MOBA-E2C: Generating MOBA Game Commentaries via Capturing Highlight Events from the Meta-Data

Dawei Zhang[1,3]*, **Sixing Wu**[2,3], **Yao Guo**[1], **and Xiangqun Chen**[1]†
[1]Key Laboratory of High-Confidence Software Technologies (MOE),
School of Computer Science, Peking University, Beijing, China
[2] School of Computer Science, Peking University, Beijing, China
[3]GamesMind Technology, Beijing, China

Abstract

MOBA (Multiplayer Online Battle Arena) games such as Dota2 are currently one of the most popular e-sports gaming genres. Following professional commentaries is a great way to understand and enjoy a MOBA game. However, massive game competitions lack commentaries because of the shortage of professional human commentators. As an alternative, employing machine commentators that can work at any time and place is a feasible solution. Considering the challenges in modeling MOBA games, we propose a data-driven MOBA commentary generation framework, *MOBA-E2C*, allowing a model to generate commentaries based on the game meta-data. Subsequently, to alleviate the burden of collecting supervised data, we propose a *MOBA-FuseGPT* generator to generate MOBA game commentaries by fusing the power of a rule-based generator and a generative GPT generator. Finally, in the experiments, we take a popular MOBA game Dota2 as our case and construct a Chinese Dota2 commentary generation dataset *Dota2-Commentary*. Experimental results demonstrate the superior performance of our approach. To the best of our knowledge, this work is the first Dota2 machine commentator and *Dota2-Commentary* is the first dataset.

1 Introduction

With the development of the live-streaming service[1] and the e-sports industry, increasing game fans now are addicted to watching online live-streaming (Yang et al., 2022). In a live-streaming channel, besides the game video, a professional streamer makes commentaries to vividly narrate the game's progress (Ishigaki et al., 2021); thus, audiences can enjoy a game more interestingly and easily by following a game live-streaming, especially when they do not have enough background knowledge.

Figure 1: A channel provides both a video and commentaries. This screenshot is captured from Twitch.

Nonetheless, not all e-sports game competitions can be narrated by live-streaming channels because 1) Streamers often only select famous game competitions in their channels to attract more audiences; 2) Unlike traditional sports such as basketball and football that have a high threshold for hosting and live-streaming, an e-sports game only requires several computers with the Internet, and thus the number of e-sports game competitions is always larger than the number of live-streaming channels; 3) The game rules are complex (OpenAI, 2019) and can be changed at any time; thus, only a few experienced people are qualified for this job.

User preferences are diverse, and thus it is not trivial to provide professional commentaries to every game competition. A feasible way is to deploy machine commentators, which can work whenever and wherever possible in a low-cost way. However, building a machine commentator is challenging (Ponomarenko and Sirotkin, 2020). *First*, it is unrealistic for machines to generate comments by watching videos like human beings because understanding a game video requires a great deal of technology and workforce. *Second*, making commentaries to narrate e-sports games requires tremendous knowledge (domain & peripheral); *Third*, a machine commentator must have the ability to capture the highlight events of a game. Consequently, we can hardly find such products on the market.

[*] The email of the first author: daweizhang@pku.edu.cn
[†] The corresponding author.
[1] For example, Twitch, https://www.twitch.tv/

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 4574 - 4585
December 7-11, 2022 ©2022 Association for Computational Linguistics

This paper explores the research field of the game commentary generation and mainly focuses on MOBA (Multiplayer Online Battle Arena) games such as Dota2 and League of Legends. With the mentioned challenges in mind, we propose a data-driven MOBA game generation framework *MOBA-E2C* along with a *MOBA-FuseGPT* generator. Instead of making commentaries based on the visual feature (i.e., video), the commentaries are narrated based on the meta-data of a game. In short, *MOBA-E2C* first uses several types of event handlers to capture the highlight events that need to be narrated from the game's meta-data. Each event uses a table of key-value attributes to record the necessary content, and then the game commentary generation problem can be subsequently regarded as a data-to-text generation task. Collecting and constructing supervised data is thorny. Finally, to generate high-quality commentaries and reduce the requirement of supervised data, our data-to-text generator *MOBA-FuseGPT* takes advantage of both the rule-based method and the pre-trained language modeling (i.e., GPT2).

In experiments, a popular MOBA game *Dota2* (Yu et al., 2018) is adopted as the case. We first designed 34 different event handlers for *MOBA-E2C* to capture highlight game events. Subsequently, we collected and constructed a Chinese Dota2 commentary generation dataset *Dota2-Commentary*, which includes 234 recent Dota2 game sessions[2] and 7,473 commentaries generated by professional human annotators. Experimental results have shown our approach has great improvement in both the default scenario and the few/zero-shot scenario. To the best of our knowledge, this work is the first Dota2 machine commentator, and *Dota2-Commentary* is the first dataset.

The contribution of this work is four-fold:

- We propose a MOBA game commentary generation framework *MOBA-E2C*, which generates commentaries based on the meta-data.
- The proposed generator *MOBA-FuseGPT* can take advantage of both the rule-based method and the pre-trained language model.
- We construct a MOBA game commentary generation dataset *Dota2-Commentary*.
- Extensive experiments and analyses demonstrated the effectiveness of our approach.

2 MOBA-E2C

Instead of using visual features, we propose a data-driven framework *MOBA-E2C* (E2C: Event-to-Commentary) to generate commentaries from the meta-data of a game.

2.1 General Paradigm and Overview

Although there are various MOBA games in the market, such as Dota2, LOL, and Honor of Kings, they follow a similar paradigm. Generally, two *teams* of *players* compete against each other on a predefined *map*, where each *player* controls a $hero$ with a set of $abilities$ and $items$. The objective is to destroy the opponent's *buildings*.

As shown in Figure 2, the work-flow of *MOBA-E2C* can be summarized as 1) constructing a live-streaming sequence of game states by collecting meta-data from the corresponding MOBA game; 2) using event handlers to capture highlight events that should be narrated from the live-streaming sequence. Each captured highlight event is formulated as a table of a set of key-value attributes; 3) employing a table-to-text generator to generate commentaries based on the identified event tables.

