Automatic Action Script Generation to Improve Execution Rate based on LLM in VirtualHome

Jin Aoyama
Aoyama Gakuin University
Sagamihara, Kanagawa, Japan
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan
c5623213@aoyama.jp

Takeshi Morita
Aoyama Gakuin University
Sagamihara, Kanagawa, Japan
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan
morita@it.aoyama.ac.jp

Shusaku Egami
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan

Takanori Ugai
Fujitsu Limited
Kanagawa, Japan
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan

Ken Fukuda
National Institute of Advanced Industrial Science and Technology (AIST)
Tokyo, Japan

ABSTRACT

VirtualHome is a home simulator that enables agents to perform tasks by following action scripts. The scripts contain instructions involving specific actions and objects. However, creating the scripts manually is a time-consuming process. Natural language statements can be used to communicate intentions and requirements, making the process more intuitive. While some research has been conducted on automatically generating action scripts from natural language sentences, the current executable rate of the scripts is only around 35-40%.

This research aims to develop action scripts that agents can execute on the VirtualHome simulator. Our system takes daily activity descriptions and utilizes a Large Language Model (GPT-3.5) and SentenceTransformers to generate action scripts. It also verifies the execution of the generated scripts using the VirtualHome simulator and adjusts them as needed.

To assess the system's performance, we use "LCSscore" and the execution rate. LCSscore is based on the Longest Common Subsequence (LCS) between the correct and generated data. LCSscore requires the generated data to match perfectly with the correct data. Therefore, we introduce a new evaluation index called "Correctness" to evaluate the accuracy of the action sequence in case the order differs from the correct data.

The experimental results show that the execution rate was nearly 100%. However, no satisfactory results were obtained in LCSscore. The Correctness results revealed a tendency to have missing action steps, although it was less likely to generate incorrect action steps.

IJCKG'23, December 6–8, 2023, Tokyo, Japan

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; **Information extraction**; **Robotic planning**.

KEYWORDS

Embodied AI; VirtualHome; Activity Description; Action Script; LLM; Knowledge Graph;

ACM Reference Format:
Jin Aoyama, Takeshi Morita, Shusaku Egami, Takanori Ugai, and Ken Fukuda. 2023. Automatic Action Script Generation to Improve Execution Rate based on LLM in VirtualHome. In *Proceedings of The 12th International Joint Conference on Knowledge Graphs (IJCKG'23).* ACM, New York, NY, USA, 9 pages.

1 INTRODUCTION

Embodied AI [4] has undergone significant growth in recent years. It involves training agents such as virtual robots to perform complex tasks using a variety of objects in real and virtual environments [12, 15, 17, 20, 21]. However, this method requires a large amount of data.

One of the simulators used for Embodied AI is VirtualHome [10, 11, 13, 14]. This enables agents to perform tasks based on action scripts, which consist of instructions involving actions and objects. However, manually creating scripts is time-consuming. The process will be more intuitive if users can communicate their intentions and requirements using natural language statements.

Although some research has been conducted to generate action scripts automatically from natural language sentences [7, 13], the executable rate of generated action scripts is currently only around 35-40%. One of the challenges in efficiently developing data for agent learning is to improve the rate.

This study aims to create action scripts that agents can execute on VirtualHome (VH) using the description of daily activities. To achieve the objective, the proposed system employs Large Language Model (LLM: GPT-3.5) and SentenceTransformers [1][16] to generate action scripts. The system then verifies the execution of the generated scripts using VH and adjusts them as necessary.

In the evaluation experiment, the system performance was assessed using an evaluation index [13] based on the Longest Common Subsequence (LCS) with the generated data and the correct data. Additionally, to evaluate the accuracy when the order of the action sequence of the generated data differs from one of the correct data, a new evaluation index called "Correctness" has been introduced. This index considers the order of actions in a sequence and provides a more comprehensive evaluation of the accuracy of the generated scripts.

The contributions of this study are as follows:

(1) We have created a system that generates action scripts that agents can perform on VirtualHome from descriptions of their daily activities in natural language.
(2) Our system improves the execution rate of action scripts compared to existing research.
(3) We introduce a new evaluation index called "Correctness" to complement the existing evaluation index using LCS, which requires perfect matches with the correct action script to evaluate the accuracy of the action sequence in case the order differs from the correct data.

The remainder of this paper is organized as follows. Section 2 discusses related work, including Embodied AI and automatic action script generation. Section 3 explains the proposed system and its components. Section 4 describes the evaluation experiments, including the evaluation data, methods, results, and discussion. Finally, Section 5 concludes the paper.

