Research on Story Text Generation Based on Transformer Model

Shiyu Shao

Shanghai Maritime University, Shanghai, China
872824410@qq.com

Abstract. The transformer model was used to train and generate story text this time because certain parts or endings of the original story were not satisfactory. This study tried to use the model training to obtain other story paths. The main purpose is to study two paths: one is how to use pre-trained models for fine-tuning to achieve the desired effect, and the other is how to build a model trained from scratch to achieve the desired effect. DeepSeek R1 will be used as a control group to evaluate the generation effect.According to the results, the pre-trained model performs better on smaller datasets, generating logical sentences and paragraphs, while the model trained from scratch has not yet achieved good results on smaller datasets. As an improvement measure, a larger dataset will be used to enhance the model's generation performance, while adjusting new hyperparameters to fit the dataset.

Keywords: Transformer, GPT-2, Story Text Generation, small datasets, machine learning

1. Introduction

In recent years, the use of Large Language Models (LLMs) and Prompt Learning has become a new paradigm for generative tasks [1]. The Transformer architecture, first proposed by Vaswani et al. [2], has revolutionized natural language processing and serves as the foundation for most modern text generation systems. By consulting relevant materials, one can easily learn how to build and fine-tune pre-trained models as well as models trained from scratch. But, the small sample size limitation greatly hinders the further development and practical application of many text generation tasks [3]. However, there is currently a lack of data on how much data can achieve a good training effect. This study mainly investigates how to generate a text with a similar style to the original text using a transformer model on a small dataset of approximately 20,000 English words, and adjust hyperparameters and other parts to achieve perfect results.

Approaches like BERT [4] and GPT [5] have demonstrated remarkable capabilities in language understanding and generation tasks, making them excellent candidates for story generation. In the research, Python modules such as PyTorch and the Transformer framework are mainly used to achieve the research objectives. This study aims to successfully validate the text learning and text generation abilities of pre-trained models and models trained from scratch on smaller datasets, expanding on previous work in long-form story generation [6]. This study aims to successfully validate the text learning and text generation abilities of pre-trained models and models trained from scratch on smaller datasets.

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

2. Methodology

This research mainly constructed three different transformer models. The first was a generative model fine-tuned by GPT-2, trained on a dataset of approximately 20,000 words of English novels. The second was a transformer model trained from scratch on a dataset of approximately 120,000 words of English novels. The third was a text generation control group using the DeepSeek-R1 API.

2.1. Fine-tuned GPT-2

This is the GPT-2 model diagram created in the paper, showing the process of training with raw data. The model procedure starts from raw data, first tokenizing the data into basic units (such as words or subwords), and then doing chunking to divide contextual fragments for model processing. After the data is changed into dense vectors by the embedding layer, it is entered into a stacked Transformer module for feature extraction and context modeling. This module collects sequence dependencies through a multi-layer self-attention mechanism. After forward propagation of the model, the difference between the expected results and the true labels is analyzed by loss calculation, and based on this, backpropagation is done. The optimizer is used to dynamically change the model parameters to minimize the loss. It is worth mentioning that the specific design of the loss function and output layer is not clearly defined in the figure, and the position of the model parameters implies its relationship with the process after partitioning, which may need to be further refined in conjunction with specific implementation. The entire process covers the core steps from data preprocessing to parameter optimization, reflecting the basic framework of end-to-end training which is shown in Figure 1 for transformer models.

