KuiLeiXi: a Chinese Open-Ended Text Adventure Game

Yadong Xi1; Xiaoxi Mao1†; Le Li1; Lei Lin1; Yanjiang Chen1; Shuhan Yang1; Xuhan Chen1; Kailun Tao1; Zhi Li1; Gongzheng Li1; Lin Jiang1; Siyan Liu1; Zeng Zhao1; Minlie Huang2; Changjie Fan1; Zhipeng Hu1

1 Fuxi AI Lab, NetEase Inc., Hangzhou, China
2 Department of Computer Science and Technology, Institute for Artificial Intelligence, State Key Lab of Intelligent Technology and Systems, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing, China.

{xiyadong, maoxiaoxi, liile, linlei}@corp.netease.com, aihuang@tsinghua.edu.cn

Abstract

There is a long history of research related to automated story generation, dating back as far as the 1970s. Recently, the rapid development of pre-trained language models has spurred great progresses in this field. Equipped with GPT-2 and the latest GPT-3, AI Dungeon has been seen as a famous example of the powerful text generation capabilities of large-scale pre-trained language models, and a possibility for future games. However, as a game, AI Dungeon lacks incentives to players and relies entirely on players to explore on their own. This makes players’ enthusiasm decline rapidly. In this paper, we present an open-ended text adventure game in Chinese, named as KuiLeiXi1. In KuiLeiXi, players need to interact with the AI until the pre-determined plot goals are reached. By introducing the plot goals, players have a stronger incentive to explore ways to reach plot goals, while the AI’s abilities are not abused to generate harmful contents. This limited freedom allows this game to be integrated as a part of a romance simulation mobile game, Yu Jian Love3. Since KuiLeiXi was launched, it has received a lot of positive feedbacks from more than 100,000 players. A demo video is available at https://youtu.be/DyYZhxMRrkk.

1 Introduction

The past few years have seen a significant improvement in the capabilities of neural networks for text generation (Radford et al., 2019; Brown et al., 2020; Zhang et al., 2020b). Large-scale pre-trained language models with tens of billions of parameters are capable of producing human-like text. This capability has spawned a range of revolutionary applications (Roller et al., 2020; Zhang et al., 2020a; Guan et al., 2020). AI Dungeon is a typical example of them. It is an open-ended text adventure game, where players are allowed to create their own adventures in any way they like. The original AI Dungeon is based on GPT-2 large, finetuned on a dataset of text adventures collected online1. Since the launch of its Colab version, AI Dungeon has gained a lot of attention on social networks.

However, from the point of view of game developers, AI Dungeon suffers from several problems, hindering it from becoming mainstream gaming. The first problem is that it relies entirely on players to explore on their own. The lack of incentives may lead to a rapid decline of players’ enthusiasm. The second problem is the boundaryless nature of generated contents. Every game is associated with a certain series of world settings where the stories take place. To integrate AI Dungeon-like technology in a game, considerable adaptation works are necessary. On the other hand, in the absence of necessary guidance and restraints, players tend to abuse AI Dungeon to create malicious or offensive contents4. In areas with more conservative values, it is of high risk to launch an AI Dungeon-like feature in a commercial product.

Considering the problems described above, we extended the original AI Dungeon so that it could be accommodated in a commercial game. When playing AI Dungeon, depending on the player’s choice of different topics, the AI will generate a story beginning, and then the player is free to explore the development of the story. Unlike AI Dungeon, in our game, players need to play a fixed character of their choice and interact with the AI to develop the story according to the pre-defined story background until they reach the specified plot goal to obtain the mission rewards. Multiple plot

---

* Equal contribution
† Corresponding Author
1 http://yujian.163.com/klt-silk/redirect.html
2 https://yujian.163.com
3 https://chooseyourstory.com/
4 https://www.wired.com/story/ai-fueled-dungeon-game-got-much-darker/
goals are contained in a story script. By elaborately design the plot goals, the difficulty of game and the freedom of players to create could be manipulated. The game supports multiplayer, scripts are created both by the game developers and by the players themselves. Because in the gaming process, the player seems to be manipulating the puppet of the character, we figuratively call this game KuiLeiXi, which refers to the puppetry in the Song Dynasty.