2.2 Live-Streaming States

MOBA-E2C first regards each game session as a live-streaming sequence of game states $S = (s_1, s_2, \cdots, s_n)$. At each time t, the corresponding game state $s_t = \{o_{i,t}\}$ records the current game progress data using the MOBA objects $\{o_{i,t}\}$.

Based on the empirical knowledge of game experts, we have designed several universal MOBA objects. As shown in Table 1, *MOBA-E2C* uses 7 different kinds of object to cover objects in a MOBA game. Specifically, for each game session, we use a *map* object to record the meta-information, such as the time, game state, etc. Next, we use two *team* objects to represent two opponents, and each *team* object includes n *player* objects. Each *player* object has a *hero* object, some *item* objects, and some *ability* objects. Finally, each team also includes some *building* objects.

For each MOBA object, we use a lot of key-value pairs to illustrate its attributes, where the key is the attribute name and the value is the corresponding attribute value. For example, in a $hero$ object, we use a *health* attribute to track the current health, a *level* attribute to track the current level.

[2] 70 supervised, 164 unsupervised

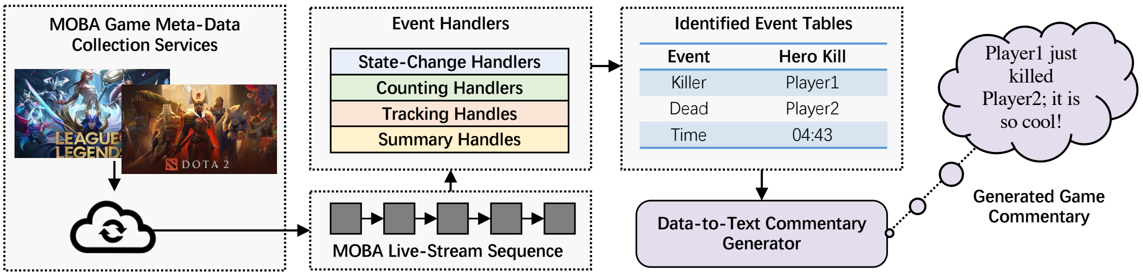


Figure 2: The work-flow of the proposed data-driven framework *MOBA-E2C*.

Objects	Description
map	the meta information (time, etc.)
$team_i$	each game has two opponents.
$building_i$	buildings to be defended/attacked.
$player_{i,j}$	each $team_i$ has n players.
$hero_{i,j}$	each $player_{i,j}$ controls a $hero_{i,j}$
$item_{i,j,k}$	each $hero_{i,j}$ has its own *items*
$ability_{i,j,k}$	each $hero_{i,j}$ has its own *abilities*

Table 1: MOBA Objects.

2.3 Highlight Event Identification

Intuitively, the commentaries are needed if some highlight events happen at the time t. Hence, the next job is to capture highlight events based on the current live-streaming states $s_{1:t}$. *MOBA-E2C* uses several event handlers to capture events that should be commentated on by monitoring the game state s_t and tracking the change between the current state s_t and the previous states $s_{1:t-1}$. Specifically, based on the knowledge of experienced human players and commentators, we propose four different types of event handlers to capture highlight events:

1. *State-Change:* By checking the difference between the current object $o_{i,t}$ and the last $o_{i,t-1}$, we can figure out an event has just happened. For example, by comparing the health of $hero_i$, we can find $hero_i$ has just died.

2. *Counting:* It monitors and counts some aspects of an object. Once the counting value reaches a milestone, it generates an event. For example, once $hero_i$ has killed others 10 times, there will be a corresponding event.

3. *Tracking:* It continuously tracks some aspects. Once the current situation meets some rules, a new event can be identified. For example, if $hero_i$ has died, a tracking handler is created to track and tell the progress of revival.

4. *Summary:* It generates an event by analyzing some aspects periodically; for example, summarizing the net worth of two teams.

Each identified event is subsequently represented as an event table $e = \{(k_i, v_i)\}^l$. Each attribute (k_i, v_i) is a key-value pair; it describes an attribute of the event. For example, given a *HeroKill* event table e_{hk} = {*(Killer, player1), (Dead, player2)*}, it means $player1$ has killed $player2$.

2.4 Decoupled Commentary Generation

The last step is to generate a commentary for each identified event table. *MOBA-E2C* decouples the module of commentary generation because game rules are frequently updated; thus, *MOBA-E2C* can quickly adapt to the newest version game by updating the non-training upper event handlers. Consequently, this job can be regarded as a data-to-text task, only referring to the given event table during the generation. To adapt to different scenarios, this work proposes three different generators, a rule-based *MOBA-RC*, a generative *MOBA-GPT*, and a fused *MOBA-FuseGPT*.

2.4.1 MOBA-RC

Rule-based methods have been widely used in building machine text generators because 1) they are easy to develop and do not require training; 2) they can generate accurate commentaries with pre-defined rules; 3) they can quickly adapt to changes in the upper logic (i.e., the rules of the game).

Considering such advantages, we first design a rule-based generator *MOBA-RC*. Specifically, for each type of event table, we first design multiple pre-defined different patterns. For example, for a *HeroKill* event, we can design patterns as the following:

- {{*Killer*}} has just killed {{*Dead*}}
- {{*Dead*}} has been killed by {{*Killer*}}

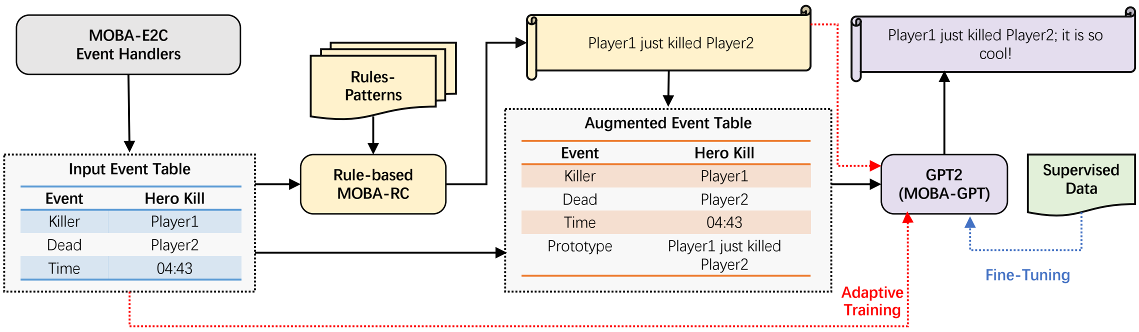


Figure 3: The proposed *MOBA-FuseGPT* fuses the powers of both the rule-based method (MOBA-RC) and the generative method (MOBA-GPT).

where each $\{\{k_i\}\}$ represents a placeholder and can be filled by the corresponding event table key-value attribute (k_i, v_i). Subsequently, given an event table e, *MOBA-RC* can generate a commentary by selecting a pattern and filling the placeholders.