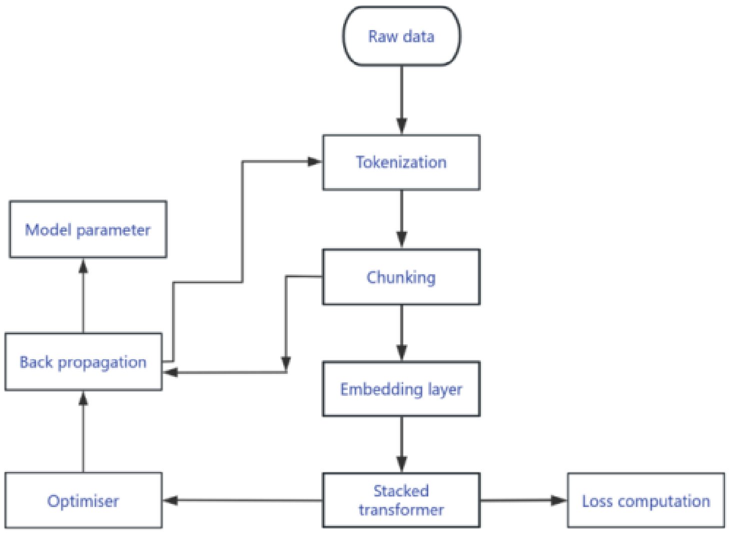


Figure 1: The structure of model

2.1.1. Input layer and transformer decoder

As illustrated by the Figure 2, the research model input processing part of that, in which the input is embedding, and the image of each token in the text; this is the process by which the computer is able to "understanding" the semantic information of words in numerical form, which is converted by the input of the text into a high-dimensional vector by which each of the tokens is first converted into the vector of a large-dimensional vector and the semantic information of those words is realized by the computer in numerical form. GPT-2 chooses to directly learn positional representations by neural networks in a way that is flexible enough to adapt to complicated language patterns; in this way, the model additionally includes a feature set that is made up of a12-layer Transformer decoder stack and is used to capture the 12 tier Core Processing layer that includes the processed vectors, each of which has two main parts of the input layer called the encoder attention mechanism for global contextual relations, which is used for feature deepening and feed forward neural networks for feature learning in each layer. The GPT-2 model diagram that was created in the study is used to train the raw data.

This architectural design follows the principles established by Radford et al. [5] in their work on GPT models as unsupervised multitask learners, which has proven particularly effective for creative text generation tasks.

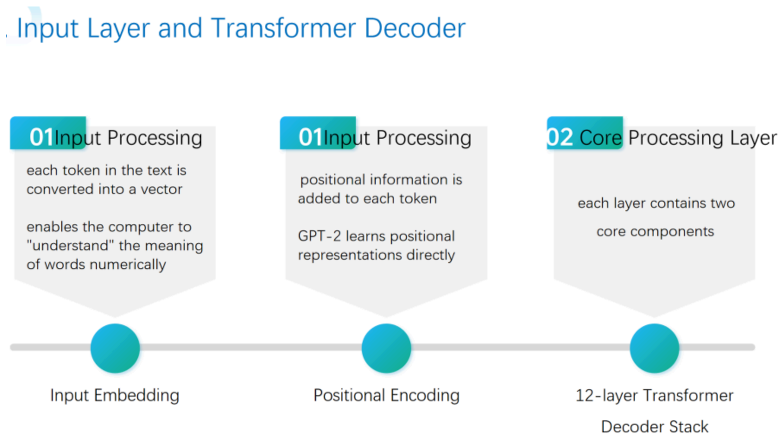

Input Layer and Transformer Decoder
01 Input Processing
each token in the text is converted into a vector
enables the computer to "understand" the meaning of words numerically
01 Input Processing
positional information is added to each token
GPT-2 learns positional representations directly
02 Core Processing Layer
each layer contains two core components
Input Embedding
Positional Encoding
12-layer Transformer Decoder Stack

Figure 2: Input layer and transformer decoder

2.1.2. Output layer & training configuration

In order to produce the output of the input layer and the model for the training arrangement, obtain the most controlled text sequence generation process by the decoder, the output generation mechanism maps the hidden states by which the decoder output probability distributions on the vocabulary; also, in order to achieve the controllable text sequence creation of the output.As illustrated by the Figure 3, a warm up + decay mechanism and eventually changing the learning rate in order to balance training stability and convergence efficiency are combined in the dynamic learning rate scheduling strategy used in training optimization as a way of optimization for the Mixed Precision Training (with the FP16 floating-point format). However, by keeping the model accurate and using the Mixed Precision Training method (which uses the FP16 floating-point format, thereby using 40% of the memory). Additionally, in small-batch training situations, gradient accumulation technology is applied, which increases hardware use by repeatedly increasing the number of gradients and uniformly changing the parameters.

