STANFORD VISION & LEARNING

TOYOTA RESEARCH INSTITUTE

ICML 2025

Understanding Complexity in VideoQA via Visual Program Generation

Cristobal Eyzaguirre, Igor Vasiljevic, Achal Dave, Jiajun Wu,
Rares Andrei Ambrus, Thomas Kollar, Juan Carlos Niebles, Pavel Tokmakov

Questions can be of varying complexity

Is there a piano in the beginning?

Who was the first to play the piano in the video?

What was the teacher doing while he was playing the piano for the first time

Estimating Question Complexity

Is there a piano in the beginning?

Who was the first to play the piano in the video?

What was the teacher doing while he was playing the piano for the first time?

Estimating Question Complexity

Is there a piano in the beginning?

```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    first_frame = ImagePatch(video_segment, 0)
    piano_detected = first_frame.detect_object('piano')
    info = {
        'Piano detected in the first frame': piano_detected
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```

Who was the first to play the piano in the video?

```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    frame_of_interest = None
    for (i, frame) in enumerate(video_segment.frame_iterator()):
        if frame.detect_object('student') and \
            frame.simple_qa('is the student playing the piano?') == 'yes':
            frame_of_interest = frame
            break
    frame_of_interest_description = frame_of_interest.caption()
    info = {
        'Description of frame when piano is played for the first time': \
            frame_of_interest_description
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```

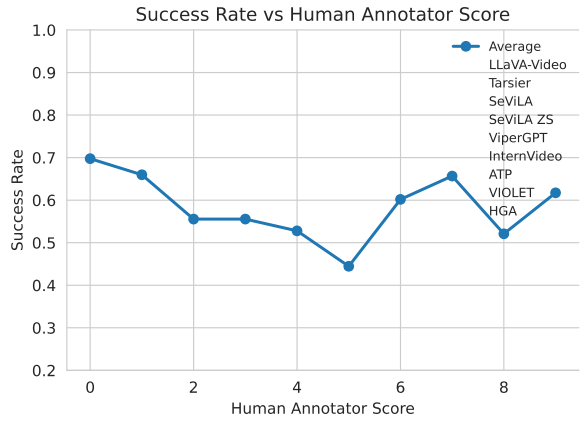
What was the teacher doing while he was playing the piano for the first time?

```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    student_playing = False
    frame_of_interest = None
    for (i, frame) in enumerate(video_segment.frame_iterator()):
        if frame.detect_object('student') and \
            frame.simple_qa('is the student playing the piano?') == 'yes':
            student_playing = True
        elif student_playing and frame.detect_object('teacher'):
            frame_of_interest = frame
            break
    description = frame_of_interest.caption()
    info = {
        'Description of frame while student plays the piano': \
            description,
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```

Estimating Question Complexity

Is there a piano in the beginning?

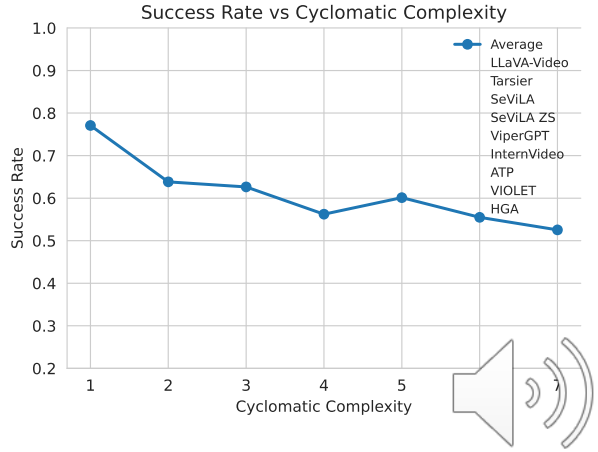
Complexity



Is there a piano in the beginning?

```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    first_frame = ImagePatch(video_segment, 0)
    piano_detected = first_frame.detect_object('piano')
    info = {
        'Piano detected in the first frame': piano_detected
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```

Complexity



CodePlexity

Query: What was the teacher doing while he was playing the piano for the first time?

CodePlexity

Query: What was the teacher doing while he was playing the piano for the first time?

