Supplementary Material

Xinyue Zhang[1]*, Haolong Li[1]*, Jiawei Ma[1], Chen Ye[1,2†]
[1]Department of Computer Science and Technology, Tongji University, Shanghai, China
[2]The Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University
{elainexinyuezhang, Furlongli322, jiavve1.10}@gmail.com, yechen@tongji.edu.cn

1. settings

1.1. Detailed Architectures

1.1.1. Models of the Stage I

Inspired by VQVAE[6], we build our encoder and decoder of stage I. Following VAEs[4, 5], our model consists of an encoder parameterizing a posterior distribution of discrete variables and a decoder with a prior distribution over output data. Notably, both the posterior and prior distributions are categorical, necessitating the sampling of categorical variables to index our codebook with $K = 30000$ categories with embedding $dim = 16$. To preserve the structural integrity of our data (formatted as 64×6 matrices post-stroke modeling) and mitigate the risk of over-convolution along specific dimensions within Convolutional Neural Networks (CNNs), our preprocessing strategy involves reshaping the input data into $6 \times 8 \times 8$ matrices prior to feeding them into the encoder module. With our encoder, one stroke after stroke modeling can be compressed to 8 stroke embeddings, facilitating efficient data representation and compression. The whole structure for stage I model is shown in Tabs. 1 and 2.

1.1.2. Models of the Stage II

In the Stage II, we undertake the fine-tuning of the DeepSeek-Coder-1.3B model [2] to generate next stroke embeddings. We set the batch size per device equal to 4, while concurrently introducing a label smoothing factor of 1×10^{-3}. Given the intrinsic characteristics of our dataset, where the stroke count for each character does not exceed 34, we ascertain that every character can be accurately encapsulated within a space delineated by 272 stroke embeddings. We establish the maximum allowable sum of strokes per individual sample as $N_c = 100$. Furthermore, considering the diversity and complexity of textual content, we set the maximum text length parameter to $N_l = 820$. This configuration enables the model to effectively capture the rich semantic nuances embedded within a comprehensive textual context.

1.2. Hyperparameters and Training Settings

Our collected glyph dataset imposes constraints on the number of instructions per stroke, limiting it to under 64 instructions, and restricting the stroke count per glyph to a maximum of 34 strokes, forming the basis for constructing our input samples.

In the Stage I, we configure our training parameters as follows: utilizing a batch size of 128, each stroke is associated with a maximum of 64 instructions, and the learning rate is set at 1×10^{-4}. In the Stage II, we conduct fine-tuning of the pre-trained DeepSeek-Coder-1.3B model for 5 epochs. The specific hyperparameters and training configurations pertaining to various stages of our model training are meticulously detailed in Tab. 3.

2. Dataset

Our dataset construction process is rooted in the extraction of textual content from 'Xinhua Dictionary' and 'The Complete Tang Poems', and the glyph content from our collected glyph dataset. These distinct datasets are gathered inde-

Layer	Activation	Skip Connect	Output Shape
Input	-	-	$6 \times 8 \times 8$
Conv	-	N	$512 \times 4 \times 4$
Conv	ReLU	N	$1024 \times 2 \times 4$
Conv	ReLU	N	$1024 \times 2 \times 4$
Conv Conv	ReLU ReLU	Y Y	$256 \times 2 \times 4$ $1024 \times 2 \times 4$
Conv Conv	ReLU ReLU	Y Y	$256 \times 2 \times 4$ $1024 \times 2 \times 4$
Linear	-	N	$16 \times 2 \times 4$
Reshape	-	N	$2 \times 4 \times 16$

Table 1. Encoder architecture of the Stage I.

*These authors contribute equally.
†Corresbonding author.