Our reference implementation of the system is publicly available on Github.[2]

2 RELATED WORK

2.1 Embodied AI

Embodied AI [4] research involves training agents such as virtual robots to learn in both real-world and virtual environments, enabling them to solve complex tasks that involve multiple objects. Some examples of Embodied AI tasks include EQA (Embodied Question Answering) [20], where a virtual robot moves through its environment and perceives the environment to answer questions about it; VDN (Vision and Dialog Navigation) [21], where a virtual robot responds to specific instructions from humans and provides guidance; and ObjectNav (Object-Goal Navigation) [15], where a virtual robot finds and guides humans to specific objects within the environment.

Savva et al. [12] devised Habitat, a flexible and high-performance simulator platform that can handle configurable agents, sensors, and standard 3D datasets. Szot et al. [17] proposed Habitat 2.0 [12], which optimizes and alternates physics simulation and rendering processes, making it 100 times faster than the original Habitat platform.

Code 1: Example Action Scripts for "Relax on Sofa"

```
Label : Relax on sofa
Description : I turn on the tv and sit in the sofa. I watch the tv.

Action Script :
[WALK] <livingroom> (1)
[WALK] <tv> (1)
[FIND] <tv> (1)
[SWITCHON] <tv> (1)
[FIND] <sofa> (1)
[SIT] <sofa> (1)
[TURNTO] <tv> (1)
[WATCH] <tv> (1)
```

2.2 VirtualHome

VirtualHome [10, 11, 13, 14] is a platform that enables agents to perform tasks based on action scripts that consist of a series of steps. Code 1 provides an example of an action script. In this action script, the activity "Relax on sofa" in the home is represented by a series of steps: going to the living room, turning on the TV, sitting on the sofa, and watching TV. In each step, an agent acts with a few objects, and each object is identified by the object id.

The preconditions that specify the environmental constraints under which the action should be performed must be met to execute an action. Information about these preconditions can be found in VH's documentation [3] and GitHub [4], which includes information about the constraints on object properties, object state, and the positional relationship between the agent and the object.

VH is distributed with the dataset containing action labels, action descriptions, and corresponding action scripts, which have been collected through crowdsourcing. However, this dataset contains many action scripts that cannot be executed in VH due to objects that are not present or preconditions that are not satisfied. As a result, around 60% of the total dataset contains action scripts that cannot be executed on VH.

2.3 Action Scripts Generation

Two research focus on generating action scripts for VitualHome from natural language sentences [7, 13].

Huang et al.[7], used GPT-3 [1], Codex [2], and BERT [3] to generate action scripts based on abstract descriptions of actions expressed in natural language such as "Make breakfast." The goal was to generate action scripts that can be executed in VH.

This method involved using GPT-3 or Codex to plan actions based on an abstract description of daily life and then generate the next step. These action steps are then converted into action scripts using BERT. Based on the previous steps, the process is repeated for subsequent actions. However, executing the action scripts in VH is challenging, and based on experimental results, only 35% of the generated action scripts could be performed in VH.

[1] https://www.sbert.net/
[2] https://github.com/JinAoyama/actionscript_generation_in_vh
[3] http://virtual-home.org/documentation/master/kb/actions.html
[4] https://github.com/xavierpuigf/virtualhome/tree/master/virtualhome/simul

On the other hand, Puig et al. [13] treat the task as a translation problem and train a seq2seq model with MLE (Maximum Likelihood Estimation method) on the dataset. They then trained the model through reinforcement learning, where the reward was based on the LCS of correct and generated data and the rate of execution on the simulator. The experimental results showed that the ratio of executable action scripts among those generated is 35-40%.

Notably, more executable action scripts should be used to generate training data for the agent.

2.4 Formal Process Generation with LLM

This section describes research on utilizing LLM to generate formal processes from natural language sentences.

Kalakonda et al. [9] present Action-GPT, a framework that seeks to enhance the quality and generality of action generation using LLM, specifically GPT-3. The framework takes natural language sentences as input and utilizes a designed prompt function to generate suitable prompts from action phrases within the sentences. These prompts are then passed onto LLM to produce a detailed description of the physical action, ultimately represented as a sequence of human poses.

Wu et al. [19] introduce SPRING, a framework that employs LLM specifically, GPT-3, GPT-3.5, and GPT-4 to reason and learn knowledge to play a game. The framework's input consists of the LaTeX source of the original academic article of the game [6] and a description of the agent's current observations as the game context. SPRING utilizes a directed acyclic graph with game-related questions as nodes and dependencies as edges. It identifies the optimal action in the environment by scanning the directed acyclic graph and computing the LLM responses to each node in topological order.

Hwang et al. [8] focused on cooking recipes. The authors acknowledge that LLM can potentially adapt existing textual information into a more easily understood and usable form and thus proposed a prompt that breaks recipes down into more straightforward steps. The prompts are based on the original recipe and the ingredient list, prompting modifications to the recipe written in complex text. The authors designed and evaluated a human evaluation task to systematically compare the original and revised recipes. The results of the evaluation showed that the annotators preferred the modified recipe.