Deploying a neural text generation model for many players is quite expensive. So we adopted a range of methods to reduce the cost, including layer drop and knowledge distillation. In addition, we implemented a highly optimized transformer in CUDA for inference. After applying these methods, the inference speed of the model is increased by 10 times, the throughput is increased by 20 times, greatly reducing the deployment cost.

KuiLeiXi has been launched as a part of Yu Jian Love. Yu Jian Love is a mobile romance simulation game where players can role play as a girl that lives in the era of Northern Song Dynasty and develop romantic relationship with different handsome male characters. Since launched, it received a lot of positive feedbacks from players and industry. We hope KuiLeiXi could inspire fellow game developers and NLP researchers to bring more NLP capabilities into games and make game content more dynamic and personalized.

2 Architecture

In this section, we will describe the implementation and optimization of KuiLeiXi in detail. As seen in Figure 1, the system consists of three components: Input Processor, Story Generator and Candidates Ranker. As both the Story Generator and Candidates Ranker are based on our in-house pre-trained language model, we will firstly describe the pre-training details. Then we will present the implementation details of the three components in order. Finally, we will introduce the optimization details for deployment.

2.1 Pre-training

Our in-house pre-trained language model for story generation is based on GPT-2 large. It has 36 layers, 1280 hidden size, 20 self-attention heads, and 725 million parameters. It is pre-trained on a dataset consisted of around 30 gigabytes of Chinese web-novels collected online. The vocabulary size is 13762 and the context length is 1024. In addition, we pre-trained a Roberta-large (Liu et al., 2019) based bidirectional transformer model (Vaswani et al., 2017) on the same dataset. It has 24 layers, 1024 hidden size, 16 self-attention heads and 317 million parameters. We used fairseq\(^5\) for training of the models.

2.2 Input Processor

The input text of a player will firstly be checked by a toxicity detection service\(^6\) to avoid potential risks. It is then processed by a semantic similarity detection model to determine if it is too semantically close to the plot goal. This is to avoid making it too easy for players to reach the plot goal. The semantic similarity detection model is based on Sentence-Bert (Reimers and Gurevych, 2019), trained on the combination of several Chinese NLI datasets (Bowman et al., 2015; Williams et al., 2018; Hu et al., 2020). The virtual adversarial training (Zhu et al., 2020) is also adopted. This approach improves the generalization of the model by adding small perturbations to the input embeddings. For every plot goal, at least three textual descriptions of that goal should be prepared. The input text will be compared with all the textual descriptions of current plot goal. If any of the similarity scores is above a certain threshold, the player will receive a message telling the player to input again. After the input text has passed the toxicity detection and semantic similarity detection, it will be concatenated to the context to form the input for story generation.

2.3 Story Generator

The story generator is in charge of generating consistent and fluent story contents based on the context and player input. In below we will describe in detail how the story generator is implemented.

2.3.1 Finetuning

Because KuiLeiXi is supposed to be launched as a part of Yujian Love, the generated text needs to be consistent with the original stories of the game in terms of language style and backdrop. Therefore, the game’s existing story scripts are critical for finetuning. However, these scripts only contain approximately 2 million tokens, barely enough for effective finetuning. So we carefully selected 10 online novels with similar language styles and backdrops to form an augmented dataset along with
in-game story scripts. For scripts from the game, we assign every line a label indicating if it is dialogue or narrative content, as seen in Figure 2. It is easy because the dialogues and narratives are naturally separated in different lines in the scripts and the dialogues are with double quotation marks. This allows the finetuned model to control the generate subsequent story contents to be dialogue or narrative content. In addition, the label can guide the model to generate more consistent content with the story’s background similar to (Keskar et al., 2019).

### 2.3.2 Inference

**Input Truncation:** At inference, the generation model receives a concatenation of the player input and the previous context as the input. As game continues, the input length will easily exceed the context length 1024. So we need to design a truncation strategy. Naively keeping the latest story context is not feasible in this application, as the pre-written story beginning corresponding to the current plot goal is neccessary to keep the story unfold without straying too far from the current plot goal. Therefore, we keep the pre-written story beginning corresponding to the current plot goal along with the latest story context as the input.