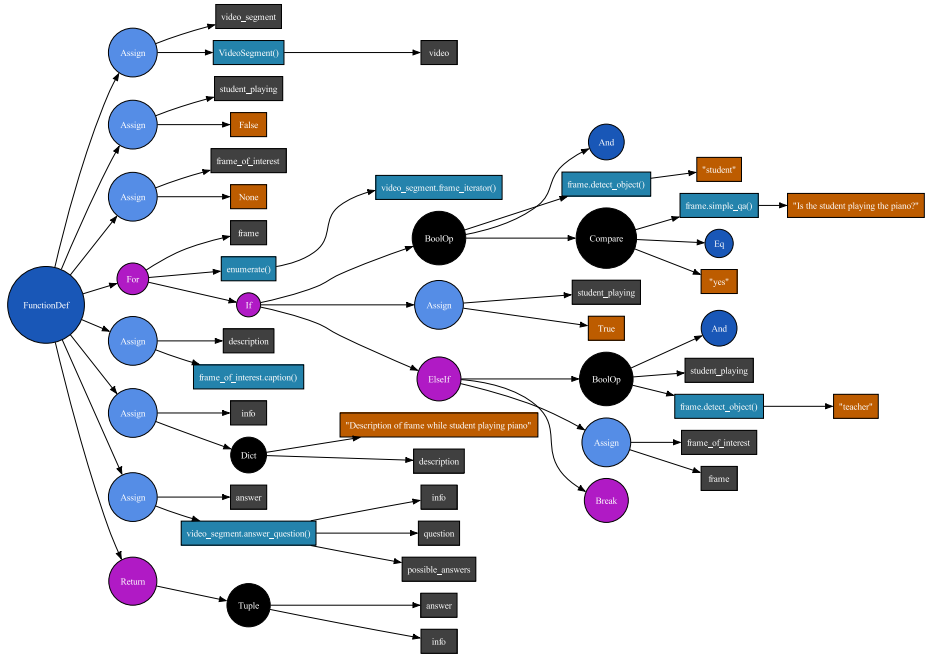
```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    student_playing = False
    frame_of_interest = None
    for frame in enumerate(video_segment.frame_iterator()):
        if frame.detect_object("student") and \
            frame.simple_qa("Is the student playing the piano?") == "yes":
            student_playing = True
        elif student_playing and frame.detect_object("teacher"):
            frame_of_interest = frame
            break
    description = frame_of_interest.caption()
    info = {
      "Description of frame while student playing piano": description
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```

Code
Generation

CodePlexity

Query: What was the teacher doing while he was playing the piano for the first time?

```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    student_playing = False
    frame_of_interest = None
    for frame in enumerate(video_segment.frame_iterator()):
        if frame.detect_object("student") and \
            frame.simple_qa("Is the student playing the piano?") == "yes":
            student_playing = True
        elif student_playing and frame.detect_object("teacher"):
            frame_of_interest = frame
            break
    description = frame_of_interest.caption()
    info = {
      "Description of frame while student playing piano": description
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```



