TA-Prompting: Enhancing Video Large Language Models for Dense Video Captioning via Temporal Anchors

Supplementary Material

7. Detailed Evaluation Explanation

We mainly compare our model with VideoChat [24], VideoChatGPT [32], TimeChat [38], Momentor [36], LITA [15], VTimeLLM [14], and VTG-LLM [12]. For VideoChat and VideoChatGPT, we directly use the scores reported in [14]. For TimeChat, we leveraged the official checkpoint to evaluate its performance on both Moment Retrieval tasks. Momentor's metric scores are also directly reported from their paper, where the model was evaluated in a zero-shot setting. For LITA, we generated results across all four datasets. Since ActivityNet-Caption was one of the domains LITA was originally trained on, we used its official checkpoint to conduct experiments on ActivityNet-Caption [19] for both DVC and MR tasks. For YouCook2, we fine-tuned LITA using LoRA to adapt it to the domain. Similarly, since VTimeLLM was not pretrained on the YouCook2 dataset, we also finetuned it using LoRA to ensure better alignment with the dataset characteristics. For other models not explicitly specified above, we report their metric scores directly as provided in their respective papers.

8. Additional Details on Model Architecture and Training

8.1. Details of Time Embedding Module

We follow previous works [16, 21, 33], applying sinusoidal positional encoding to map each 2-dimensional temporal anchor into a 4096-dimensional representation, matching the hidden size of Vicuna v1.5 [60]. A two-layer MLP is then used to reduce the dimension to 512 and subsequently project it back to 4096, aligning with the LLM's feature space. For reconstructing the event timestamp $\tilde{Y}$ from the LLM output, we reuse this same two-layer MLP, followed by a linear layer with output dimension 2, to predict the center and duration (c_i, d_i) of each event.

8.2. Temporal-Aware Video Event Captioning learning template

For each event, we use the following format for captioning: "<sep> $< s >$ $< s >$ 'i-th event caption'.". The special token <sep> marks the start of each event generation. For simplicity, we use "*", a string not utilized during training, to represent <sep> in our training template. The first <s> token indicates the position to be replaced by the encoded temporal anchor, while the second <s> marks the beginning of each event caption. Through this training template, *TA-Prompting* effectively utilizes temporal anchors to generate accurate and contextually relevant event captions.

During self-attention within the LLM, we adjust the attention maps of the tokens corresponding to the temporal anchors such that they only attend to the video tokens. This design ensures that the temporal anchor focuses solely on the video context, as its primary function is to denoise and extract the relevant event timestamps from the video. By restricting its attention scope, TA-Prompting effectively refines the localization of events, ensuring that each generated caption is temporally aligned with the corresponding video segment.

8.3. TA-Prompting's pre-training process

Before fine-tuning *TA-Prompting* on task specific objectives such as Dense Video Captioning and TemporalQA, we first pre-trained our LLM backbone (Vicuna v1.5-7B) with LoRA using Boundary Perception training data from VTimeLLM [14], which is processed from InternVid [51]-10M-FLT. This pre-training step equips the LLM with an initial temporal awareness capability. To achieve this, we reformat the annotations to Temporal-Aware Video Event Captioning learning template, as depicted in Sec 8.2, and train the LLM with LoRA for two epochs following this procedure: In the first epoch, we train the model using only textual supervision, leaving the first <s> tokens unchanged rather than replacing them with encoded temporal anchors. This allows the LLM to first learn the structure and linguistic patterns of event descriptions. In the second epoch, we apply the full training procedure, incorporating encoded temporal anchors to align captions with their corresponding temporal contexts. The LoRA configuration and training hyper-parameters remain identical to those detailed in Sec 4.3.

9. The effectiveness of Temporal-Aware Video Event Captioning training

In our main paper, we compare *TA-Prompting* against various VideoLLMs, most of which do not include event localizer modules. To verify the effectiveness of our joint optimization, we combine VTimeLLM [14] with event localizer from PDVC [46]. During inference, we iteratively sample a timestamp from the generated proposals as input to VTimeLLM, where the selection probability of each proposal is weighted by its proposal score generated by PDVC. This process continues, with the chosen proposal guiding VTimeLLM in generating event descriptions, until the LLM

terminates generation or all proposals are exhausted. The result is presented in Table 6, which indicates that the combination of event localizer and VideoLLM doesn't necessarily lead to performance improvement.