**Decoding Strategy:** We use the top-k sampling (Fan et al., 2018) for decoding. Sampling temperature and $k$ are set to 0.8 and 8 respectively. We observed that the model tend to copy from the input. To alleviate this issue, we adopt the penalized sampling technique (Keskar et al., 2019; See et al., 2019). In general, penalty sampling penalizes words that occur throughout the context by default, reducing their sampling probability. However, we argue that this is inappropriate, especially for penalizing words that are far from the decoding position. The reasons are twofold. Firstly we observed that, the model tends to copy from words closer to the decoding position, rather than a very distant context, like contents with more than 200 words away from the decoding position. Secondly, we conducted statistics in the webnovel corpus, and the probability of the next word appearing in the previous 800 words reached 75%, indicating that copying from context is also common in real world texts. In summary, if the probability of words occurring in very distant contexts is also penalized at inference, the distribution of the generated text will be significantly different from the real world text distribution, which may reduce the generation performance. Therefore, we only penalize the probability of words that have appeared in previous 200 words prior to the decoding position.

Given the input tokens $G[1, 2, \ldots, t]$ and the context window size $c$, the probability distribution $p_i$ for the next token is defined as:

$$p_i = \frac{\exp(x_i/(I(i \in g[(t - c) : t])))}{\sum_j(\exp(x_j/(I(j \in g[(t - c) : t])))})$$

$$I(e) = \theta \text{ if } e \text{ is True else } 1$$
We set $\theta$ to 1.2, which makes a balance between generated text quality and elimination of duplication.

2.4 Candidates Ranker

For each player input, we generate 5 candidate stories for re-ranking. The candidate stories are then sent to the ranker to select the best one to return to the user.

To ensure the quality, we developed a series of filtering rules to remove inappropriate candidates in the stories. Firstly, if a candidate story contains a character name that does not appear in *Yujian Love*, the story will be moved out of the candidates. Secondly, candidate stories that contain a lot of content copied from context will be removed. Thirdly, stories with inappropriate content detected by toxicity detection service will be removed. Fourthly, if a character described in a story behaves inconsistently with his or her gender, that story will also be removed. We trained a discriminator model to detect whether a character in the story behaves inconsistently with his/her gender. The training data is generated automatically. We use the original text as a positive sample and the text after the character name replacement as a negative sample. When replacing character names, a character is replaced with the name of a character having the other gender.

For the remaining stories, we rank them based on the weighted sum of two metrics. The first is the overlapping score, which is calculated based on the overlapping of tokens in the generated story and context. Generally, when the overlapping score is higher, repetition is heavier and will hurt the text quality. The second is the goal matching score, which measures how likely a story entails the current plot goal. Given the list of context tokens $C$, the list of generated story tokens $G$ and the length $l$ of $G$, the overlapping score is defined below:

$$Score_{overlap} = \frac{\sum I(i \in G)}{l}$$

$$I(i) = 1 \text{ if } i \text{ in } G \text{ else } 0$$

Determining whether a story contains a specified plot is a typical textual entailment problem. However, because players can create story scripts and submit them to the game community, it is intractable to create a dataset dealing with numerous possible plot goals. So we had to approach the problem from a different angle. We argue that it is easier to solve this problem by transforming it into a problem similar to Next Sentence Prediction (NSP), i.e., determining whether a plot goal can be coherently connected to the generated story. It is well known that the original NSP task proposed in BERT (Devlin et al., 2019) is too easy, many latest pre-trained language models have abandoned it (Liu et al., 2019; Lan et al., 2019). We argue that discriminating the randomly sampled negative examples is relatively easy so we adopt a novel strategy to enhance the difficulty of NSP. When generating the training dataset, in addition to the randomly sampled sentences, we also take the next sentence of next sentence as a negative sample with a certain probability. We finetuned the pre-trained Roberta-large based model as described in Section 2.1 on this generated dataset. The finetuned model is then used as a discriminator to detect whether the plot goal can be smoothly connected to the generated story.
Figure 3: The script and character selection interfaces