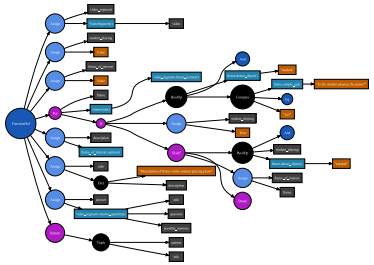
Code Generation

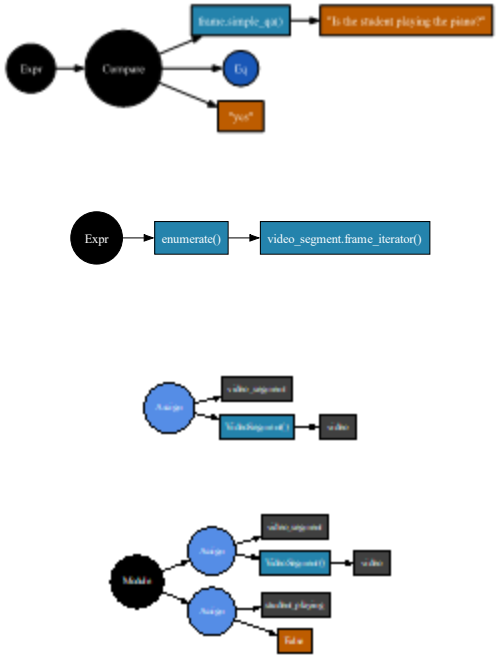
AST Generation

CodePlexity

Query: What was the teacher doing while he was playing the piano for the first time?

```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    student_playing = False
    frame_of_interest = None
    for frame in enumerate(video_segment.frame_iterator()):
        if frame.detect_object("student") and \
            frame.simple_qa("Is the student playing the piano?") == "yes":
            student_playing = True
        elif student_playing and frame.detect_object("teacher"):
            frame_of_interest = frame
            break
    description = frame_of_interest.caption()
    info = {
      "Description of frame while student playing piano": description
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```





1
0
1
1

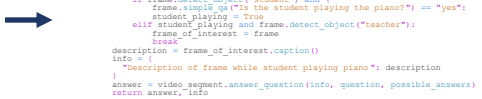
Code Generation

AST Generation

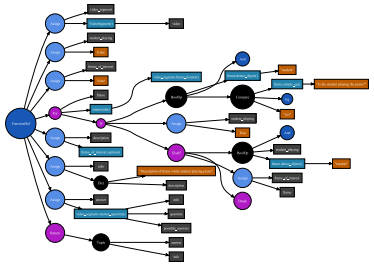
Subtree Encoding

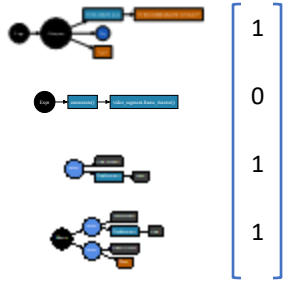
CodePlexity

Query: What was the teacher doing while he was playing the piano for the first time?