Method	CIDEr	METEOR	SODA_c
VTimeLLM-7B [14]	20.0	6.0	4.8
TA-Prompting-7B (Ours)	**29.2**	**7.0**	**6.1**

Table 6. Comparison of TA-Prompting and the combination of VTimeLLM with an event localizer on the ActivityNet Captions. [19]

10. Details of Event Coherent Sampling

The Event Coherent Sampling algorithm, as detailed in Section 3.3, aims to select the optimal temporal anchor for event caption generation. However, this original iterative process can be time-consuming, as it involves repeatedly evaluating similar temporal anchors, often leading to overlapping and repetitive captions that describe the same events.

To mitigate this in practical scenarios, we arrange the temporal anchors chronologically and group similar temporal anchors into clusters, forming a set of temporal anchor clusters. Each cluster represents a set of similar events, allowing us to process these anchors simultaneously. By expanding the batch size to include all anchors within a cluster, we generate captions in parallel, reducing redundant evaluations and improving efficiency. After processing a cluster, we proceed to the next one sequentially. We extend the batch size to ensure that each temporal anchor in the current cluster interacts with all previously generated event captions.

Although the decoding strategy depicted above reduces redundant computation, it introduces new challenges, such as the exponentially growing batch size and the complexity of selecting the final answer from multiple candidates. To address these issues, we employ the scoring function $\mathcal{S}$ defined in Eq. 6. If the batch size exceeds the certain batch size threshold during the expansion, we apply $\mathcal{S}$ to filter out captions with lower scores, thereby maintaining the batch size within the specified limit. A high score of $\mathcal{S}$ indicates that the generated caption aligns with the visual content and maintains coherence across consecutive events. By applying the modified Event Coherent Sampling with the scoring function $\mathcal{S}$ during the inference stage, we are able to produce captions that are coherent across events and visually faithful, while avoiding excessive computational costs, repeated descriptions, and contextually inconsistent outputs. The algorithm 2 outline the overall process of Event Coherent Sampling approach.

Algorithm 2 Event Coherent Sampling

Model: Event Localizer $\theta_{\mathcal{E}}$
Input: video v

Extract Temporal Anchors $\hat{Y}$ from video v using Event Localizer $\theta_{\mathcal{E}}$
Sort the Temporal Anchors in order and group similar temporal anchors $\hat{Y}$ to form a set $\mathbf{X}$ of temporal anchor clusters.
output = $\{\}$
while (Temporal Anchor cluster set $\mathbf{X}$ not $\emptyset$) **do**
  Generate an event caption C_i conditioned on each temporal anchor $(\hat{c}_i, \hat{d}_i)$ in the extracted temporal anchor cluster from $\mathbf{X}$ using a pop-left operation.
  calculate $\mathcal{S}_i$ for each event caption and select index j that has highest score $\mathcal{S}_j$
  $\hat{Y} \leftarrow \hat{Y} - (\hat{c}_j, \hat{d}_j)$
  output $\leftarrow$ output $+ (t_j^s, t_j^e, C_j)$
  if EOS in output **then**
    break
  end if
end while
Output: Final sequence $\{(t_i^s, t_i^e, C_i)\}_{i=1}^{N}$ containing coherent event captions.

Method	sec. per sample	ActivityNet-Caption		
		SODA_c	CIDEr	METEOR
TA-Prompting with **ECS**	7.08	**6.1**	**29.2**	**7.0**
VTimeLLM with **BS**	6.98	5.8	27.6	6.8

Table 7. Comparison of inference time (second per sample) between standard beam search (BS) and proposed event coherence sampling (ECS) on ActivityNet Caption dataset [19].

To evaluate our inference time, we compare *TA-Prompting* using Event Coherent Sampling (ECS) with VTimeLLM [14] using standard beam search (BS). Standard beam search is chosen as the comparison baseline because both ECS and BS involve the process of expanding a single input caption into multiple output sequences during inference. The comparison results are presented in Table 7, with the beam size for beam search set to 3. For this experiment, we use a single A100 GPU for each model and Python's built-in time package to measure inference time. The results show that our ECS achieves better performance while maintaining a similar computation time compared to beam search.