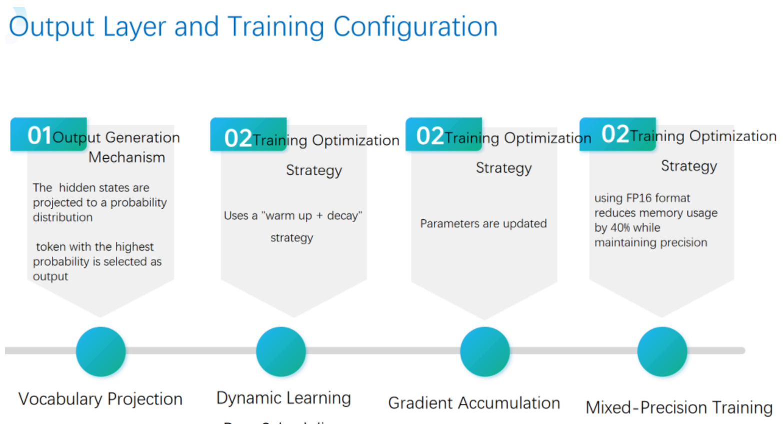

Output Layer and Training Configuration
01 Output Generation Mechanism
The hidden states are projected to a probability distribution
token with the highest probability is selected as output
02 Training Optimization Strategy
Uses a "warm up + decay" strategy
02 Training Optimization Strategy
Parameters are updated
02 Training Optimization Strategy
using FP16 format reduces memory usage by 40% while maintaining precision
Vocabulary Projection
Dynamic Learning
Gradient Accumulation
Mixed-Precision Training

Figure 3: Output layer and training configuration

2.1.3. Key generation techniques

In Key Generation Techniques shown in the Figure 3, models optimize text generation performance through multidimensional strategies. Firstly, Dynamic Context Management automatically truncates when the input length exceeds 1,024 tokens, retaining only the last 900 tokens to balance computational efficiency and contextual relevance. The generation process adopts an auto-regressive generation process, which achieves coherent output through word-by-word iterative prediction.To control the generation style, the model introduces a probability distribution sharpness adjustment that supports multiple output modes ranging from conservative (parameter 0.3), balanced (0.7) to highly creative (1.2). In terms of quality optimization, Top-p (Core) sampling is combined to dynamically screen the minimum candidate word set with a cumulative probability of 90%. Temperature sampling and repetition penalty are also applied to improve diversity while avoiding redundancy. In addition, the truncation strategy further constrains the generation boundary to ensure that the output conforms to the expected length and logic. This approach is similar to the style-guided planning method proposed by Kong et al. [7], which enables more controlled stylistic elements in story generation.The collaborative application of these technologies enables models to achieve an efficient balance between flexibility, accuracy, and creativity, ultimately driving high-quality natural language generation tasks.

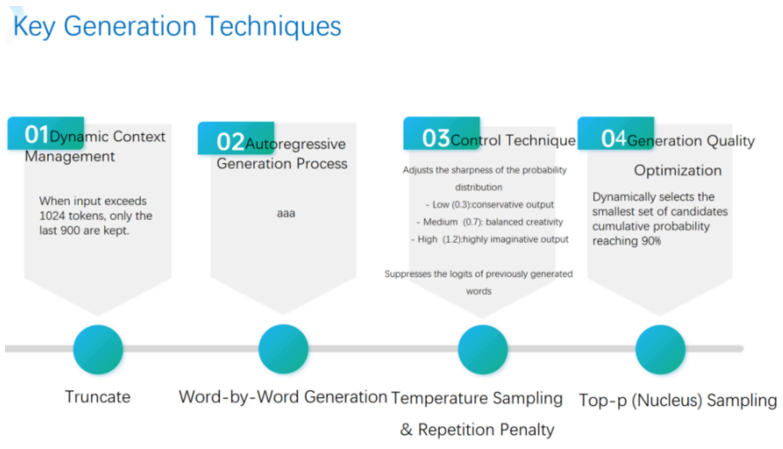

Key Generation Techniques
01 Dynamic Context Management
When input exceeds 1024 tokens, only the last 900 are kept.
02 Autoregressive Generation Process
aaa
03 Control Technique
Adjusts the sharpness of the probability distribution
- Low (0.3):conservative output
- Medium (0.7): balanced creativity
- High (1.2):highly imaginative output
Suppresses the logits of previously generated words
04 Generation Quality Optimization
Dynamically selects the smallest set of candidates cumulative probability reaching 90%
Truncate
Word-by-Word Generation
Temperature Sampling & Repetition Penalty
Top-p (Nucleus) Sampling