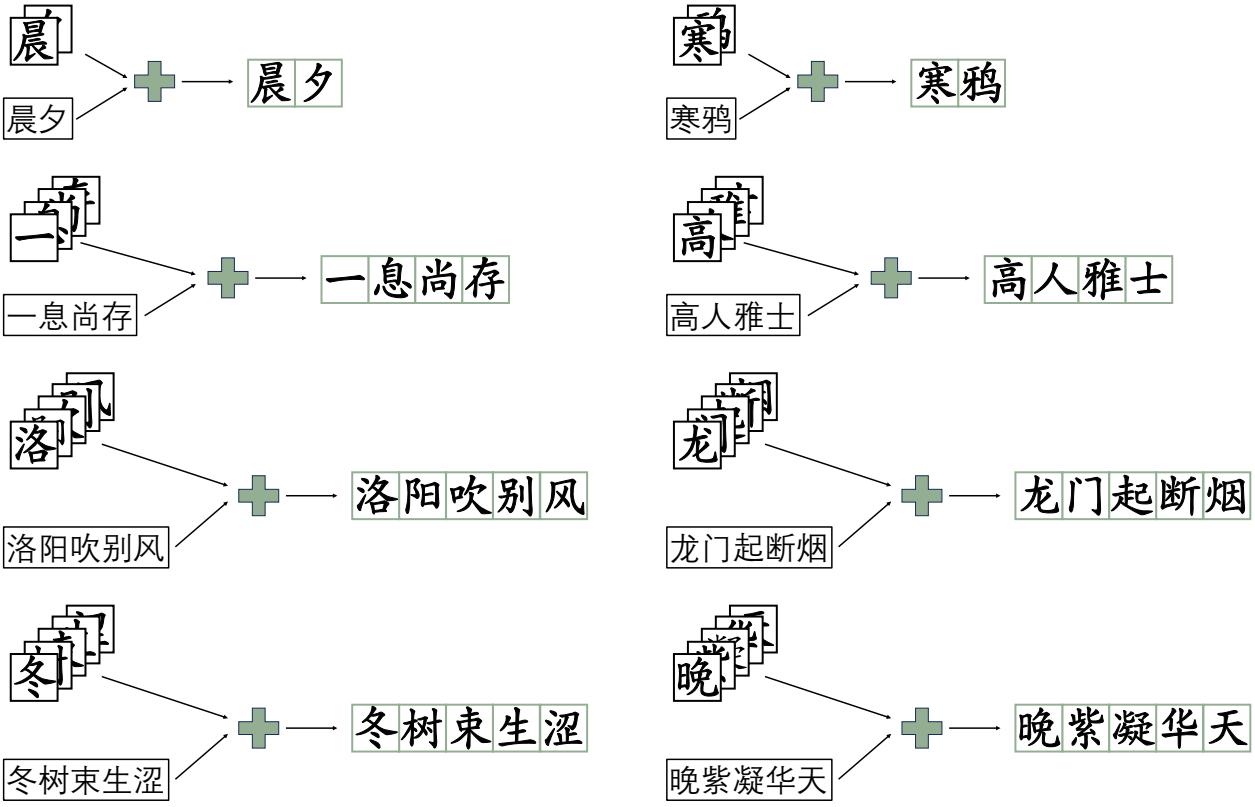

Figure 1. Examples of our dataset construction process.

Layer	Activation	Skip Connect	Output Shape
Input	-	-	$2 \times 4 \times 16$
ConvTrans	-	N	$1024 \times 2 \times 4$
ConvTrans	ReLU	Y	$256 \times 2 \times 4$
ConvTrans	ReLU	Y	$1024 \times 2 \times 4$
Conv	ReLU	Y	$256 \times 2 \times 4$
Conv	ReLU	Y	$1024 \times 2 \times 4$
Conv	ReLU	Y	$256 \times 2 \times 4$
Conv	ReLU	Y	$1024 \times 2 \times 4$
ConvTrans	ReLU	N	$512 \times 4 \times 4$
ConvTrans	ReLU	N	$6 \times 8 \times 8$

Table 2. Decoder architecture of the Stage I.

pendently. By leveraging the textual information encapsulated within words, idioms, and verses, we systematically query the glyph dataset to identify and extract the relevant data related to the text elements in focus. Subsequently, this retrieved information is amalgamated to form the composite training dataset. The detailed process of integrating this data is shown visually in Fig. 1, explaining the sequential steps involved in harmonizing textual and glyph data sources to construct a comprehensive training dataset.

3. Stroke Instruction and Calculation

The SVG instructions used are listed in Tab. 4.