11. Specifying the inputs to the event localizer

Following LM4VisualEncoding [35], we use the LLM's output video features as visual inputs (refer to Figure 2) to the Event Localizer, demonstrating that LLMs can focus on more informative visual tokens. During inference, video

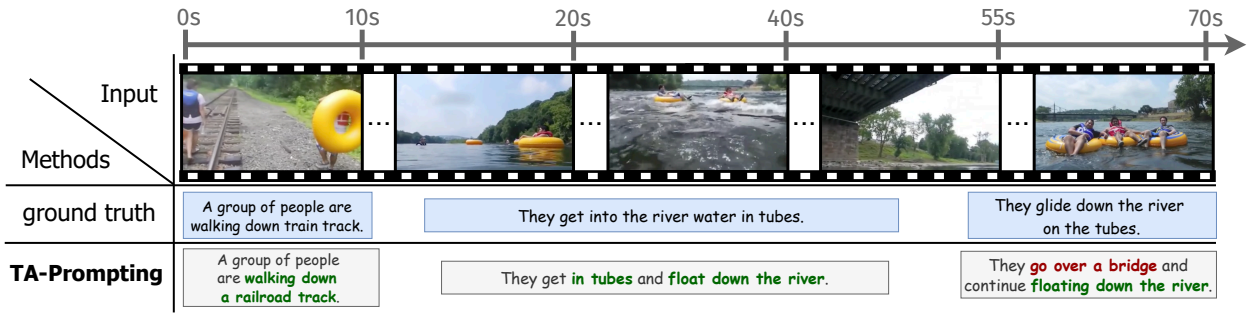

Figure 5. Visualization of a failure case with qualitative comparison of predicted timestamps and captions. The captions in green indicate correct alignment with the ground truth ones, while the captions in red indicate descriptions that are irrelevant to the visual content.

features are processed by the LLM, and the resulting features are fed into the Event Localizer to generate temporal anchors. These anchors are then sequentially provided to the LLM for event caption generation. Since video features pass through the LLM only once, the inference efficiency of our model is comparable to that of general VideoLLMs.

12. Choice of Timestamp Prediction During Inference

In our implementation, we ultimately use the temporal anchors $\hat{Y}$ generated by the Event Localizer as the predicted timestamps during inference, rather than the denoised $\tilde{Y}$. This decision is supported by our experimental results, where we observe comparable performance using $\tilde{Y}$ (SODA_c **6.1**, CIDEr **29.3**, METEOR 6.9) and temporal anchors $\hat{Y}$ (SODA_c **6.1**, CIDEr 29.2, METEOR **7.0**). The similarity in performance can be attributed to the effectiveness of the Event Confidence Scorer (ECS), which helps filter out noisy temporal anchors during the decoding process. As a result, both options are reliable for our final timestamp prediction.

13. Possibility of generalization to other datasets

To explore the generalization ability of TA-Promping beyond its training data (ActivityNet-Caption [19] and YouCook2 [62]), we randomly selected a video from YouTube and present result in Figure 6. Although there is no ground truth in this sample, the visualization shows that TA-Prompting has the potential to generalize to out-of-domain tasks.

14. Limitation and Failure Cases

In our work, we represent video content by extracting CLIP features from uniformly sampled frames and concatenating them into a single visual feature sequence. These visual features could potentially be improved by leveraging recent

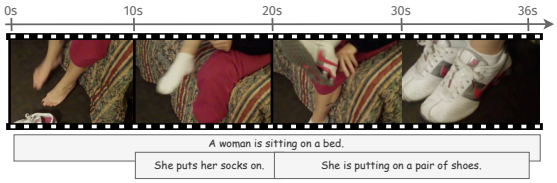

Figure 6. Visualization of dense captions on an out-of-domain YouTube video generated by TA-Prompting. Based on our observation, the video shows a woman sitting on a bed, sequentially putting on socks and then shoes. The visualization demonstrates that TA-Prompting accurately captures the key events occurring in the video.

VLLM methods developed for long video understanding. Another limitation of our approach is the risk of hallucinated captions, which may result from the choice of LLM backbone. As illustrated in Figure 5, *TA-Prompting* can sometimes generate descriptions that are unrelated to the actual video content.

15. More Quantitative Results

We present additional visualization results to provide a clearer comparison of our model's performance. For the dense video captioning task, we further show the visualization result on the YouCook2 [62] dataset compared with VTimeLLM [14], as shown in Figure 7. For the moment retrieval task, we compare our results against VTimeLLM [14] and LITA [15], both of which have demonstrated significant performance among all the VideoLLM models used for our comparisons in Table 1. Figure 9, illustrate the results on AcivityNet-Caption [2], while Figure 10 shows the results on Charades-STA [11] dataset. Figure 8 shows the results of ActivityNet-RTL [15].