2.4.2 MOBA-GPT

To break the limitation of fixed patterns, we subsequently propose a generative model to learn to generate commentaries. Considering the shortage of supervised event-commentary training data, the proposed *MOBA-GPT* makes use of the pre-trained GPT2 (Radford et al., 2019) as our generative backbone model. The pre-trained model can transfer the knowledge from the large-scale unsupervised training data to the small-scale downstream applications, alleviating the data shortage (Li et al., 2021b).

Similar to other common pre-trained language models (Zhang et al., 2021), GPT2 can only operate plain text. Hence, in the training, given a supervised training instance (e, Y), where $e = \{(k_i, v_i)\}^l$ is the input event table and $Y = (y_1, \cdots, y_m)$ is the target commentary, we first construct a plain text sequence I as an input:

$$I = (T_s, T_g, \theta(e[1]), \cdots, \theta(e[l]), T_c, Y, T_e) \quad (1)$$

where T_s and T_e are two special symbols to indicate the start and the end of the sequence, T_g and T_c are another two special symbols to indicate the start of the linearized table input and the start of the commentary; $\theta(e[i])$ linearizes the i-th key-value attribute (k_i, v_i) of e to the plain text with two special symbols T_{Key} and T_{Value}:

$$\theta(e[i] = (k_i, v_i)) = (T_{Key}, k_i, T_{Value}, v_i) \quad (2)$$

Afterward, the training (fine-tuning) objective can be formulated as minimizing the following negative log-likelihood:

$$\mathcal{L} = \sum_{t=t_Y:|I|} -log(P_{GPT}(y_t|y_{1:t-1}, I_{1:t_Y-1})) \quad (3)$$

where t_Y corresponds to the start position of Y.

2.4.3 MOBA-FuseGPT

Rule-based *MOBA-RC* generates commentaries based on the predefined rules; hence, it can accurately generate commentaries, and quickly adapt to new types of events by adding rules. However, limited by the fixed patterns, generated commentaries lack enough diversity. On the other hand, generative *MOBA-RC* is no longer limited by the fixed patterns, but the training requires a large amount of supervision data, which is a thorny challenge in the context of generating game commentaries. Although using pre-trained language models can alleviate this issue more or less, the domain/task-specific knowledge is still in shortage. Consequently, both methods are still not satisfactory enough.

As illustrated in Figure 3, to reduce the impact of the shortage of supervised data and improve the performance in the few-shot scenarios, *MOBA-FuseGPT* further augments the *MOBA-GPT* by infusing the power of the rule-based *MOBA-RC*.

Adaptive Training Although a pre-trained GPT2 model can transform the implicit knowledge learned from the large-scale unsupervised data to the game commentary generation, there is still a gap because the task and the domain are totally different. Consequently, we propose to conduct pseudo-supervised adaptive training before the fine-tuning. Given a set of game sessions, we first identify the corresponding event tables and then use the rule-based *MOBA-RC* to generate a set of commentaries. Thus, we can obtain a set of event-commentary pairs $\{(e, Y^{'})\}$. Such event-commentary pairs can be regarded as pseudo-supervised training data to adjust the GPT2 before fine-tuning on the human-annotated $\{(e, Y)\}$.

Prototype-Augmented Generation Generating texts with prototypes has shown great potential in text generation (Wu et al., 2019). Inspired by them, we propose a prototype-augmented generation by regarding the commentary generated by *MOBA-RC* as the prototype commentary. Specifically, given an event table e, we first adopt the *MOBA-RC* to generate a commentary $Y^{'}$; then, we can construct a new event table:

$$e^p = e \cup (Prototype, Y^{'}) \tag{4}$$

where $Prototype$ is a special key to indicate the usage of $Y^{'}$. Thus, by using the augmented $\{(e^p, Y)\}$, the model can be more effective in generating the target Y by referring to the $Y^{'}$.

3 Experiment

This paper takes Dota2, one of the most popular MOBA games, as our case. This section describes the construction of the Dota2 implementation of *MOBA-E2C* and the dataset *Dota2-Commentary*.

3.1 Dota2 Event Handlers

We implement a *MOBA-E2C* for Dota2. Thus, as listed in Appendix A, we have designed 34 event handlers for capturing different Dota2 highlight events, where 8 of them are regarded as the zero-shot event types in the dataset partition.

3.2 Dataset

To our best knowledge, there is no open-released supervised Dota2 commentary generation dataset or other MOBA commentary generation dataset. To this end, this paper constructed a MOBA commentary generation dataset, *Dota2-Commentary*[3].

Data Source We collected 234 Dota2 game sessions, where each session is a GSI[4] file. Then, we extracted live-game states of each game session and employed 34 different event handlers to capture events for all 234 game sessions. Subsequently, 70 sessions of them were used to create supervised event-commentary data by humans; the remaining 164 sessions were used to construct pseudo-supervised event-commentary data for the adaptive training.

3.2.1 Supervised Instances

We employed two annotators to generate high-quality event-commentary data. Two annotators have 3+ years of gaming experience. The annotation process can be summarized as follows:

1. We designed a tool for the annotation (see Appendix B). Annotators were required to write a commentary based on the given event table and the corresponding game replay video.

2. To ensure the quality of human commentaries, the tool will automatically check the submitted commentary. If a submission is too similar to the existing commentaries, this submission will be rejected and the corresponding volunteer is required to make a new commentary.

The whole process lasted about one month, and we finally obtained 7,473 human commentaries.

Dataset Partition We divided the selected 70 Dota2 game sessions into two parts: **Seen** contains 60 sessions and **Unseen** contains 10 sessions. Similarly, the event types have been decided into two parts: **Default** contains events generated by 26 types of event handlers and **ZeroShot** contains events generated by the remaining 8 event handlers.