Figure 4: Key generation techniques

Table 1: Main hyper-parameters

Hyper-parameters	Value
Attention heads	12 (per layer)
Hidden layer dimension	768
Maximum position code	1024
Word List Size	50257
Fine tuning learning rate	2e-5
Generating temperature	0.7

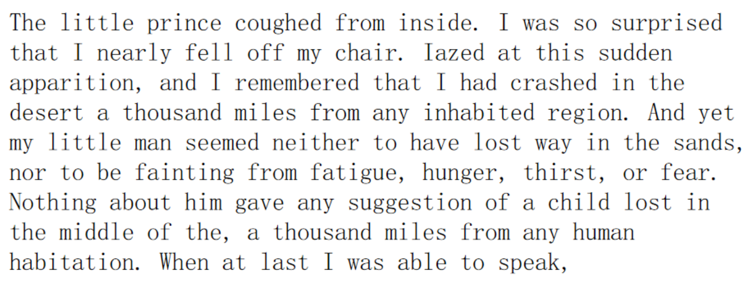
2.1.4. Generated results

The following are the confused evaluations: using 11,000 words of the dataset; and using 10,000 of the total dataset for the training process.

The term "perplexity" has always been used to assess the functionality of language models used in style imitation jobs. Lower ambiguity means having greater performance in the following word in a certain order to forecast the next word.

Another key characteristic that matters most to grammatical fluency, which was usually measured via questionnaires or by human reviews, A greater score for fluency means the produced text is more natural and logical.

The model is able to keep the same way in the generated text according to the principle of consistency. Coherence is the ability of the whole text to flow and make sense. In terms of performance, the model is able to describe and reproduce the target style; therefore, these measures are very important in the evaluation of the performance of style imitation.



```
The little prince coughed from inside. I was so surprised
that I nearly fell off my chair. Iazed at this sudden
apparition, and I remembered that I had crashed in the
desert a thousand miles from any inhabited region. And yet
my little man seemed neither to have lost way in the sands,
nor to be fainting from fatigue, hunger, thirst, or fear.
Nothing about him gave any suggestion of a child lost in
the middle of the, a thousand miles from any human
habitation. When at last I was able to speak,
```