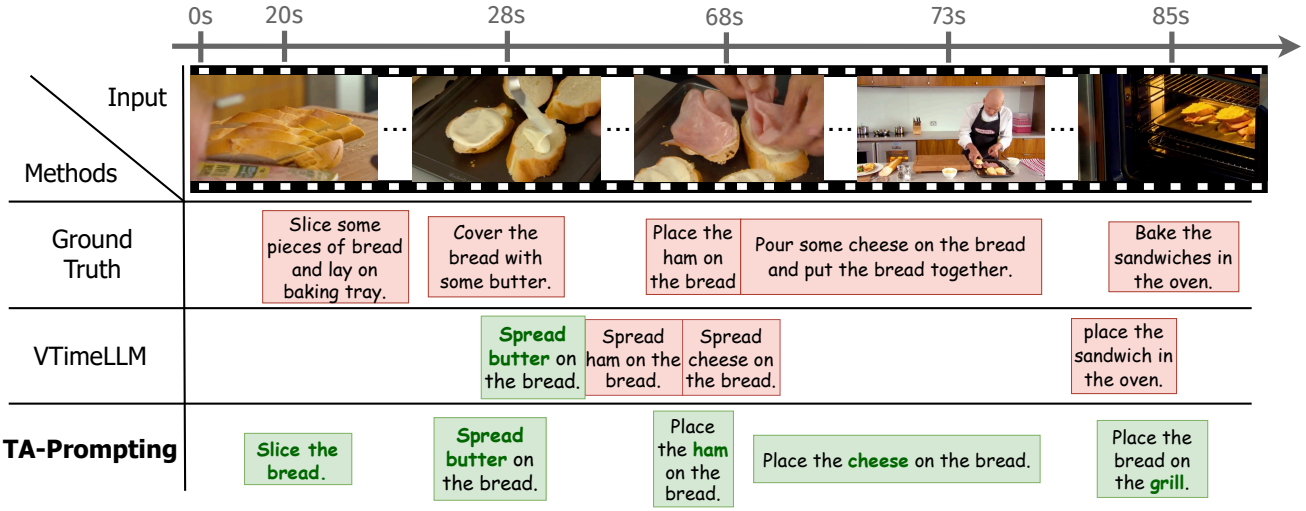

Figure 7. Qualitative results of the dense video captioning task on the Youcook2 dataset [62]. We compare TA-Prompting with VTimeLLM [14]. The width of each stripe indicates the predicted duration by each model.

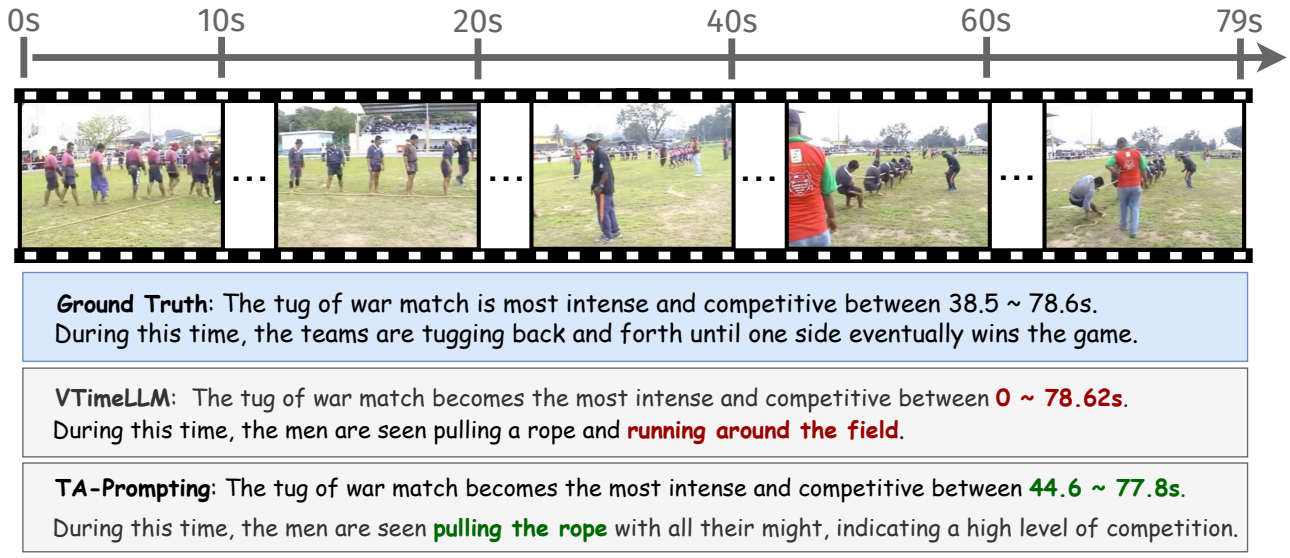

Figure 8. Qualitative results on ActivityNet-RTL [15]. The captions in green indicate correct alignment with the ground truth ones, while the captions in red indicate descriptions that are irrelevant to the visual content.