As reported in Table 2, the model is trained/validated on the **Seen+Default** data. In the test, to deeply demonstrate the performance in different scenarios, there are four different test sets, namely, **Seen+Default**, **Seen+ZeroShot**, **Unseen+Default**, **Unseen+ZeroShot**.

3.2.2 Pseudo-Supervised Instances

For each session, we employ the rule-based *MOBA-RC* to generate commentaries and then construct pseudo-supervised training instances. *MOBA-RC* generates a commentary by randomly selecting a pattern; thus, to improve the number of pseudo-supervised instances, we repeat the generation three times (selected from 1,3,5) for each session.

3.3 Settings

3.3.1 Comparison Models

We first evaluated the following data-to-text models, which generate commentaries by using the linearized event tables : 1) *S2S:* Seq2Seq model (Sutskever et al., 2014) has been widely used in the field of text-generation. In this implementation, the encoder is a 2-layer 768d bi-GRU network, the decoder is another 2-layer 768d GRU

[3] https://github.com/gamesmind/MOBA-E2C

[4] This is an official Dota2 format

	Supervised							*Pseudo Supervised*	
Partition	*All*	*Training*	*Dev*	*Test*				Training	Dev
				Seen	*Seen-ZS*	*Unseen*	*Unseen-ZS*		
Session	60 Seen + 10 Unseen	Seen	Seen	Seen	Seen	Unseen	Unseen	164	
Events	26 Default + 8 ZeroShot	Default	Default	Default	Zero	Default	Zero	Default	
Instances	7,473	5,064	500	304	304	1,242	59	175,627,	5,000

Table 2: The statistics of event-commentary instances in *DOTA2-Commentary*.

network. The tokenization and vocab use the solution of the following *BERT*. 2) *BERT*: Based on *S2S*, it replaces the encoder to a pre-trained Chinese BERT encoder *hfl/chinese-bert-wwm-ext* (102M parameters, 768d, 12L, 8H, 21,128 subwords (Cui et al., 2021). 3) *MOBA-RC:* The rule-based Dota2 commentary generator proposed by this paper. 4) *MOBA-GPT:* The Chinese MOBA GPT2 pre-trained/trained by us. 5) *MOBA-FuseGPT*: The proposed method. Besides, we also evaluated several re-writing models. Specifically, given an event table e, we first employ the rule-based *MOBA-RC* to generate a commentary $Y^{'}$, and then force the model to learn to generate ground-truth commentary Y, i.e., $P(Y|Y^{'})$. Similar to the data-to-text models, the re-writing models include *S2S+RW*, *BERT+RW*, and *GPT2+BW*.

Implementation Codes were implemented by PyTorch, and the hugging-face transformer[5]. In the (fine-tuning) training, the batch size is set to 32, GPT2 and BERT use AdamW optimizer and 1e-5 learning rate, and other modules use Adam optimizer and 1e-4 learning rate. In the inference stage, we adopt greedy decoding to generate commentaries. Such codes run on a NVIDIA-RTX2080Ti/3090.

The Pre-training of GPT2 We find the resources of general-purpose small-size Chinese GPT2 models are rare. Thus, we pre-trained a Chinese GPT2 with two NVIDIA-RTX3090 by ourselves. In detail, the GPT2 configuration is 768d,12L, and 12H. The vocabulary includes 30K subwords and 200 special symbols, and the maximum length is 1,024. We trained this GPT2 on a Chinese corpus, which includes 113M utterances/5.22B tokens. The batch size is 512, the optimizer is AdamW, the learning rate is 1.5e-4, 4000 warm-up steps, and 640,000 training steps.

Adaptive Training *MOBA-FuseGPT* has an additional adaptive learning process. We find if we keep using 32 batch size and 1e-5 learning rate, the model tends to be over-fitted. Thus, compared to the fine-tuning, the batch size is increased to 1,024, and the learning rate is also increased to 1e-4.

3.3.2 Metrics

The evaluations were conducted at the character level because models use different tokenization solutions. We used the *F1* (Unigram-F1), *RG* (Rouge-L)(Lin, 2004), *BLEU* (BLEU-4) (Papineni et al., 2002) to evaluate the character-overlapping relevance; we also use the embedding-based *EM-A* (Embedding-Average) and *EM-X* (Embedding-Extreme) to the semantic relevance (Liu et al., 2016). To evaluate the diversity and the informativeness, following (Zhang et al., 2020), we report the *D2* (Distinct-2) and the 4-gram entropy *Ent*.

4 Results

4.1 Evaluation of rules and event handlers

	Unacceptable	*Neutral*	*Acceptable*
Proportion	4.0%	23.5%	**72.5%**

Table 3: The evaluation of rules and event handlers.

The rule-based *MOBA-RC* plays an important role in the *MOBA-E2C*. It is necessary to check the correctness of the pre-defined rules and patterns. Therefore, we employed two volunteers to validate the effectiveness of pre-defined rules and patterns. We used *MOBA-RC* to generate commentaries for 6 different Dota2 game sessions; then, we sampled 100 cases from the generated commentaries and employed humans to annotate. As reported in Table 3, commentaries generated by *MOBA-RC* are highly usable, only 4% are unacceptable. This demonstrates rules and patterns are well-defined, which can conduct the job accurately.