```
def execute_command(video, possible_answers, question):
    video_segment = VideoSegment(video)
    student_playing = False
    frame_of_interest = None
    for frame in enumerate(video_segment.frame_iterator()):
        if frame.detect_object("student") and \
            frame.simple_qa("Is the student playing the piano?") == "yes":
            student_playing = True
        elif student_playing and frame.detect_object("teacher"):
            frame_of_interest = frame
            break
    description = frame_of_interest.caption()
    info = {
      "Description of frame while student playing piano": description
    }
    answer = video_segment.answer_question(info, question, possible_answers)
    return answer, info
```





1
0
1
1

CodePlexity

Code Generation

AST Generation

Subtree Encoding

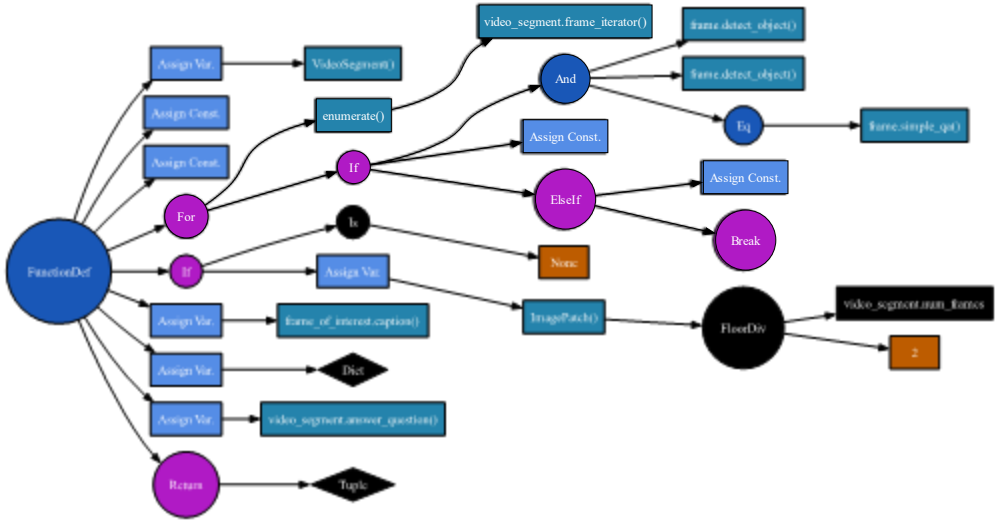
Complexity Estimation

CodePlexity

	Train Models				Val. Models				
	SeViLA	ViperGPT	ATP	VIOLET	HGA	SeViLA ZS	InternVideo	Tarsier	LLaVa-Video
Dependency Tree Depth	12.9	7.9	11.1	15.9	7.4	13.5	17.7	10.1	6.9
GPT-4 (OpenAI, 2023b)	9.6	8.9	11.6	5.8	7.8	14.6	13.9	10.8	5.2
BERT (Devlin et al., 2018)	12.5	6.0	18.3	17.3	7.7	14.3	21.1	10.8	11.4
Lines of Code	16.4	15.3	14.2	12.0	9.9	16.2	17.5	14.4	9.38
Cyclomatic Complexity	18.2	14.2	18.7	15.9	8.9	17.2	24.2	16.7	11.5
CodePlexity (Ours)	26.7	21.3	21.0	15.8	**14.1**	**25.6**	**26.6**	**24.9**	**17.3**

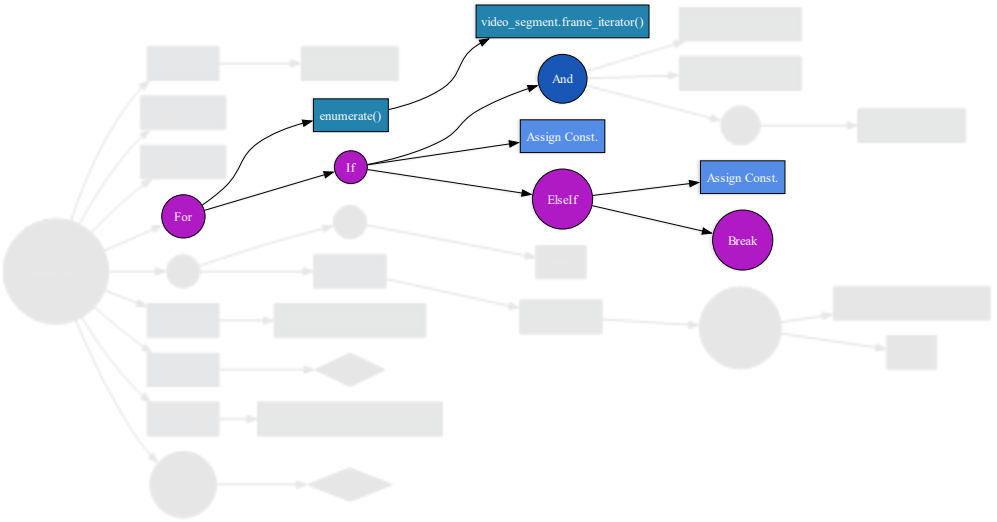
Analysis

QUERY: What did the man sitting on top do after he came off the person on the ground?



Analysis

QUERY: What did the man sitting on top do after he came off the person on the ground?



Generating a *Hard* dataset

CodePlex-QA Generations are *all* complex:

What does the waiter do after taking orders from the customers?

Who was the first to play the piano in the video?

What was the person's reaction after picking up the broom?

What action does the barber do to the customer multiple times in the video?

Who was the first person to walk during the conversation?

What is the man wearing while riding the camel?

What is the person holding while reading the book?

What happens to the cards after they are split into four?

What action is performed by the salon worker after filing the customer's nails?

What is the man holding after he stands up?

What is the man in the blue shirt doing most of the time?

What is the patient doing after standing up from the bed?

What does the person do after sneezing?

Who did the driver talk to at the end of the video?

What is the man in the tuxedo doing during the cheering?

Summary of Contributions

1. We demonstrate that generated code complexity can serve as a robust metric of question complexity in VideoQA and propose a novel approach for automatically quantifying it.

2. We present CodePlexity, a novel approach that identifies the key sources of complexity for existing VideoQA models.

3. Using CodePlexity, we automatically construct CodePlexQA, a novel benchmark that is challenging for VideoQA methods.