China (Grant No. 61936010/61876096) and the National Key R&D Program of China (Grant No. 2018YFC0830200). We would like to thank THUNUS NExT Joint-Lab for the support.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

John A. Bateman. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(1):15–55.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Towards knowledge-based personalized product description generation in e-commerce. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.*, pages 3040–3050.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Hercules Dalianis and Eduard H. Hovy. 1993. Aggregation in natural language generation. In *Trends in Natural Language Generation, An Artificial Intelligence Perspective, Fourth European Workshop, EWNLG '93, Pisa, Italy, April 28-30, 1993, Selected Papers*, pages 88–105.

Kees van Deemter, Mariët Theune, and Emiel Krahmer. 2005. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24.

Pablo A Duboue and Kathleen R McKeown. 2003. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 121–128. Association for Computational Linguistics.

Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 183–191.

Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. 2018. Topic-to-essay generation with neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 4078–4084.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In (McIlraith and Weinberger, 2018), pages 5141–5148.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artif. Intell.*, 63(1-2):341–385.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In (Su et al., 2016), pages 329–339.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. ACL.

Ioannis Konstas and Mirella Lapata. 2013. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1503–1514. ACL.

Karen Kukich. 1983. Design of a knowledge-based report generator. In *21st Annual Meeting of the Association for Computational Linguistics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 15-17, 1983.*, pages 145–150.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In (Su et al., 2016), pages 1203–1213.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the*

North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, pages 110–119.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1106–1115.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In (McIlraith and Weinberger, 2018), pages 4881–4888.

Sheila A. McIlraith and Kilian Q. Weinberger, editors. 2018. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press.

Susan W McRoy, Songsak Channarukul, and Syed S Ali. 2003. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381–420.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 720–730.

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*.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M. Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1539–1550.

Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors. 2017. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1792–1801.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

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 *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3295–3301. AAAI Press.

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujian Li, Baobao Chang, and Zhifang Sui. 2018. Order-planning neural text generation from structured data. In (McIlraith and Weinberger, 2018), pages 5414–5421.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In (Palmer et al., 2017), pages 2210–2219.

Dinghan Shen, Asli Çelikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, Jianfeng Gao, and Lawrence Carin. 2019. Towards generating long and coherent text with multi-level latent variable models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2079–2089.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *NIPS*, pages 3483–3491.

Amanda Stent, Rashmi Prasad, and Marilyn A. Walker. 2004. Trainable sentence planning for complex information presentations in spoken dialog systems. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain.*, pages 79–86.

Jian Su, Xavier Carreras, and Kevin Duh, editors. 2016. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2692–2700.

Sam Wiseman, Stuart M. Shieber, and Alexander M. Rush. 2017. Challenges in data-to-document generation. In (Palmer et al., 2017), pages 2253–2263.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018. Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3940–3949. Association for Computational Linguistics.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 654–664.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100.