4.2 Automatic Results

We have reported results in Table 4. By comparing the geomean scores of models, we can see that *MOBA-FuseGPT* has achieved the best overall performance in every group, demonstrat-

[5] https://huggingface.co/

Test Set	*Test Seen*									*Test Seen-ZS*								
Model	F1	RG	BLEU	EM-A	EM-X	D2	Ent	**Mean**	**Ratio**	F1	RG	BLEU	EM-A	EM-X	D2	Ent	**Mean**	**Ratio**
S2S	33.7	26.8	13.2	0.72	0.71	6.4	5.5	5.78	0.74	22.5	17.6	6.2	0.67	0.63	2.0	3.9	3.61	0.50
BERT	47.5	40.5	22.4	0.80	0.77	27.9	7.4	9.18	1.17	35.6	29.1	11.7	0.76	0.75	13.7	6.5	6.71	0.92
MOBA-GPT	48.8	43.1	25.8	0.80	0.79	27.2	7.3	9.47	1.21	43.5	**37.5**	20.6	**0.78**	**0.77**	12.0	6.5	7.68	1.06
MOBA-RC	34.7	28.6	14.9	0.73	0.72	30.3	**7.6**	7.82	1.00	36.1	28.5	18.5	0.75	0.72	**14.5**	7.1	7.26	1.00
S2S-RW	37.5	30.7	14.7	0.74	0.72	11.5	6.1	6.75	0.86	25.6	21.3	8.5	0.69	0.69	5.0	4.9	4.74	0.86
BERT-RW	42.3	34.3	17.9	0.77	0.75	22.0	7.5	8.19	1.05	28.0	22.6	8.3	0.72	0.70	12.6	6.9	5.83	1.05
GPT2-RW	44.7	38.7	22.0	0.78	0.77	**31.2**	7.5	9.15	1.17	38.5	31.9	18.0	0.76	0.75	17.0	**7.2**	7.66	1.06
MOBA-FuseGPT	**50.3**	**44.5**	**26.3**	**0.81**	**0.79**	30.3	7.5	**9.78**	**1.25**	**44.2**	36.8	**21.5**	**0.78**	**0.77**	12.8	6.8	**7.85**	**1.08**
Test Set	*Test Unseen*									*Test Unseen-ZS*								
Model	F1	RG	BLEU	EM-A	EM-X	D2	Ent	**Mean**	**Ratio**	F1	RG	BLEU	EM-A	EM-X	D2	Ent	**Mean**	**Ratio**
S2S	33.6	26.2	12.4	0.71	0.70	1.8	5.5	4.73	0.67	23.3	18.1	7.1	0.67	0.63	8.3	3.9	4.56	0.52
BERT	46.8	40.5	23.9	0.79	0.77	10.2	7.7	8.03	1.14	36.6	28.8	13.7	0.76	0.76	39.8	5.9	7.92	0.91
MOBA-GPT	48.1	43.2	27.3	0.80	**0.79**	10.5	7.6	8.37	1.19	45.6	40.0	23.2	**0.80**	**0.78**	33.0	5.8	9.07	1.04
MOBA-RC	34.8	29.7	15.5	0.73	0.72	12.5	**8.1**	7.03	1.00	38.2	30.8	21.6	0.77	0.74	**42.1**	**6.3**	8.72	1.00
S2S-RW	38.6	32.0	15.8	0.74	0.72	4.5	6.3	6.04	0.86	27.8	23.4	10.2	0.70	0.69	18.5	4.9	6.03	0.69
BERT-RW	40.0	32.4	15.9	0.76	0.74	8.3	7.8	6.91	0.98	28.5	23.7	8.5	0.72	0.69	34.7	**6.3**	6.72	0.77
GPT2-RW	44.5	38.8	23.4	0.78	0.76	**13.8**	8.0	8.27	1.18	37.6	31.6	17.9	0.76	0.75	40.5	**6.3**	8.46	0.97
MOBA-FuseGPT	**50.0**	**44.7**	**28.7**	**0.81**	**0.79**	11.8	7.7	**8.69**	**1.24**	**47.5**	**40.1**	**24.1**	**0.80**	**0.78**	35.2	6.2	**9.34**	**1.07**

Table 4: Automatic evaluation results. Besides the introduced metrics, **Mean** reports the geomean of all metrics. Four test sets have different data distributions, to make the comparison more simple, **Ratio** reports the ratio of a model's geomean score relative to the ***MOBA-RC***'s geomean score in each group.

ing the effectiveness of our approach. In data-to-text models, the naive S2S learns to generate commentaries from scratch, and thus it has the weakest performance in every group. After introducing the pre-trained language model, the advanced BERT/MOBA-GPT has significantly better performance. Moving to the rule-based *MOBA-RC*, we can find the results are quite interesting. The performance in the zero-shot test sets (*Seen/Unseen-ZS*) is more powerful than in the normal test sets (*Seen/Unseen*). For example, *MOBA-RC* is worse than BERT in normal test sets, but is significantly better than BERT in zero-shot test sets. It indicates the necessity of using rule-based methods in real scenarios because of the adaption ability to the new requirements. Based on the *MOBA-RC*, we also evaluated several re-writing models. We can find such re-writing models work well only if the data distribution is similar in both the training stage and the test stage. The proposed *MOBA-FuseGPT* combines the advantages of all previous methods, and thus undoubtedly has the best overall performance.

The performance of the rule-based *MOBA-RC* will not be affected by the test set, and we can regard it as a constant baseline. Intuitively, if the performance relative to *MOBA-RC* is greater than 1.0 (i.e. **Ratio**), we can say this model is better than the *MOBA-RC*. In four test groups, only our *MOBA-GPT* and *MOBA-FuseGPT* can satisfy this.

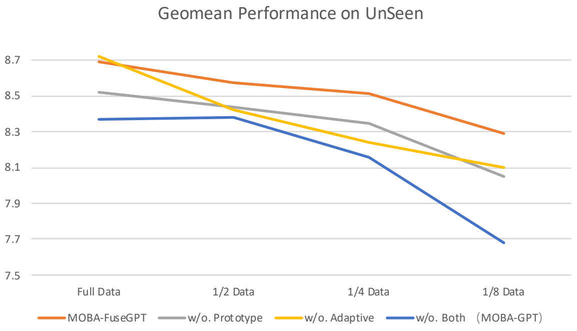


Figure 4: The geomean score in few-shot scenarios. The detail can be find in Appendix C

4.3 Ablation Study in Few-Shot Scenarios

Generative methods have great potential in developing machine commentators, but they also require supervised training data. To reduce the burden of collecting supervised data, besides using the pre-trained language model, *MOBA-FuseGPT* infuses the power of the rule-based method with the *Prototype-Augmented Generation* and the *Adaptive Learning*. To evaluate them, we tested ablated models in few-shot scenarios. As shown in Figure 7, both two techniques are effective. Even only using $\frac{1}{8}$ training data (about 533 instances), the geomean score of *MOBA-FuseGPT* still outperforms/on pair with the baselines trained on full data (see Table 4). It means our approach can be quickly migrated to other MOBA games.