Our scalable vector graphics (SVG)[1] format mainly uses linear, quadratic, and cubic Bézier curves to describe the glyph outline. We make transformations on linear and quadratic ones to unify them into representations of cubic Bézier curves. Equations of Bézier curves (Linear Bézier curves B_L, Quadratic Bézier curves B_Q, Cubic Bézier curves B_c) are listed in Eqs. (1) to (3).

$$B_L(t) = (1-t)P_0 + P_1,\ t \in [0,1] \tag{1}$$

$$B_Q(t) = (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2,\ t \in [0,1] \tag{2}$$

$$B_C(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3,\ t \in [0,1] \tag{3}$$

Each equation contains a parameter $t \in [0,1]$ to describe the curve, so we expand the equation according to the pa-

Config	Value
Dataset Settings	
Fixed Length of Strokes per Glyph	$N_g = 34$
Fixed Length of Instructions per Stroke	$N_s = 64$
Settings of the Stage I	
Code Book Category	$K = 30000$
Stroke Embedding Dim	$dim = 16$
Optimizer	Adam [3]
Batch Size	128
Learning Rate	$1e-5$
Training Epoch	2000
Settings of the Stage II	
Max Strokes Sum per Sample	$N_c = 100$
Max Text Length per Sample	$N_l = 820$
Optimizer	Adam [3]
Batch Size	4
Learning Rate	$5e-5$
Label Smoothing Factor	$1e-3$
Training Epoch	5
Settings of our Human Evaluation	
Factor of Identifiablity	$\alpha_{Ide} = 0.4$
Factor of Aesthetics	$\alpha_{Aes} = 0.3$
Factor of Literature Quality	$\alpha_{Lit} = 0.3$
Factor of Experts	$\beta_{exp} = 0.7$
Factor of Graduate Students	$\beta_{stu} = 0.3$

Table 3. Hyperparameter settings.

Order	Parameters	Meaning
M	P_0	Move to
L, H, V	P_1	Linear Line
Q, T	P_1, P_2	Quadratic Curve
C, S	P_1, P_2, P_3	Cubic Curve
Z	-	Close Curve

Table 4. Our SVG instructions. Each instruction is represented by (Order, Parameters) tuple in SVG form.

rameter t, and adjust the coefficients to transform Eqs. (1) and (2) into the expression of Eq. (3) in Eqs. (4) and (5).

$$\begin{aligned} B_L(t) =&(1-t)^3 P_0 + 3t(1-t)^2(P_0 + \frac{2}{3}(P_1 - P_0)) \\ &+ 3t^2(1-t)P_1 + t^3 P_1, t \in [0,1] \end{aligned} \tag{4}$$

$$\begin{aligned} B_Q(t) =&(1-t)^3 P_0 + 3t(1-t)^2(P_0 + \frac{2}{3}(P_1 - P_0)) \\ &+ 3t^2(1-t)(P_2 + \frac{2}{3}(P_1 - P_2)) + t^3 P_2, \\ &t \in [0,1] \end{aligned} \tag{5}$$

4. Results

4.1. Missing Stroke Generation

Given varying strokes and positions as input, our model produces diverse outcomes through the prediction of stroke embeddings, thereby composing strokes and characters following stroke sequences. The generation process is visually represented in Fig. 2. Notably, as the input strokes decrease in number, our model showcases the ability to accurately predict the missing strokes essential for completing the entire glyph, as shown in Fig. 4.

4.2. Multiple Chinese Characters Generation

When presented with a variety of distinct glyphs as input, our model demonstrates its competence in producing a diverse range of words, idioms, and verses, each infused with aesthetic depth and significant cultural references, as shown in Figs. 3, 5 and 6.

The outcomes generated not only showcase creativity but also function as valuable resources for linguistic analysis and artistic interpretation. Significantly, the materials generated by our model, as shown in Figs. 5 and 6, represent novel compositions previously absent from the dataset, underscoring the model's ability to generate innovative and imaginative content.

References

[1] J David Eisenberg and Amelia Bellamy-Royds. *SVG essentials: Producing scalable vector graphics with XML.* " O'Reilly Media, Inc.", 2014. 2

[2] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. Deepseek-coder: When the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024. 1

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 3

[4] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. 1

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. 1

[6] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 1

Figure 2. Generating glyphs stroke by stroke. Gray strokes are inputs given to our model and black strokes are generated in stroke orders with high quality. Each line from left to right shows the generating order.