2.5 Knowledge Graph Reasoning Challenge for Social Issues 2022 (KGRC4SI)

The Knowledge Graph Reasoning Challenge for Social Issues 2022 (KGRC4SI) was held in Japan from September 2022 to May 2023. The challenge aimed to encourage the development of systems capable of identifying and explaining dangerous situations that might occur in the homes of older people [5].

To achieve this goal, the organizers used VirtualHome2KG [5] to convert videos into knowledge graphs that depict "when," "where," "who," "to what object," and "what action" was taken, as well as the resulting "state" and "location." [18]

The primary task of the KGRC4SI was to identify dangerous situations from the provided video and knowledge graph. In addition, the task included explaining the specific factors contributing to the dangerous condition and presenting safer alternatives within the given scope.

Since not all videos and knowledge graphs included dangerous situations, data representing numerous daily life situations are needed, and preparing data was one of the tasks of the KGRC4SI. All six entries in the KGRC4SI utilized knowledge graphs, and two employed knowledge graph embedding and graph neural networks. Since more data is needed for these works to function correctly, this task is also expected to contribute to this area.

The evaluation of this research will utilize data provided by KGRC4SI [18].

3 PROPOSED SYSTEM

3.1 Outline

This section presents our proposed method, which consists of three components. Our proposed method aims to generate an executable action plan from an input description of daily activities using a combination of LLM, SentenceTransformers, and VH.

The configuration of the proposed method is illustrated in Fig 1.

Component 1 takes an input description of daily activities and processes it using LLM to generate an action plan. The generated action plan contains a list of action steps separated by commas. For example, Component 1 receives "pick up phone and sit on the sofa" and outputs "grab cellphone, walk to sofa, sit on sofa."

Component 2 uses the SentenceTransformers sentence similarity technique to convert the action plan into an action script. The converted action script is then passed to Component 3. For example, Component 2 receives "grab cellphone, walk to sofa, sit on sofa" and outputs ["[GRAB] <cellphone> (1)", "[WALK] <sofa> (1)", "[SIT] <sofa> (1)"].

Component 3 takes an action script as input, verifies its execution using the VH, and modifies it if necessary. The modified action script is then passed back to Component 2 until the action script becomes executable and no further modifications are required. For example, Component 3 inputs ["[GRAB] <cellphone> (1)", "[WALK] <sofa> (1)", "[SIT] <sofa> (1)"] and outputs: ["[WALK] <cellphone> (1)", "[GRAB] <cellphone> (1)", "[WALK] <sofa> (1)", "[SIT] <sofa> (1)"]. This means that the input sequence of actions has added an extra step of "[WALK] <cellphone> (1)" at the beginning.

3.2 Rewriting Description with LLM

This section explains Component 1 in Fig 1. This research utilizes GPT-3.5 as the LLM to rephrase the input description. The input description is rewritten for two primary reasons.

(1) Depending on the author, the granularity of the input description's content may differ, with some being abstract and others being detailed. However, the input description must be described since VH necessitates detailed action instructions.
(2) The input description may include actions, objects, or rooms that do not exist in VH, making it impossible to simulate them.

We utilized LLM to enhance the input description. This involved creating a list of all possible actions that can be performed within

[5] https://challenge.knowledge-graph.jp/2022/index_en.html

Code 2: LLM Prompt in Component 1

```
Rewrite the description of activity entered by the user as a
    comma−separated action plans, using only defined
    Available Actions and Available Objects.
Do not omit the object of the action and do not use pronouns.

Definition:
Available Actions: ( Insert list of actions )
Available Objects: ( Insert list of objects )
```

Table 1: Hash Table for Action Script Conversion

Key	Value
Walk bathroom	[WALK] ⟨ bathroom ⟩ (1)
Walk bedroom	[WALK] ⟨ bedroom ⟩ (1)
⋮	⋮
Sit chair	[SIT] ⟨ chair ⟩ (1)
Sit sofa	[SIT] ⟨ sofa ⟩ (1)
⋮	⋮

VH and identifying all the objects and rooms within it. This information creates a detailed action plan description while ensuring the LLM output adheres to specific restrictions. Code 2 shows the main content of the LLM prompt.

3.3 Conversion to Action Script with Sentence Embeddings

This section discusses Component 2 in Figure 1.

Initially, we create all the possible action statements that can be performed on VH and objects and rooms within VH. We also created a hash table shown in Table 1 to obtain the action scripts corresponding to the created action statements.