Event 1	事件类型 *Event Type*: 英雄状态变化 Hero Alive; 状态 *State*: 复活 Resurrection 玩家 *Player*: 萧瑟 XiaoSe 英雄 *Hero*: 虚空假面 Faceless Void
MOBA-RC	虚空假面复活了。 Faceless Void has resurrected.
MOBA-GPT	萧瑟复活了，加油伐木啊虚空假面。 XiaoSe has resurrected. Come on, cutting down trees, Faceless Void.
MOBA-FuseGPT	虚空假面复活了，能否守住一波？ Faceless Void has resurrected, can he guard the team?
Human	虚空假面已经复活了。Faceless Void has already resurrected.
Event 2	事件类型 *Event Type*: 建筑破坏总结(夜魇) Team Kill List (Dire); 夜魇破坏的建筑数目 *The number of buildings destroyed by Dire*: 1 夜魇被破坏的建筑数目 *The number of Dire's destroyed buildings*: 3
MOBA-RC	天辉已经推掉夜魇3座建筑了。 Radiant has destroyed Dire's three buildings.
MOBA-GPT	夜魇已经被推掉3座塔了。 Dire's three buildings has been destroyed.
MOBA-FuseGPT	天辉已经推掉了夜魇3座建筑。Radiant has destroyed Dire's three buildings.
Human	夜魇已经失守了3个建筑，但却仅仅拿下了天辉的1个建筑,双方差距在持续扩大中。Dire has lost three buildings, but only destroyed one Radiant's building; the gap between the two teams continues to widen.

Table 5: Case Study. The first case is sampled from *Unseen*, and the second case is sampled from *Unseen-ZS*. It is worth noting that Dota2 has two teams (competitors), namely, Dire and Radiant.

4.4 Case Study

Two cases are shown in Table 5. In the first case, the commentary generated by *MOBA-RC* is correct but not attractive. *MOBA-MOBA* tried to make the commentary more attractive, but it generated some wired words. Undoubtedly, the commentary generated by *MOBA-FuseGPT* is not only correct but also more attractive than the human-generated commentary. The second case is sampled from the zero-shot *Unseen-ZS*. The situation of *MOBA-RC* is similar to the last case because it only requires pre-defined rules. Both two generative *MOBA-GPT* and *MOBA-FuseGPT* outputted acceptable commentaries, demonstrating the applicability in the zero-shot scenario. However, both are not as informative as human-generated commentary.

5 Related Work

The development of e-sports has greatly enriched people's amateur life and brought great commercial value. Subsequently, many AI-based tools/models have been developed for e-sports to meet the increasingly growing demands. The most popular applications include: 1) predicting the outcome/death (Yu et al., 2018; Akhmedov and Phan, 2021; Wang et al., 2018; Qi et al., 2018; Katona et al., 2019); 2) identifying the player/role (Yuen et al., 2020; Demediuk et al., 2019); 3) recommending items/heroes/characters (Looi et al., 2019; Porokhnenko et al., 2019; Aznin et al., 2019); 4) AI Players (OpenAI, 2019); and many others (Ponomarenko and Sirotkin, 2020; Marchenko and Suschevskiy, 2018; Block et al., 2018). Compared to such works, we study the machine commentary generation, which not only involves the knowledge of e-sports but also the NLP techniques.

Text generation is an important task in both the academic and industry (Li et al., 2021b). Generally, it learns to generate texts based on the given data (Li et al., 2021a), such as generating biology based on infobox (Bai et al., 2020), generating descriptions (Li et al., 2020), and many others. Only recently, there is a work to generate commentaries for traditional sports games (Ishigaki et al., 2021). However, no previous work has tried to generate MOBA commentary because e-sports games are pretty complex, and it is not easy to collect supervised training data. Compared to such works, this work focuses on a more specific task, namely, generating game commentaries.

6 Conclusion

This paper proposes constructing machine game commentators that can work at any time and place. This paper focuses on the MOBA games and proposes a novel data-driven commentary generation framework *MOBA-E2C*. Instead of using visual features, *MOBA-E2C* generates commentaries by using the meta-data of a game. *MOBA-E2C* regards each game session as a live-streaming sequence of game states, and then employs several event handlers to capture events. Subsequently, we use *MOBA-FuseGPT* to generate commentaries based on the identified events. It infuses the advantages of rule-based methods and the generative pre-trained language model. In the experiments, we take Dota2 as the case study and construct a dataset *Dota2-Commentary*. Extensive experiments have demonstrated the effectiveness of our approach.

Limitation and Future Work This work focuses on exploring the research field of MOBA game machine generators and setting the baseline; thus, we propose a data-driven framework *MOBA-E2C*, a generator *MOBA-FuseGPT*, and a dataset *Dota2-Commentary*. However, it can be easily found that this work did not try to use a new network architecture. In the future, based on the foundation of this work, we will continue to investigate the potential of MOBA game machine generator and brings more brilliant models.

Ethical Considerations

This work constructed a new dataset by employing two human annotators. Such two annotators are employees in a commercial company (GamesMind Technology), and thus they have paid with the corresponding salaries. The salary level is higher than the local minimum wage.

From the perspective of the technique, we have filtered out irrational rules and data; thus, the commentaries generated by our approach have no ethical issues. Meanwhile, compared to human commentators (streamers), our machine commentator can better avoid the ethical issue.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant No.62141208.

References

Kodirjon Akhmedov and Anh Huy Phan. 2021. Machine learning models for DOTA 2 outcomes prediction. *CoRR*, abs/2106.01782.

Mohammad Z. A. Z. Aznin, Norizan M. Diah, and Nur A. S. Abdullah. 2019. Expert system for dota 2 character selection using rule-based technique. In *IVIC*.

Yang Bai, Ziran Li, Ning Ding, Ying Shen, and Hai-Tao Zheng. 2020. Infobox-to-text generation with tree-like planning based attention network. In *IJCAI*.

Florian Block, Victoria J. Hodge, Stephen Hobson, Nick Sephton, and et.al. 2018. Narrative bytes: Data-driven content production in esports. In *TVX*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert.

Simon Demediuk, Peter York, Anders Drachen, and et.al. 2019. Role identification for accurate analysis in dota 2. In *AIIDE*.