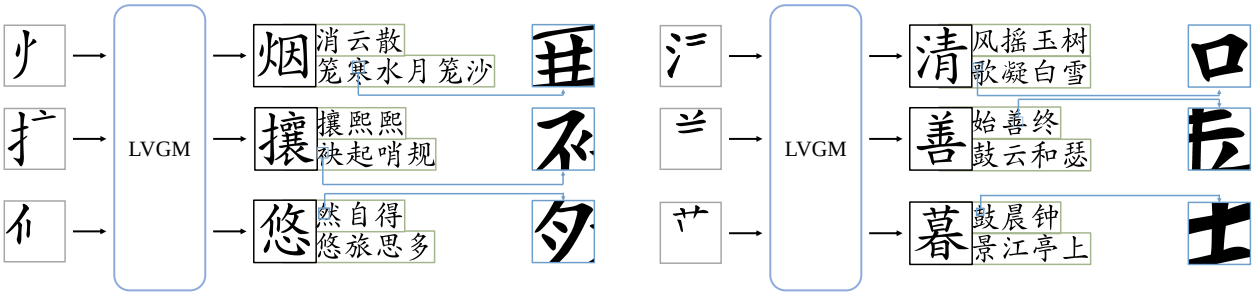

Figure 3. Examples of our generation results.

Figure 4. Generating glyphs by decreasing input strokes. For each triplet, the left part is input data, and the middle one is the predicted strokes. We present a mixed complete glyph on the right part.

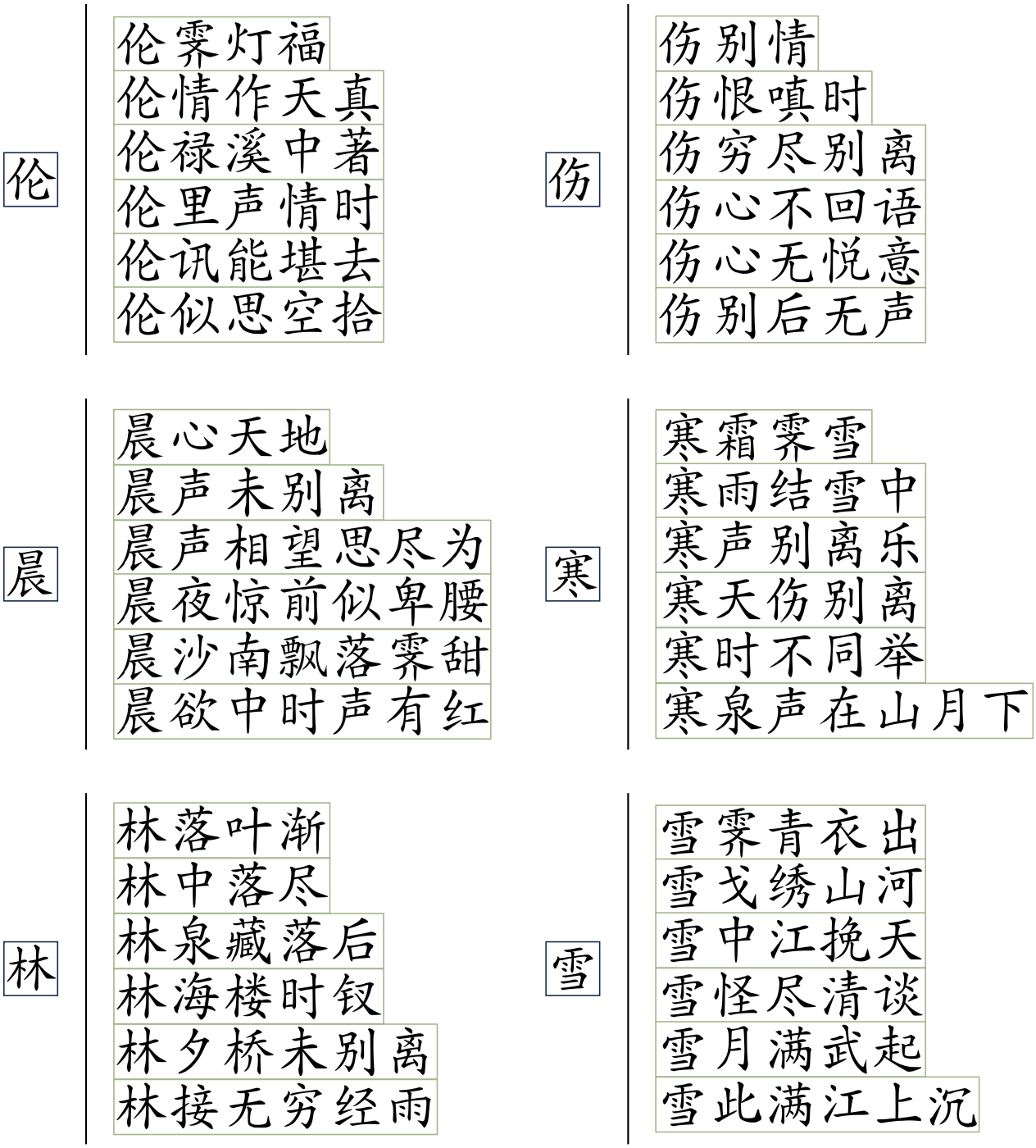

Figure 5. Examples of our generation results. These results demonstrate our model can generate unprecedented meaningful textual content.