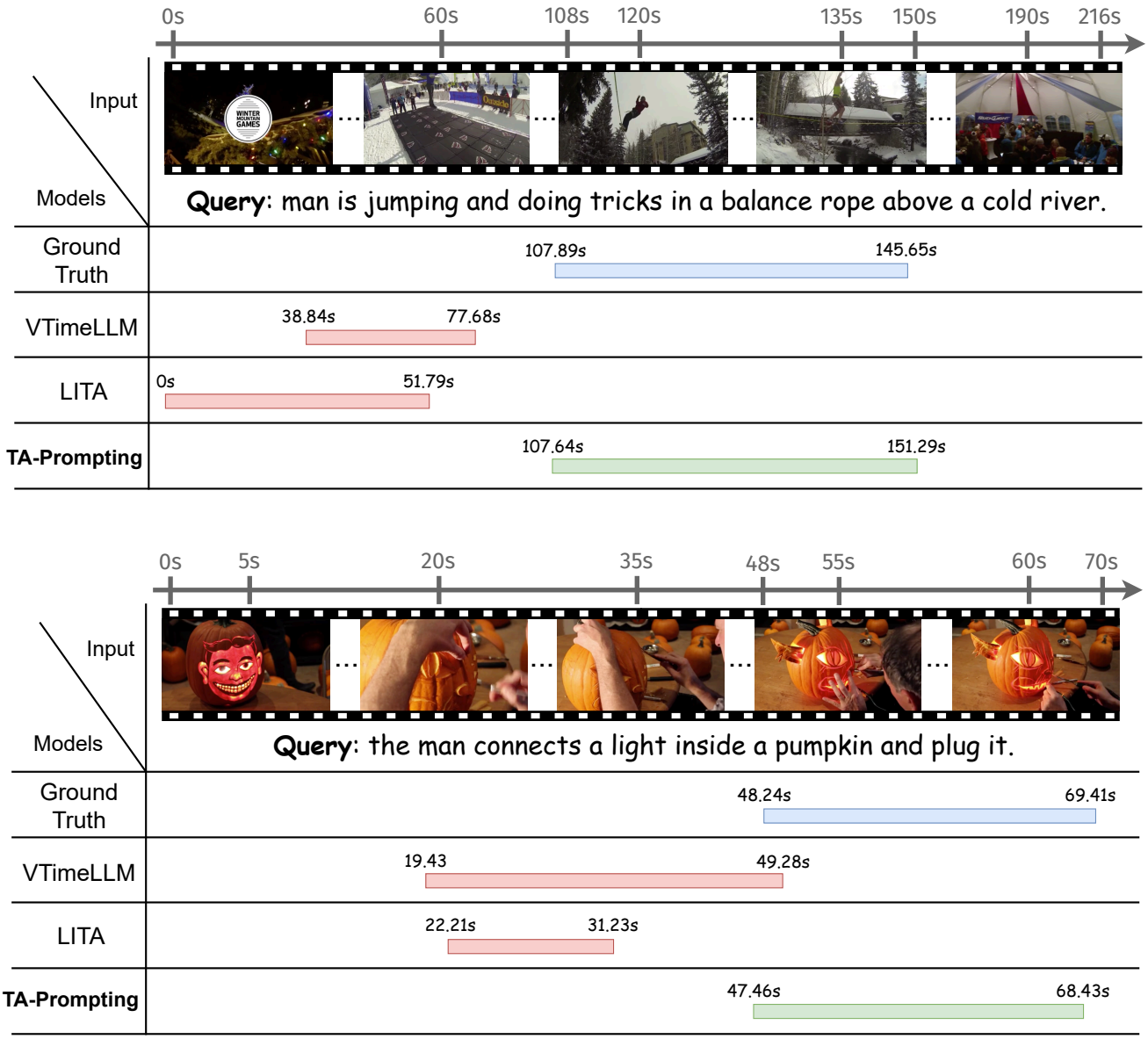

Figure 9. Qualitative results of the moment retrieval task on the ActivityNet-Caption dataset [2]. We compare TA-Prompting with two previous state-of-the-art models: LITA [15] and VTimeLLM [14]. The width of each stripe indicates the predicted duration by each model, with the start and end times labeled at the beginning and end of the stripe, respectively.