Figure 5: The generation results of fine-tuned GPT2

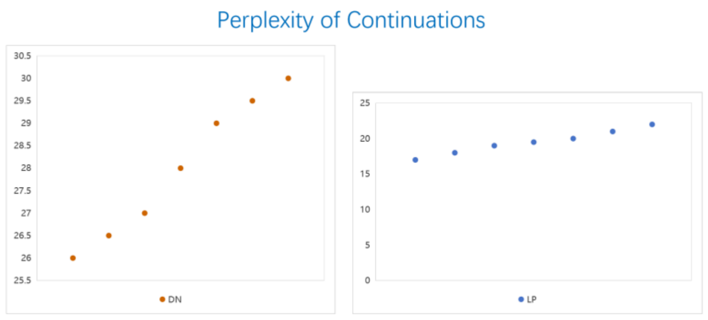


Figure 6: Perplexity of continuations

As shown in the Figure 5 and Figure 6, the produced text illustrates how the model performs exceptionally with regard to fluency and confusion indicators. The narrative structure as a whole is logical, with natural linkages between the two. For instance, the scene is gradually unmistakable; actions, such as "I almost fell off the chair" and "I stared at this suddenly appearing little person in surprise" are the only ones that appear in the scene. The gentle and consistent emotional expression is in line with the original work The Little Prince. The rationality of vocabulary selection (high-frequency terms like "desert" and "habitation") is reflected in the feature of low confusion because it is very appropriate for the contextual situation and the complexity of syntactic structures (like compound sentences) in the context of words (like "Iazed" and "mading") that are near to the distribution of actual texts. Similar evaluation approaches have been used by Bulut and Yildirim-Ersari [8] in their work on story generation for reading comprehension. Nonetheless, there are still local faults in the specifics, such as spelling errors (where "Iazed" should be "Gazed"), and semantic ambiguity (which is caused by comma breaks in the middle of the thousand miles), which is indicative of the model's inadequate mastery of low-frequency word spelling and punctuation rules for lengthy sentences. Even though these mistakes did not substantially affect the model's overall ability to learn language patterns, they do so because they might be residual patterns of particular errors in the training data that are too small to accurately screen candidates.

2.2. A transformer model trained from scratch on a dataset

2.2.1. Self-trained transformer model

This is a model diagram built in the paper, demonstrating the procedure of training with raw data.

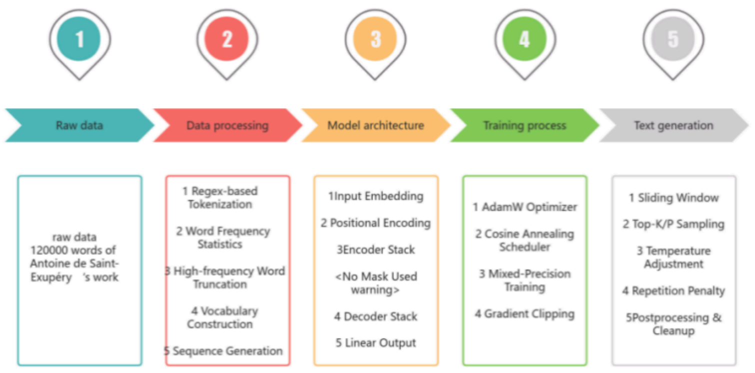


Figure 7: Different parts of self-trained transformer

2.2.2. Structure description

As depicted in the Figure 7, using a transformer model that was trained from the start, data preparation, architecture design, training optimization, and text creation are among the four main steps of the training process in this process. First, the dataset must be segmented in order to extract words from punctuation and optimize the size of the vocabulary by truncating high-frequency words. The text is then transformed into an index sequence to produce input target pairs that will be adjusted to meet the model's training requirements. The embedding layer in the model architecture is in charge of mapping word indices into dense vectors by using sine position encoding to inject sequence order data, following the approach described in the original Transformer paper [2]. The Transformer module consists of an encoder and a decoder. The encoder is in charge of seeing all of the input because there is no masking mechanism, and the decoder masks future lexical elements to preserve auto-regressive properties. The input is fed into both the encoder and decoder during forward propagation, and the hidden state is finally converted into a word list probability distribution by the linear output layer, which predicts the position of the next word element. To improve stability, the model is integrated using gradient pruning and gradient training (GradScaler) in the text generation stage. The output is then subjected to space and temperature regulation procedures, as well as constraints on the input to the final 128 historical word elements. The model diagram that was built for the study shows how to train with raw data.