Next, we generate a list of action plans by rewriting the input sentences using LLM. Then, based on these plans, we convert them into action scripts using the SentenceTransformers sentence similarity technique. For each item in the action plan list, we identify the action sentences with the highest similarity in the created dictionary and convert them into the corresponding action scripts. Algorithm 1 shows the algorithm of Component 2.

To achieve this, we utilized the pre-trained model all-MiniLM-L6-v2 [6] due to its high speed and quality.

3.4 Verification and Modification of Action Script Execution

This section describes Component 3 in Figure 1. Component 3 tests whether the generated action scripts can be run on VH. If the action script fails to execute, the system identifies the line that caused the error and outputs the error message. If the error message indicates that the action cannot be performed because the agent is too far away from the target object, the system fixes it by adding a line of action like "WALK" or "FIND" that will move the agent closer to the object. However, if the action script fails to be executed for reasons other than the errors above, then the failed line is deleted as it cannot be corrected.

[6]https://www.sbert.net/docs/pretrained_models.html

Algorithm 1 Component2 Algorithm

```
model ← SentenceTransformers
hashTable ← hash table for action acript conversion
keys ← hash table keys
keysModel ← model.encode(keys)

function mostSimilarity(sentence)
    cosSim ← cosSim(model.encode(sentence), keysModel)[0]
    allCombinations ← []

    for i = 0 to length(cosSim) − 1 do
        APPEND(allCombinations, [cosSim[i], i])
    end for

    allCombinations ← sorted(allCombinations,
    key=lambda x: x[0], reverse=True)

    return keys[allCombinations[0][1]]
end function

steps ← Component1 output
actionScript ← []
for i = 0 to length(steps) − 1 do
    APPEND(actionScript, hashTable.get(mostSimilarity(steps[i])))
end for
```

4 EXPERIMENTS

4.1 Experiment Evaluation Overview

In this evaluation experiment, we assess the accuracy and the ratio of action scripts generated from natural language descriptions of daily activities with two datasets. To measure accuracy, we use the LCS-based evaluation index [13]: "LCSscore". However, this index requires perfect matches generated action scripts with the correct action scripts. Therefore, we have defined a new evaluation index called "Correctness" to assess the accuracy of the action sequence even when the order of the generated data differs from the correct data.

4.2 Evaluation Datasets

VH comprises seven houses called "scenes", each with various rooms and objects like Table 2.

Two datasets, VH dataset [13] and KGRC4SI dataset [18], are used in this research. VH dataset is a social cloud dataset that provides data for each scene separately. However, Puig et al. [13] use the VH dataset for evaluation without segregation into corresponding scenes. We use their results as a baseline experiment. The KGRC4IS dataset was created for KGRC4SI to develop systems that can identify and explain dangerous situations in the homes of older people. Table 3 shows the number of evaluation data available for each

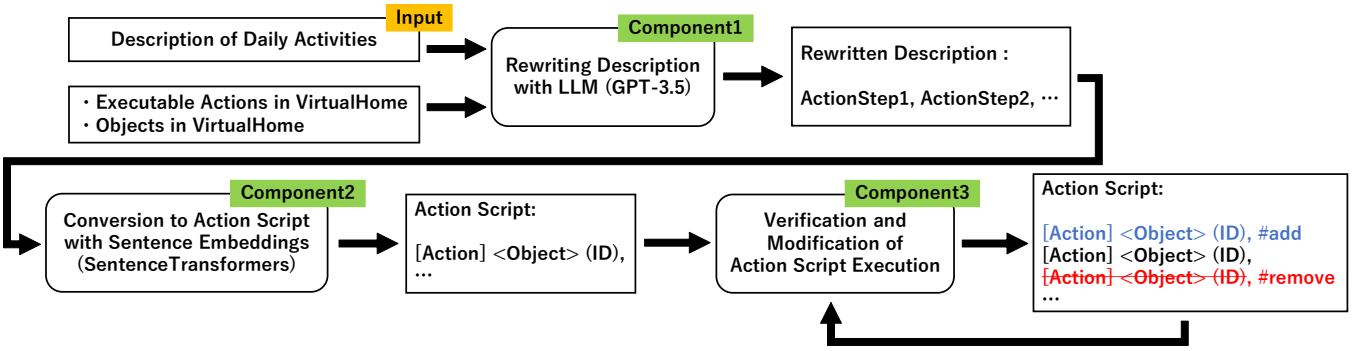

Figure 1: Method Configuration

Table 2: Number of Rooms and Objects in Each Scene

Env	Room		Object	
	types	total	types	total
Scene1	4	4	111	439
Scene2	4	4	98	315
Scene3	4	5	98	395
Scene4	4	4	107	354
Scene5	4	4	94	353
Scene6	4	4	96	320
Scene7	4	4	121	349

Table 3: Number of Action Scripts for Each Scene

Env	VH dataset		KGRC4IS dataset	
	original	target	original	target
Scene1	518	194	133	98
Scene2	510	214	77	48
Scene3	525	199	119	85
Scene4	520	169	66	46
Scene5	500	206	122	82
Scene6	497	199	126	92
Scene7	1023	358	63	40
total	4093	1539	706	491

scene. Data containing actions and objects that do not exist in the corresponding scenes or cannot be executed are excluded from this experiment.