Tatsuya Ishigaki, Goran Topic, Yumi Hamazono, Hiroshi Noji, Ichiro Kobayashi, Yusuke Miyao, and Hiroya Takamura. 2021. Generating racing game commentary from vision, language, and structured data. In *INLG*.

Adam Katona, Ryan J. Spick, Victoria J. Hodge, Simon Demediuk, and et.al. 2019. Time to die: Death prediction in dota 2 using deep learning. In *IEEE CoG*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021a. Few-shot knowledge graph-to-text generation with pretrained language models. In *ACL*, pages 1558–1568, Online. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021b. Pretrained language models for text generation: A survey. *CoRR*, abs/2105.10311.

Ziran Li, Zibo Lin, Ning Ding, Hai-Tao Zheng, and Ying Shen. 2020. Triple-to-text generation with an anchor-to-prototype framework. In *IJCAI*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, and et.al. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation.

Wenli Looi, Manmeet Dhaliwal, Reda Alhajj, and Jon G. Rokne. 2019. Recommender system for items in dota 2. *IEEE Trans. Games*.

Ekaterina Marchenko and Vsevolod Suschevskiy. 2018. Analysis of players transfers in esports. the case of dota 2. In *Proceedings of the 22nd International Academic Mindtrek Conference*. ACM.

OpenAI. 2019. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680.

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

Alexander A. Ponomarenko and Dmitry V. Sirotkin. 2020. Dota underlords game is np-complete. *CoRR*, abs/2007.05020.

Iuliia Porokhnenko, Petr Polezhaev, and Alexander Shukhman. 2019. Machine learning approaches to choose heroes in dota 2. In *FRUCT*.

Zhen Qi, Xiangbo Shu, and Jinhui Tang. 2018. Dotanet: Two-stream match-recurrent neural networks for predicting social game result. In *BigMM*. IEEE.

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

Ilya Sutskever, Oriol V, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Nanzhi Wang, Lin Li, Linlong Xiao, Guocai Yang, and Yue Zhou. 2018. Outcome prediction of DOTA2 using machine learning methods. In *ICMAI*. ACM.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *AAAI*, pages 7281–7288.

Zelong Yang, Yan Wang, Piji Li, Shaobin Lin, Shuming Shi, Shao-Lun Huang, and Wei Bi. 2022. Predicting events in moba games: Prediction, attribution, and evaluation. *IEEE Transactions on Games*, pages 1–1.

Lijun Yu, Dawei Zhang, Xiangqun Chen, and Xing Xie. 2018. Moba-slice: A time slice based evaluation framework of relative advantage between teams in MOBA games. In *Computer Games - 7th Workshop, CGW*.

Sizhe Yuen, John D. Thomson, and Oliver Don. 2020. Automatic player identification in dota 2. *CoRR*, abs/2008.12401.

Houyu Zhang, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2020. Grounded conversation generation as guided traverses in commonsense knowledge graphs. In *ACL*, pages 2031–2043.

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 Sun, Zhiyuan Liu, Minlie Huang, Wentao Han, Jie Tang, Juanzi Li, Xiaoyan Zhu, and Maosong Sun. 2021. Cpm: A large-scale generative chinese pre-trained language model. *AI Open*, 2:93–99.

A Event Handlers

This work selects Dota2 as the case to evaluate the proposed *MOBA-E2C*. Thus, the first job is to design a set of event handlers to capture highlight event tables for Dota2.

As listed in Table 6, we have designed 34 event handlers for capturing different highlight events for Dota2, where 8 of them are regarded as the zero-shot event types in the dataset partition.

B Annotation Tool

As shown in Figure 5, we have designed a visual tool for the annotation.

It is worth noting that volunteers are required to generate a commentary based on the given event table and the corresponding game replay video at the same time.

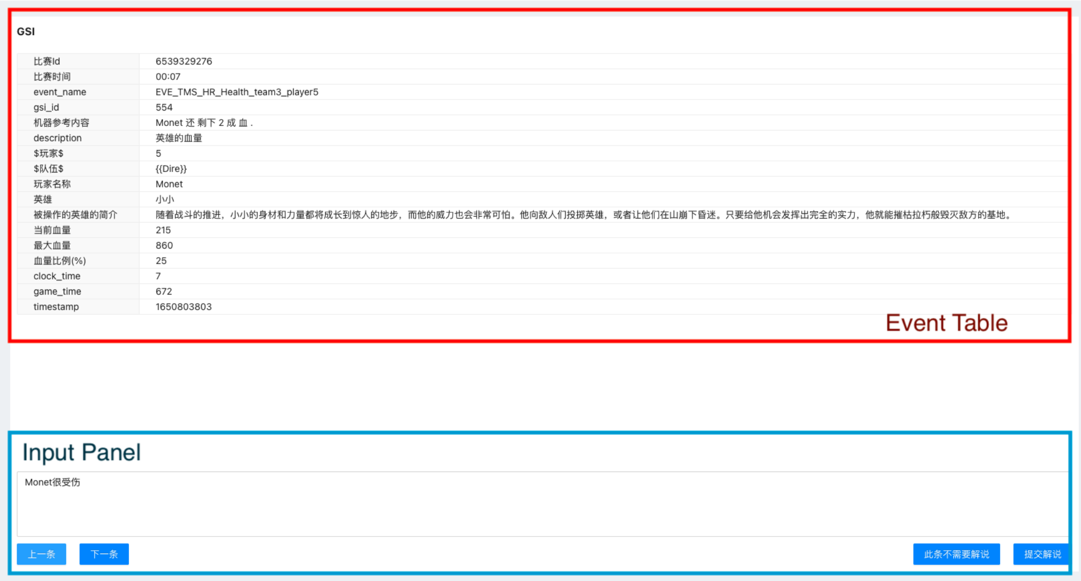
To ensure the quality of human commentaries, the tool will automatically check the submitted commentary. If a submission is too similar to the existing commentaries, this submission will be rejected and the corresponding volunteer is required to make a new commentary.