2.2.3. Generated results

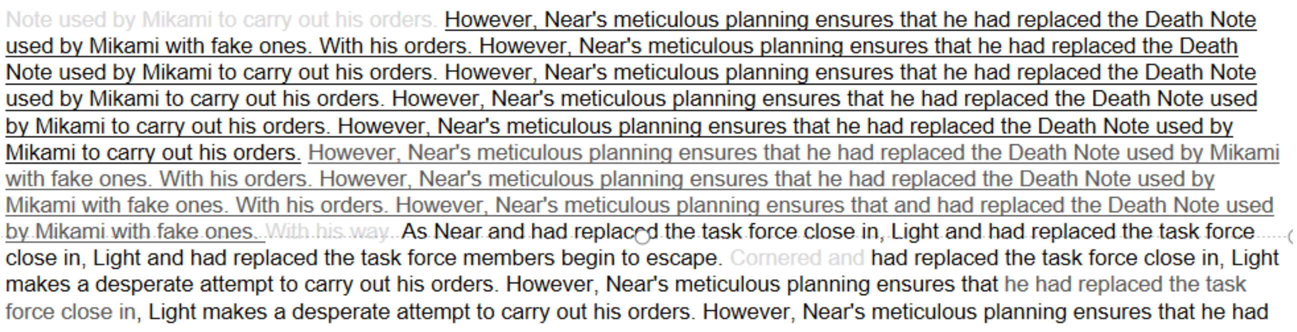

Note used by Mikami to carry out his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami with fake ones. With his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami to carry out his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami to carry out his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami to carry out his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami to carry out his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami with fake ones. With his orders. However, Near's meticulous planning ensures that he had replaced the Death Note used by Mikami with fake ones. With his orders. However, Near's meticulous planning ensures that and had replaced the Death Note used by Mikami with fake ones. With his way. As Near and had replaced the task force close in, Light and had replaced the task force close in, Light and had replaced the task force members begin to escape. Cornered and had replaced the task force close in, Light makes a desperate attempt to carry out his orders. However, Near's meticulous planning ensures that he had replaced the task force close in, Light makes a desperate attempt to carry out his orders. However, Near's meticulous planning ensures that he had

Figure 8: The generation results of self-trained model

From this text generated by Transformer shown in Figure 8, it can be observed that the model exhibits significant underfitting issues when trained from scratch on a dataset of approximately 120,000 words. Although similar sentence structures are repeatedly used in generating text (such as "However, Nears meticulous planning arrangements..."), key nouns, such as "Death Note" and "Mikami" frequently have spelling errors (such as "Milam" and "Nicams") and grammatical confusion (such as "io cay oul is orders"), exposing the model's insufficient learning of vocabulary accuracy and contextual association rules. This problem is further reflected in the rupture of narrative logic: the content is constantly rewritten around the same event but lacks detailed progress, and there are even contradictions before and after (such as the mixture of "Light makes an operate attempt" and "replaced the task force"), reflecting significant deficiencies in the model's ability to capture long-term dependencies and logical coherence modeling. Although Top-K/P sampling and temperature adjustment strategies may have been used, the text is still filled with meaningless fragments such as "mebculous" and "dosporate atompt", indicating that the screening mechanism for candidate words in the decoding stage has not been effectively operated. The root cause is still that the model did not fully grasp the language distribution rules during the training process, which may be related to incomplete language pattern coverage caused by insufficient data volume. This contrasts with the findings of Yang et al. [6], who demonstrated that specialized neural architectures could better generate long-form stories, suggesting our approach might benefit from similar architectural modifications.

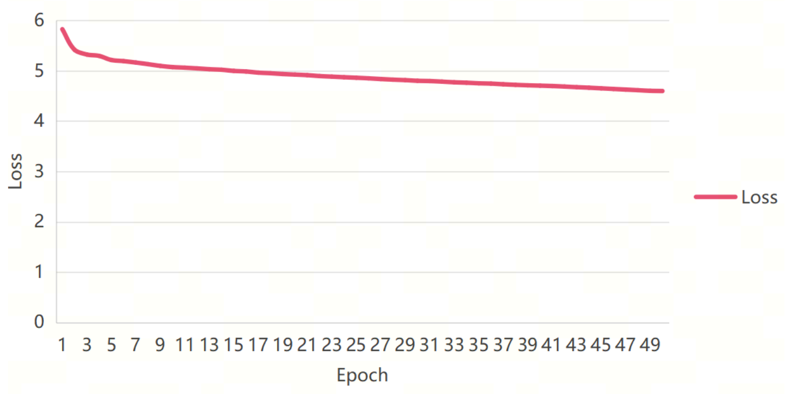


Figure 9: Loss rate

It can be seen from Figure 9 that the model trained from scratch did not perform well on data of around 120,000 words, and the results clearly showed underfitting. This is the loss rate obtained after multiple parameter adjustments, and it is easy to overfitting, indicating that the model has not been able to learn language logic.