4.3 Evaluation Index

We evaluate the proposed system using three indices: LCSscore, Correctness, and Simulator.

LCSscore

The LCSscore is calculated based on the Longest Common Subsequence (LCS) between the generated and correct action scripts. It is defined as the length of the LCS divided by the greater lengths of the generated and correct action scripts. To evaluate the similarity between different action scripts, we use three different methods: LCSscore(Action), which is calculated with LCS only on actions; LCSscore(Object), which is calculated with LCS only on objects; and LCSscore(Step), which is calculated with LCS on steps(combinations of actions and objects).

$$GAL = GeneratedActionList \tag{1}$$
$$GOL = GeneratedObjectsList \tag{2}$$
$$GSL = GeneratedStepList \tag{3}$$
$$CAL = CorrectActionList \tag{4}$$
$$COL = CorrectObjectsList \tag{5}$$
$$CSL = CorrectStepList \tag{6}$$
$$LCSscore(Action) = \frac{LCS(GAL, CAL)}{max(length(GAL), length(CAL))} \tag{7}$$
$$LCSscore(Object) = \frac{LCS(GOL, COL)}{max(length(GOL), length(COL))} \tag{8}$$
$$LCSscore(Step) = \frac{LCS(GSL, CSL)}{max(length(GSL), length(CSL))} \tag{9}$$

Correctness

In an action script, "WALK" and "FIND" are used for the same purpose, which is to approach an object. Thus, the agent's behavior remains the same whether the "WALK" or "FIND" action is taken. LCSscore treats the sequence of actions as different when the order of actions is not identical to the correct data, even if the actions' contents are accurate. This is an issue because the order of actions does not affect the agent's behavior.

To address this problem, we developed a new evaluation index called "Correctness," This metric measures the correctness of the generated data, considering the order of actions. It is based on how much the generated data can match the correct data steps (combinations of actions and objects). The Algorithm 2 counts the number of steps in which the generated data matches the steps of the correct data as the number of correct answers. Precision, Recall, and F1 values are calculated to evaluate the accuracy of the generated data.

Simulator

The Simulator is an index used to evaluate the ratio of executable action scripts. It can be defined in three ways:

(1) The ratio of action scripts in which all steps are executable out of all action scripts.

Algorithm 2 Correctness Algorithm

```
function pretreatment(L)
    for i = 0 to length(L)) − 1 do
        L[i] ← L[i].replace("FIND", "WALK")
    end for

    compressedList ← []
    for i = 0 to length(L) − 1 do
        if i = 0 or compressedList[−1] ≠ L[i] or "WALK" not in
L[i] then
            APPEND(compressedList, L[i])
        end if
    end for
    return compressedList
end function

A ← correct data
B ← generated data
A ← pretreatment(A)
B ← pretreatment(B)
B_length ← length(B)
count ← 0

for i = 0 to length(A) − 1 do
    for j = 0 to length(B) − 1 do
        if A[i] = B[i] then
            count += 1
            Remove B[j] from B
            break
        end if
    end for
end for

Precision ← count/B_length
Recall ← count/length(A)
F1 ← 2 × Precision × Recall/(Precision+Recall)
```

(2) The ratio of steps that could be executed from all steps in the action script.
(3) The average ratio of all action scripts from the first step to the step that could be executed.

The existing research [13] used for comparison did not specify the definition of the Simulator, but it was assumed that they used the first description above. Therefore, this research will use the first description to define the Simulator.

4.4 Evaluation Methods

Action scripts were generated for the VH dataset and the KGRC4SI dataset using the comment on the second line of the data as input. The columns of actions after the fourth line were considered correct and compared with the generated action scripts.

In VH, objects are related to classes (e.g., "fruit": "watermelon", "apple", "banana"]). The class names in the action script are automatically mapped to the object names when the simulation is run. The action scripts in the VH dataset are described by class name, except for objects that do not belong to a class.

Our system generates action scripts based on object names, which may lead to discrepancies compared to the VH dataset's action scripts. As a result, we also create and evaluate action scripts where the object names in the generated scripts are replaced with class names. This object-to-class names mapping is performed using dictionaries[7].