Figure 4: The screenshot at the beginning of the game. The right figure is the English translation of the left figure. Text in the orange box shows the story background. Text in red box shows the current plot goal. Text in the white boxes are the players’ inputs. Text in the purple boxes are the generated stories.
2.5 Optimization

Our original story generation model is of 36 layers and 725 millions of parameters. It takes around 10 seconds to generate a piece of story with one RTX 2080ti, which is totally unacceptable. To improve the inference speed, we need to compress the original model. We firstly adopted the layerdrop technique (Fan et al., 2020), reducing the number of layers to 20. We then used the knowledge distillation technique (Hinton et al., 2015) to distill this 20-layer model. Finally, we finetuned the distilled model over the story dataset. Our experiments showed that combining layerdrop and knowledge distillation performs better than directly performing knowledge distillation.

In addition, we optimized the incremental decoding implementation in fairseq to reduce computation overhead. We developed custom CUDA kernels for better support of long sequence and large hidden size. We also developed an inference server supporting dynamic batching and variable input length. After applying these methods, the inference speed is increased by 10 times, the throughput is increased by 20 times. We have integrated these optimization techniques into a python library named as Easy and Efficient Transformer (EET). It has been opensourced at https://github.com/NetEase-FuXi/EET.

3 Demonstration

In this section, we demonstrate how to play KuiLeiXi.

First, we demonstrate how to start a game. After entering the game, if there is no ready game for joining, you can click the create stage button to start a new game. You then need to pick a story script from the candidates, as demonstrated in Figure 3a. The scripts are both created by game developers and players. Scripts submitted by players will be voted by all players and the winners become playable. After picking the script, you need to choose the character you want to play, as demonstrated in Figure 3b. The playable characters in each script are different. Wait for other players joining your game until the number of players exceeds the minimum player limit. Then you could either start the game or wait for other players joining as additional characters or audience.

After the game starts, all players can see the story background as well as the first plot goal. Players will play in order. The order is randomly decided at the start of the game and does not change during the game. When it is your turn to play, you can choose to write a dialogue with your character or describe a narration. Figure 4a shows the situation at the beginning of the game. Overall, you need to consider the development of the current story, the persona of the character you play and the plot goal. After completing the input, the AI will generate the corresponding story to unfold based on the input,
so on and so forth until the current plot goal is reached. Once the current plot goal is reached, the AI will show a pre-written storyline, as well as a new plot goal, as seen in the Figure 5a. Usually a script has multiple plot goals. When the final plot goal is reached, players win and are rewarded with game props. The whole process of playing will be saved in the database, and players can share it with their friends or make it public on social networks.

4 Conclusion

In this paper, we demonstrate KuiLeiXi, an open-ended text adventure game in Chinese. In order for it to be released as part of a commercial game, we have made many innovations based on AI Dungeon. We believe that the current advances in NLP technology can not only reduce the cost of game content development to a certain extent, but also make the game world more dynamic and personalized. We hope our work will be of interest to fellow game developers and NLP researchers. In future work, we will further explore the generation of game quests and ambient dialogues with up-to-date NLP techniques.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A Knowledge-Enhanced Pre-training Model for Commonsense Story Generation. Transactions of the Association for Computational Linguistics, 8:93–108.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Hai Hu, Kyle Richardson, Liang Xu, Lu Li, Sandra Kübler, and Lawrence Moss. 2020. OCNLI: Original Chinese Natural Language Inference. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3512–3526, Online. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. arXiv preprint arXiv:1909.05858.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. CoRR, abs/1909.11942.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.
Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. 2020. Recipes for building an open-domain chatbot. arXiv preprint arXiv:2004.13637.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019. Do massively pretrained language models make better storytellers? In Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), pages 843–861, Hong Kong, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Rongsheng Zhang, Xiaoxi Mao, Le Li, Lin Jiang, Lin Chen, Zhiwei Hu, Yadong Xi, Changjie Fan, and Minlie Huang. 2020a. Youling: an AI-assisted lyrics creation system. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 85–91, Online. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, Fanchao Qi, Xiaozhi Wang, Yanan Zheng, Guoyang Zeng, Huanqi Cao, Shengqi Chen, Daixuan Li, Zhenbo Su, Zhiwei Hu, Yadong Xi, Minlie Huang, Wentao Han, Jie Tang, Jiantao Li, Xiaoyan Zhu, and Maosong Sun. 2020b. Cpm: A large-scale generative chinese pre-trained language model.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freellb: Enhanced adversarial training for natural language understanding. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net.