2.3. Overall evaluation

2.3.1. Manual evaluation results

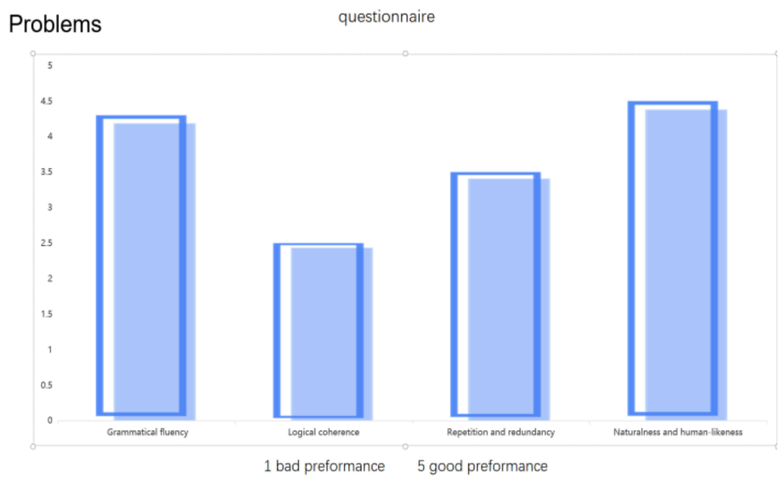


Figure 10: The survey results of GPT2 fine-tuned

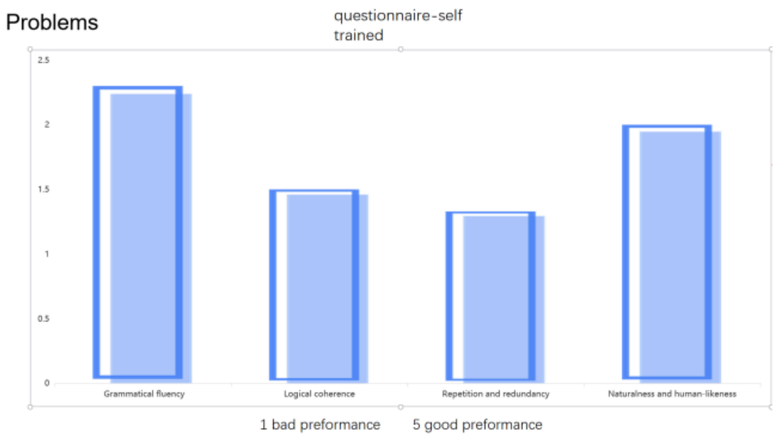
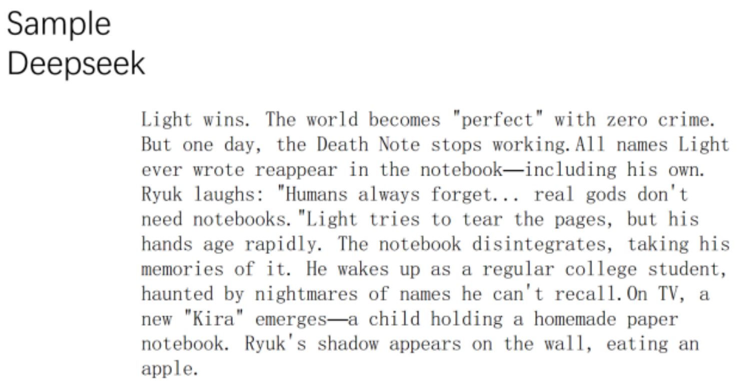


Figure 11: The survey results of self-trained model


Sample
Deepseek

Light wins. The world becomes "perfect" with zero crime. But one day, the Death Note stops working. All names Light ever wrote reappear in the notebook—including his own. Ryuk laughs: "Humans always forget... real gods don't need notebooks."Light tries to tear the pages, but his hands age rapidly. The notebook disintegrates, taking his memories of it. He wakes up as a regular college student, haunted by nightmares of names he can't recall. On TV, a new "Kira" emerges—a child holding a homemade paper notebook. Ryuk's shadow appears on the wall, eating an apple.