For assessing the system using the LCSscore as the evaluation index, we will use the results of the experiment conducted by Puig et al. [13] on the VH dataset as a baseline for comparison. The evaluation values for the KGRC4SI dataset are also given for reference.

Regarding the evaluation using Correctness as the evaluation index, the respective evaluation values for the VH and KGRC4SI datasets are presented.

Note that version 2.3 of the simulator was used for the evaluation with the VH dataset. The version of the simulator used in the baseline experiment is unknown. For evaluation with the KGRC4SI dataset, we used the AIST version of the VH simulator [8] with extended executable actions. The KGRS4SI dataset is designed explicitly for the AIST version of the VH simulator.

4.5 Experiment Results

Evaluation with LCSscore and Simulator

In this research, we measured the LCSscore and Simulator using the VH data set and compared them with the baseline. Table 4 shows the results of these measurements. The heading "Action" shows the value of the LCS measured only by the action in the action script. "Objects" shows the LCS measured only for the objects of the action script, and "Steps" shows the LCS measured in steps of the action script (combinations of actions and objects). The "Mean" column represents the average of "Action," "Objects," and "Steps." The "Simulator" column shows the Simulator values in the simulator.

Methods MLE, PG(LCS), and PG(LCS+Sim) were taken from the descriptions of the existing research [13] mentioned earlier, and the scene used for the Simulator measurements is unknown. MLE is a Seq2Seq model trained using Maximum Likelihood Estimation methods. PG(LCS) is a reinforcement learning model that uses LCS rewards for MLE, and PG(LCS+Sim) is a reinforcement learning model that uses LCS and Simulator rewards for MLE.

In this research, we used two methods: Ours(Class) and Ours(Object), and the results presented are an average of the experimental outcomes for all scenes. Ours(Class) refers to action scripts generated using the class names for objects, while Ours(Object) refers to action scripts generated using the object names. Ours(Object) has a low LCSscore but the Simulator close to 100%. On the other hand, Ours(Class) has

[7] https://github.com/xavierpuigf/virtualhome/blob/master/virtualhome/resources/class_name_equivalence.json
[8] https://github.com/aistairc/virtualhome_unity_aist/releases/

Table 4: LCSscore

Method	Action	Objects	Steps	Mean	Simulator (%)
MLE	0.777	0.723	0.686	0.729	38.6
PG(LCS)	0.803	0.766	0.732	0.767	35.5
PG(LCS+Sim)	0.806	0.775	0.740	0.774	39.8
Ours(Class)	0.493	0.477	0.415	0.462	55.3
Ours(Object)	0.493	0.119	0.095	0.235	99.2

Table 5: LCSscore(KGRC4SI dataset) (which contains Abnormal category data

Env	Action	Objects	Steps	Mean	Simulator (%)
Scene1	0.448	0.429	0.352	0.409	90.8
Scene2	0.520	0.529	0.430	0.493	79.1
Scene3	0.439	0.396	0.347	0.394	92.9
Scene4	0.472	0.445	0.371	0.429	82.6
Scene5	0.446	0.407	0.340	0.398	93.9
Scene6	0.439	0.380	0.330	0.383	93.5
Scene7	0.481	0.493	0.417	0.464	77.5
Mean	0.464	0.440	0.369	0.424	87.2

an inferior LCSscore compared to the baseline. However, its Simulator has a slight advantage.

Table 5 shows the measurement results of LCSscore and Simulator using the KGRC4SI dataset. It shows the outcomes for each scene utilized in the experiment. Although the Simulator is 87%, the LCSscore is low.

Evaluation with Correctness

In this experiment, we measured the Correctness of the action scripts generated with the class names and object names using the VH dataset and KGRC4SI dataset. Table 6 shows the results of measuring the Correctness of the action scripts generated with the class names using the VH dataset. Precision is high. However, the Recall is not satisfactory. On the other hand, Table 6 shows the results of measuring the Correctness of the action scripts generated with the object names using the VH dataset. Both Precision and Recall are low. Table 7 shows the results of measuring Correctness using the KGRC4SI dataset. Compared to Table 6, the values are lower for all scenes.

4.6 Discussion

The results presented in Section 4.5 show that Ours(Object) has a low LCSscore. This is because the VH dataset has action scripts written in the class names, which we do not consider a significant practical issue for the generated action scripts.

Ours(Class), on the other hand, has a low Simulator because classes in action scripts are only sometimes mapped to objects in the VH environment during simulation. However, this is not considered a practical problem since Ours(Class) is generated for evaluation purposes.

We are initially considering simulating Ours(Object), and although the LCSscore results for Ours(Class) are inferior to the baseline, the Simulator results for Ours(Object) are superior.

Table 6 in Section 4.5 shows the results that show the generated step's high Precision, indicating that it is more likely to be correct and less likely to generate an incorrect step. However, The Recall results reveal that only about 54% of the steps in the correct action script were fulfilled.