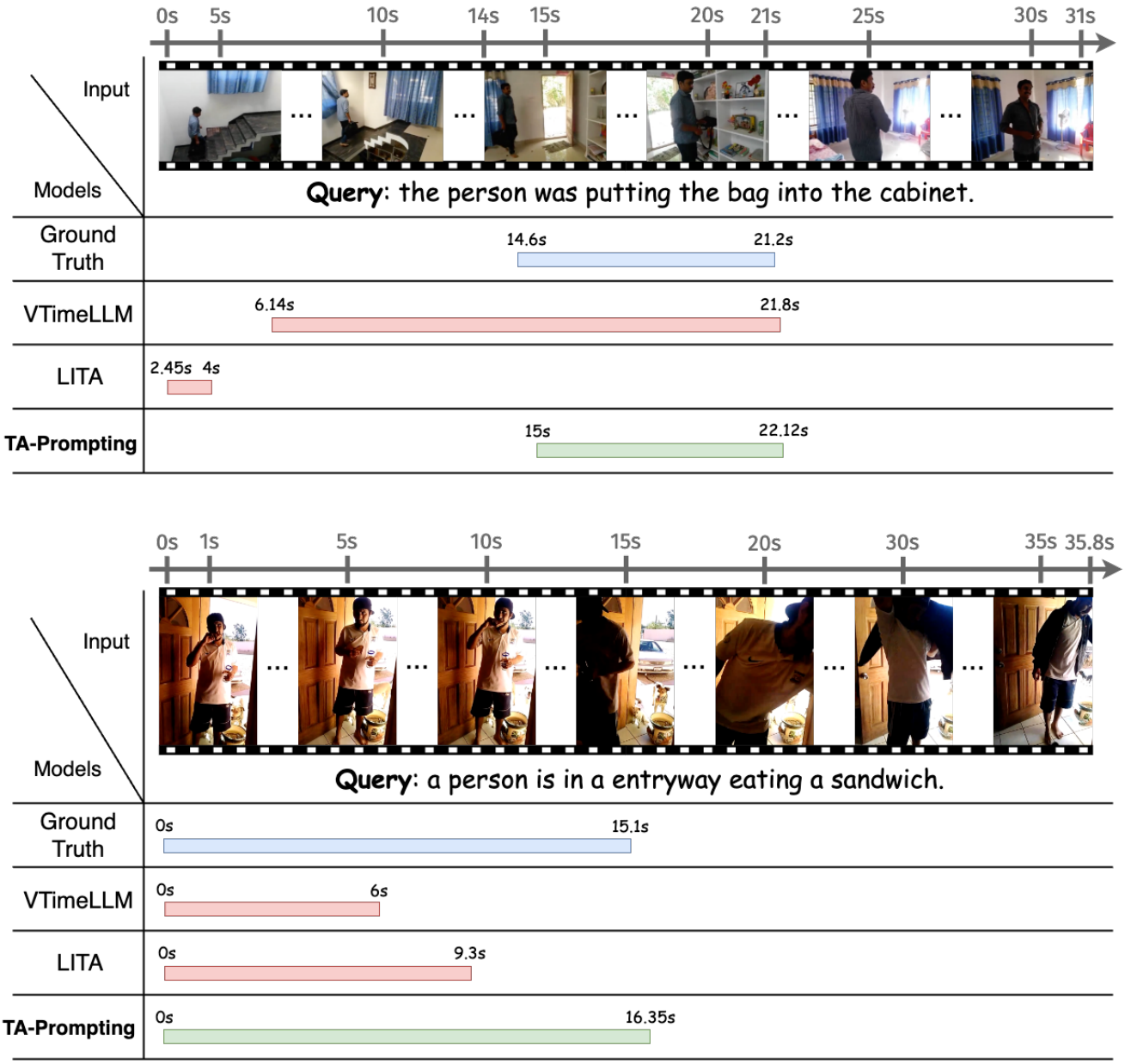

Figure 10. Comparison of TA-Prompting with two previous state-of-the-art models, LITA [15] and VTimeLLM [14], on the moment retrieval task using the Charades-STA dataset [11]. Each stripe represents the predicted segment by the respective models, with the beginning and end points clearly marked to indicate the retrieved duration.