C Ablation Study

Although generative methods have more potential in developing machine commentary generation, they also require supervised training data. Hence, *MOBA-FuseGPT* infuses the power of rule-based method with the *Prototype-Augmented Generation* and the *Adaptive Learning*. We test ablated models in few-shot scenarios. As illustrated in Table 7, both two techniques are effective. Meanwhile, even only using $\frac{1}{8}$ training data (about 533 instances), the geomean score of *MOBA-FuseGPT* still outperforms/on pair with baselines. It means our approach can be quickly migrated to other MOBA games.

ID	ZS	Event Type	Handler Type	Description
1	N	*Time Now*	*State-Change*	Report the current game time.
2	N	*Winner*	*State-Change*	Report the winner of the game.
3	N	*Daytime Change*	*State-Change*	Report the change of game daytime (i.e., day or night).
4	Y	*Game Pause*	*State-Change*	Notify when the game is paused or resumed.
5	N	*BanPick*	*State-Change*	Report the banned/picked hero.
6	N	*Hero Alive*	*State-Change*	Notify when a hero has died or resurrected.
7	N	*Hero Level Up*	*State-Change*	Notify when a hero has upgraded.
8	N	*Hero Talent*	*State-Change*	Notify when a hero has selected a talent.
9	N	*Item Business*	*State-Change*	Notify when an item has been sold/bought by a hero/player.
10	N	*Item Usage*	*State-Change*	Notify when an item has been used by a hero/player.
11	N	*Ability Usage*	*State-Change*	Notify when an ability has been used by a hero/player.
12	N	*Ability Level Up*	*State-Change*	Notify when a hero's ability has upgraded.
13	N	*Last Hits*	*Counting*	Count the number of last hits.
14	Y	*Denies*	*Counting*	Count the number of denies.
15	Y	*Kills*	*Counting*	Count the number of kills.
16	N	*Deaths*	*Counting*	Count the number of deaths.
17	Y	*Assists*	*Counting*	Count the number of assists.
18	Y	*Kill Streaks*	*Counting*	Count the number of kill streaks.
19	Y	*Campus Stacked*	*Counting*	Count the number of campus stacked.
20	N	*Roshan Generation*	*Counting*	Track the generation of Roshan.
21	N	*Building Health*	*Tracking*	Track the health of a building.
22	N	*Roshan State*	*Tracking*	Track the state of Roshan.
23	N	*Player's Money*	*Tracking*	Track the money of a player and when reaching the milestone.
24	Y	*Player's Net Worth*	*Tracking*	Track the net worth of a player.
25	N	*Hero's Health*	*Tracking*	Track the health of a hero.
26	N	*Hero's Resurrection*	*Tracking*	Track the progress of a hero's resurrection.
27	N	*Activated Runes*	*Tracking*	Track the number of activated runes.
28	N	*Kill List*	*Summary*	Summarize the kill list by a hero/hero.
29	N	*Destroyed Buildings*	*Summary*	Summarize the destroyed buildings for Dire/Radiant.
30	N	*Team Destroyed Buildings*	*Summary*	Compare the destroyed buildings between Dire and Radiant.
31	N	*Team Net Worth*	*Summary*	Compare the the total net worth between Dire and Radiant.
32	N	*Team Kill List (Radiant)*	*Summary*	Compare the killed heroes between Dire and Radiant (team A=Radiant).
33	Y	*Team Kill List (Dire)*	*Summary*	Compare the killed heroes between Dire and Radiant (team A=Dire).
34	N	*Smoked Heroes*	*Summary*	Summarize the heroes that just smoked.

Table 6: List of Dota2 Event Handlers. **ZS** denotes the zero-shot, and Y/N means Yes/No.



比赛Id	6539329276
比赛时间	00:07
event_name	EVE_TMS_HR_Health_team3_player5
gsi_id	554
机器参考内容	Monet 还 剩下 2 成 血 .
description	英雄的血量
$玩家$	5
$队伍$	{{Dire}}
玩家名称	Monet
英雄	小小
被操作的英雄的简介	随着战斗的推进，小小的身材和力量都将成长到惊人的地步，而他的威力也会非常可怕。他向敌人们投掷英雄，或者让他们在山崩下昏迷。只要给他机会发挥出完全的实力，他就能摧枯拉朽般毁灭敌方的基地。
当前血量	215
最大血量	860
血量比例(%)	25
clock_time	7
game_time	672
timestamp	1650803803

Monet很受伤

Figure 5: The adopted visual annotation tool.

	Full Training Data								*1/2 Training Data*							
Model	**Mean**	F1	RG	BLEU	EM-A	EM-X	D2	Ent	**Mean**	F1	RG	BLEU	EM-A	EM-X	D2	Ent
MOBA-FuseGPT	8.69	50.03	44.73	28.65	0.81	0.79	11.76	7.72	8.57	49.29	43.97	27.73	0.81	0.79	11.48	7.71
w/o. Prototype	8.52	50.25	44.39	29.06	0.81	0.79	10.38	7.56	8.44	48.84	43.40	27.44	0.80	0.79	10.84	7.69
w/o. Adaptive	8.72	49.41	44.54	27.15	0.81	0.79	12.84	7.89	8.42	47.37	42.84	25.66	0.80	0.79	11.85	7.72
w/o. Both (MOBA-GPT)	8.37	48.05	43.16	27.32	0.80	0.79	10.50	7.64	8.38	47.95	42.82	26.88	0.80	0.79	10.82	7.64
	1/4 Training Data								*1/8 Training Data*							
Model	**Mean**	F1	RG	BLEU	EM-A	EM-X	D2	Ent	**Mean**	F1	RG	BLEU	EM-A	EM-X	D2	Ent
MOBA-FuseGPT	8.51	48.40	43.42	26.96	0.81	0.79	11.65	7.68	8.29	45.90	40.75	24.73	0.80	0.78	11.92	7.89
w/o. Prototype	8.35	48.42	43.40	27.12	0.81	0.79	10.24	7.57	8.05	45.35	40.07	23.83	0.80	0.77	10.62	7.72
w/o. Adaptive	8.24	45.26	41.04	23.96	0.80	0.78	12.08	7.70	8.10	43.02	38.17	21.63	0.78	0.76	13.61	7.92
w/o. Both (MOBA-GPT)	8.16	46.80	41.55	26.10	0.80	0.78	10.02	7.59	7.68	41.39	36.96	20.82	0.78	0.76	10.95	7.66

Table 7: Experiments on Test Unseen