5 Appendices

In the figures below, we present a complete gameplay record of KuiLeiXi.

![Figure 6: The first part of the game play.](image-url)
Figure 7: The second part of the game play.

Lv Mei: "Sure, sure."
Lv Mei was just about to pay, but her expression changed.
Fang Zhaozhao: "What's wrong?"
Lv Mei was shocked: "I lost my wallet!"

Current target: Fang Zhaozhao caught the thief.

Thief
Ah.

Fang Zhaozhao: "Then let's hurry back and find it."
Lv Mei and Fang Zhaozhao returned to Xichun House together, but did not see Fang Zhaozhao's figure. Lv Mei couldn't help but wonder if he had gone far away. Lv Mei looked around for him and suddenly saw a familiar back.

Fang Zhaozhao
It must be the thief. I will catch him back.

Lv Mei immediately ran over, only to find that Fang Zhaozhao was gone. Fang Zhaozhao: "I'm here!"
Lv Mei is shocked to find that Fang Zhaozhao has caught the thief.

Fang Zhaozhao reached the target: Fang Zhaozhao caught the thief.

Fang Zhaozhao picked up the thief and threw him to the ground.
Lv Mei in anger: "You have hands and feet. You can do anything good, why do you want to steal things?"
Thief: "How can you rich families understand the suffering of the poor people. I come from a poor background. When I was a child, I couldn't afford to study. I couldn't be a big shot, and now I do some small business. A hard, but the other day it was banned by the government!"
Eyes of the thief turned red: "Several children are at home, still waiting for me to earn money to support..."

Current target: Lv Mei wanted to help the thief.

Madam
The duke.

Fang Zhaozhao: "Madam, I'll handle this matter!"
Fang Zhaozhao pulled the thief aside. Fang Zhaozhao: "Did you steal something?"
Thief: "I don't care to steal it, I just took a casual look. I didn't expect it to be so expensive. I'm a small business. I don't make much money."

Figure 8: The third part of the game play.

Lv Mei
Madam, is there still a shortage of handymen in your store? Take him in as a favor and give him some pay.

Madam: "Hey, my children at home are studying. How can I leave it to him?"
Lv Mei: "Then you let him follow us, it is all the same to help."
Madam: "That's an idea."
Madam brought the thief to Lv Mei.
Madam: "What's your name?"

Thief
Not telling you.

Lv Mei: "Then what do you want?"
Thief: "I... I came here to do some jobs, I... I don't have any silver."
Lv Mei: "Then I will give you money. You work here for a month and earn enough silver before you can leave."

Fang Zhaozhao
Lv Mei, don't think I don't know you secretly helped that thief.

Lv Mei lowered her head in embarrassment. Lv Mei picked up the thief from Fang Zhaozhao's hand, and he looked at Lv Mei with a disgusted look on his face. Thief: "Hm, how can you fool me with that little trick of yours?"

Madam
Then what else do you want to do?

Thief: "I'm going to go and settle the score with that bad guy!"
Fang Zhaozhao: "Don't, I'll accompany you."
Fang Zhaozhao took the thief away. Lv Mei was bored alone, so she went to a nearby tea stall to drink tea.

Lv Mei
I'll give you some money, you stop stealing and go do a proper business.

Fang Zhaozhao came down from the tea stall and handed Lv Mei a package. Fang Zhaozhao: "You take this, there is no need to be formal in front of me." Fang Zhaozhao: "I still have things to do, so I won't play with you anymore. See you later."
Figure 9: The fourth part of the game play.

Figure 10: The fifth part of the game play.