These results show that while the wrong steps are unlikely to be generated, the missing steps are numerous. Many missing steps may be due to including action steps in the corresponding action scripts that must be explicitly described in the input action descriptions. In such a situation, it becomes challenging to complement these action steps with LLM.

Moreover, if the input action description is abstract, LLM will generate general action steps. However, an abstract description is difficult to match because multiple possible action steps exist.

The Simulator evaluation results show that the Simulator may be less than 100% due to the initial position of the agent in the VH environment. The execution of an action is dependent on the relative position of the agent and the target object. In VH, the agent's initial position is randomly assigned when not specified. This may explain why the Simulator is less than 100%. Therefore, it is essential to specify the initial position when generating the action script, and the fixed initial position information should be included as part of the generated data.

Regarding the KGRC4SI dataset in Table 3, some action scripts failed to execute during the experiment despite creating videos and KGs with VirtualHome2KG. This may be due to the initial settings of the execution environment, such as the initial position.

There are three possible reasons why the evaluation results using the KGRC4SI dataset in Table 7 are lower than those using the VH dataset in Table 6.

(1) The number of actions in the AIST version of VH has more than doubled compared to the original VH, making it difficult to match actions. Adding actions such as Walk, WalkForward, and WalkToward to natural language expressions has made it challenging to determine which action is being referred to, thus lowering the evaluation index.
(2) Since the action scripts in the KGRC4SI of the dataset are written using object names, it is important to identify the objects precisely.
(3) The KGRC4SI dataset is categorized according to the data type, with Abnormal data describing dangerous behaviors in the home. However, the input text for Abnormal data includes an extra sentence explaining why the behavior is dangerous, even though it is irrelevant to the content.

Table 7 shows the results of measuring Correctness from the KGRC4SI dataset, excluding the data in a category named Abnormal, to verify the third reason. This is because many of the action scripts

Table 6: Correctness

Env	Class			Object		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Scene1	0.740	0.581	0.651	0.157	0.131	0.143
Scene2	0.703	0.548	0.616	0.161	0.124	0.140
Scene3	0.726	0.531	0.613	0.140	0.103	0.119
Scene4	0.689	0.491	0.573	0.215	0.155	0.180
Scene5	0.678	0.516	0.586	0.127	0.100	0.112
Scene6	0.732	0.551	0.629	0.178	0.135	0.154
Scene7	0.682	0.549	0.608	0.197	0.154	0.173
Mean	0.707	0.538	0.611	0.168	0.129	0.146

Table 7: Correctness(KGRC4SI dataset) (# of Ab. is the number of Abnormal Category data)

Env	with Abnormal category data			without Abnormal category data			
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	# of Ab.
Scene1	0.570	0.469	0.514	0.591	0.481	0.530	19
Scene2	0.607	0.538	0.570	0.634	0.563	0.596	11
Scene3	0.539	0.452	0.492	0.551	0.468	0.506	16
Scene4	0.649	0.494	0.561	0.680	0.499	0.575	10
Scene5	0.548	0.443	0.490	0.563	0.462	0.507	16
Scene6	0.548	0.449	0.494	0.568	0.472	0.515	17
Scene7	0.652	0.518	0.577	0.671	0.541	0.599	8
Mean	0.588	0.480	0.528	0.608	0.498	0.547	-

in the Abnormal category had extra descriptions attached to them. The right column shows the number of data in the category initially included. The evaluation values are slightly higher than the results in Table 7. We think this confirms the third reason.

The lower evaluation results using the KGRC4SI dataset in Table 5 compared to those using the VH dataset in Table 4 can be attributed to the abovementioned reasons. Additionally, the execution conditions of the actions available in the AIST version of VH depend on the location relationship between the agent and the target object. This may be one of the reasons for the low Simulator compared to the evaluation results using the VH dataset.

5 CONCLUSION

This research proposed a system that generates action scripts that virtual agents can execute in VH, using descriptions of daily activities as input. The system utilizes LLM (GPT-3.5) and SentenceTransformers to generate action scripts, which are verified using VH. The system detects errors and refines the action scripts to improve accuracy. This includes shortening the distance to the objects and eliminating actions that cannot be executed.

In the initial phase, the proposed system was tested with the VH dataset and compared to existing research using metrics like LCSscore, which measures the similarity between generated and correct data and the execution rate (Simulator). While the LCScore of the proposed system was lower than that of existing research. Since the Simulator reached nearly 100%, achieving the primary goal of generating executable action scripts. The evaluation indicated Precision of Correctness is high, but low Recall. This suggests that generating wrong steps was less likely, but many steps needed to be included. Furthermore, it was observed that the Simulator did not reach 100%. This is due to the initial position of the agent. Addressing this issue involves generating action scripts that consider the agent's initial point.