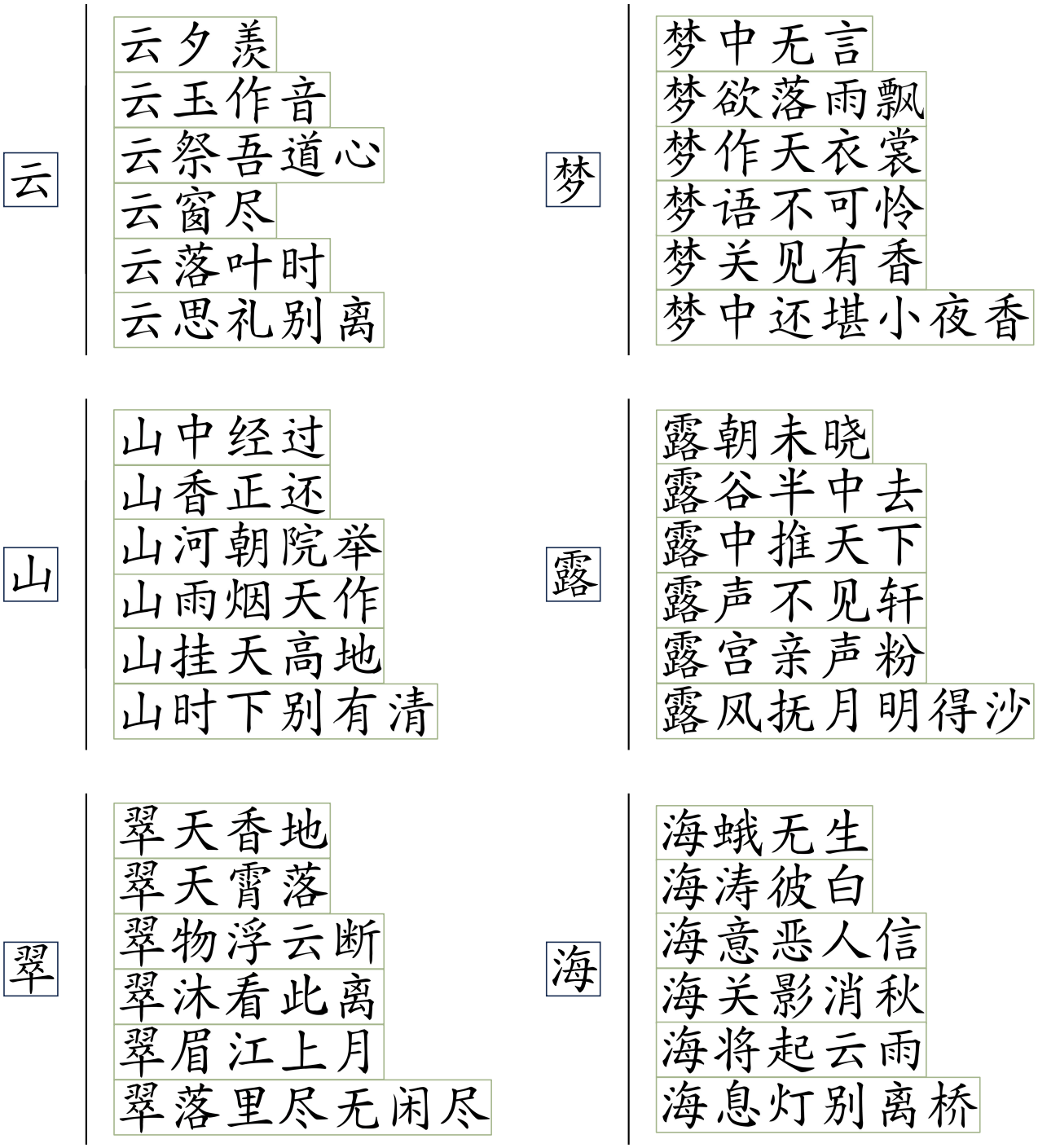

Figure 6. Examples of our generation results. These results demonstrate our model can generate unprecedented meaningful textual content.