Figure 12: The generation results of self-trained model

This is the evaluation result of text generation compared to the results generated by DeepSeekR1 API from four aspects obtained through a questionnaire survey in 100 people. Figure 10 shows the evaluation of the generation result of GPT2 fine-tuning model, Figure 11 shows the evaluation of the generation result of the model trained from scratch, and Figure 12 shows the result generated by DeepSeekR1 API as an successful Comparison. The key issue in text generation is controllability, which means that the model should generate text that meets multiple conditions given input conditions, including grammar, vocabulary, and structure, all of which should conform to the norms of human natural language [9]. Recent work by Xu et al. [10] has demonstrated the importance of incorporating psychological states in knowledge-based story generation, which could potentially improve the emotional coherence of generated narratives in future work. From a manual perspective, it can be seen that the generation of GPT2 fine-tuning models is relatively successful, while the generation of models trained from scratch is relatively unsuccessful.

2.3.2. Analysis of failure causes

This study believes there are three reasons for the failure of models trained from scratch.

Data Limitations: Small datasets (e.g., 120,000 words of short stories) lack diversity.

Training Instability: Without mixed-precision training or gradient clipping, convergence is slow and unreliable.

Design Errors: The encoder doesn’t have masking, the driving information leakage"during training.

One major cause is the dataset limitation because it might be difficult for the models that were trained from the very beginning to perform well on such a small dataset.

3. Conclusion

In the next direction of the model's improvement, the main goal will be to improve data quality, architecture robustness, and optimization strategies. The performance improvement of pre-trained models is mostly dependent on large-scale and varied, high-quality datasets; this can be used to fully utilize the model's capacity for generalization; In order to solve the problem of deep network gradient degradation, the architecture design must also strengthen the support for sequence modeling, such as precisely controlling the visible range of information through masking mechanisms, and combining residual connections to reduce the problem of deep network gradient degradation. Simultaneously, training efficiency will greatly increase with the development of optimization techniques. While gradient accumulation technology obtains comparable large-scale parameter update effects in small batch scenarios, hybrid precision training minimizes the use of video memory by allocating computing resources fairly. Large-scale training of complicated models can be made possible by the synergy of the two. Following approaches similar to those explored by Bulut and Yildirim-Ersari [8] and Kong et al. [7], we could also explore style-guided planning and automatic story generation for specific applications. By incorporating psychological state modeling as demonstrated by Xu et al. [10], we might also enhance the emotional coherence of our generated stories. The model will be able to achieve a better balance between performance, efficiency, and practicality by cooperatively optimizing these directions.

References

[1] Yuxin Huang, Yuan Zhao, Zhengtao Yu, et al. Event-driven Story Generation Method Based on Three-act Structure Thinking Chain and Semantic Self-consistency [J]. Pattern Recognition and Artificial Intelligence, 2024, 37(07): 571-583.

[2] Vaswani, Ashish, Noam Shazeer, et al. Attention is All You Need. Advances in Neural Information Processing Systems, (2017): 5998-6007.

[3] Yawei Sun. Research on Text Generation Technology Based on Pre-trained Language Models under Small Sample Size Constraints [D]. Harbin Institute of Technology, 2021.

[4] Devlin, Jacob, Mingwei Chang, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (2019): 4171-4183.

[5] Radford Alec, Jeff Wu, Child R., Duan L., Amodei D., and I. Sutskever. Language Models are Unsupervised Multitask Learners. OpenAI Technical Report (2019).

[6] Yazheng Yang, Boyuan Pan, Deng Cai, and Huan Sun. TopNet: Learning from Neural Topic Model to Generate Long Stories. IEEE Transactions on Affective Computing 12, no. 4 (2021): 639-653.

[7] Xiangze Kong, Jialiang Huang, et al. Stylized Story Generation with Style-Guided Planning. IEEE Transactions on Affective Computing 13, no. 5 (2022): 1059-1072.

[8] Bulut Okan, Seyma Nur Yildirim-Ersari. Automatic Story and Item Generation for Reading Comprehension Assessments with Transformers. IEEE Transactions on Affective Computing 13, no. 5 (2022): 1045-1058.

[9] Jianshu Chen. Research and Application of Controllable Text Generation Based on Pre-trained Language Models [D]. University of Electronic Science and Technology of China, 2022.

[10] Feifei Xu, Xinpeng Wang, and Shanlin Zhou. Story Generation Using Knowledge Graph Under Psychological States. IEEE Transactions on Affective Computing 12, no. 4 (2021): 654-666.