Using the KGRC4SI dataset, the evaluation resulted in a Simulator score of 87%. While partially achieving the objective of generating executable action scripts, LCSscore and Correctness were lower than the results from the VH dataset. We discussed the reasons for this difference.

A potential future solution to improve LCSscore and Correctness could involve machine learning, using our experiment dataset for training data. However, the baseline highlights the challenge of generating executable action scripts. A potential countermeasure would be to adapt the action scripts to the environment after generation or verify their execution and modify them as needed.

Furthermore, the performance of LLM is significantly dependent on the content of the prompt. Therefore, it will be necessary in the future to compare and analyze the results when the prompt content is altered.

Finally, it should be noted that this research is based on VH and has not been tested in real-world experiments. Therefore, conducting real-world experiments may be a potential future challenge.

ACKNOWLEDGMENTS

This paper is based on results obtained from projects, JPNP20006 and JPNP180013, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). This work was partially supported by JSPS KAKENHI Grant Number 23K11221.

REFERENCES

[1] 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. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *CoRR* abs/2107.03374 (2021). arXiv:2107.03374 https://arxiv.org/abs/2107.03374

[3] 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*. 4171–4186. https://doi.org/10.18653/v1/N19-1423

[4] Jiafei Duan, Samson Yu, Hui Li Tan, Hongyuan Zhu, and Cheston Tan. 2022. A Survey of Embodied AI: From Simulators to Research Tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence* 6, 2 (2022), 230–244. https://doi.org/10.1109/TETCI.2022.3141105

[5] Shusaku Egam, Takanori Ugai, Mikiko Oono, Koji Kitamura, and Ken Fukuda. 2023. Synthesizing Event-centric Knowledge Graphs of Daily Activities using Virtual Space. *IEEE Access* (2023), 1–1. https://doi.org/10.1109/access.2023.3253807

[6] Danijar Hafner. 2021. Benchmarking the Spectrum of Agent Capabilities. In *International Conference on Learning Representations*.

[7] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*. PMLR, 9118–9147.

[8] Alyssa Hwang, Bryan Li, Zhaoyi Hou, and Dan Roth. 2023. Large Language Models as Sous Chefs: Revising Recipes with GPT-3. *arXiv preprint arXiv:2306.13986* (2023).

[9] Sai Shashank Kalakonda, Shubh Maheshwari, and Ravi Kiran Sarvadevabhatla. 2023. Action-GPT: Leveraging Large-scale Language Models for Improved and Generalized Action Generation. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*. 31–36. https://doi.org/10.1109/ICME55011.2023.00014

[10] Shuang Li, Xavier Puig, Chris Paxton, Yilun Du, Clinton Wang, Linxi Fan, Tao Chen, De-An Huang, Ekin Akyürek, Anima Anandkumar, et al. 2022. Pre-trained language models for interactive decision-making. *Advances in Neural Information Processing Systems* 35 (2022), 31199–31212.

[11] Yuan-Hong Liao, Xavier Puig, Marko Boben, Antonio Torralba, and Sanja Fidler. 2019. Synthesizing Environment-Aware Activities via Activity Sketches. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] Manolis Savva, Abhishek Kadian*, Oleksandr Maksymets*, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. 2019. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[13] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8494–8502.

[14] Xavier Puig, Tianmin Shu, Shuang Li, Zilin Wang, Yuan-Hong Liao, Joshua B Tenenbaum, Sanja Fidler, and Antonio Torralba. 2020. Watch-And-Help: A Challenge for Social Perception and Human-AI Collaboration. In *International Conference on Learning Representations*.

[15] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. 2023. PIRLNav: Pretraining With Imitation and RL Finetuning for ObjectNav. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 17896–17906.

[16] 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*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[17] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. 2021. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[18] Takanori Ugai, Shusaku Egami, Swe Nwe Nwe Htun, Kouji Kozaki, Takahiro Kawamura, and Ken Fukuda. 2023. *Synthetic Multimodal Dataset for Daily Life Activities*. https://doi.org/10.5281/zenodo.8046267

[19] Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. 2023. SPRING: GPT-4 Out-performs RL Algorithms by Studying Papers and Reasoning. *arXiv preprint arXiv:2305.15486* (2023).

[20] Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, Tamara L Berg, and Dhruv Batra. 2019. Multi-target embodied question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6309–6318.

[21] Yi Zhu, Fengda Zhu, Zhaohuan Zhan, Bingqian Lin, Jianbin Jiao, Xiaojun Chang, and Xiaodan Liang. 2020. Vision-dialog navigation by exploring cross-modal memory. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10730–10739.