Apple Core-dination: Linguistic Feedback and Learning in a Speech-to-Action Shared World Game

Susann Boy[*]
Saarland University

AriaRay Brown
Saarland University

Morgan Wixted
Saarland University

{susannb,ariaray,morganw}@coli.uni-saarland.de

Introduction. We investigate the question of how adaptive feedback from a virtual agent impacts the linguistic input of the user in a shared world game environment. To do so, we carry out an exploratory pilot study to observe how individualized linguistic feedback affects the user's instructional speech input. We introduce a speech-controlled game, Apple Core-dination, in which an agent learns complex tasks using a base knowledge of simple actions.

Apple Core-dination is a shared-world language building game that situates the user and agent in a common virtual space. The agent is equipped with a learning mechanism for mapping new commands to sequences of simple actions, as well as the ability to incorporate user input into written responses. To build a framework for mapping new language and actions to existing knowledge, our game adopts concepts from the semantic parsing model of Artzi and Zettlemoyer (2013), which makes use of situated cues and common constructs found in instructional language. We seed the knowledge of the agent by providing an initial lexicon of text-to-action mappings for basic movements and communicative functionalities. The agent repeatedly shares its internal knowledge state by responding to what it knows and does not know about language meaning and the shared environment.

Our paper focuses on the linguistic feedback loop in order to analyze the nature of user input. Feedback from the agent is provided in the form of visual movement and written linguistic responses. Particular attention is given to incorporating user input into agent responses and updating the speech-to-action mappings based on commands provided by the user. Since the portrayal of an agent can affect how the user perceives its emotional intelligence (Chita-Tegmark et al., 2019) and trustworthiness (Fan et al., 2017), we consider visual cues that may also contribute to the user's choice of language when interacting. The focus of building agent knowledge is based on enriching the mappings of linguistic input to sequences of simple actions for a given user session. Our system also gives the user control over the mechanism of speech input by providing multiple key-press activated speech-to-text models. Through our pilot study, we analyze task success and compare the lexical features of user input. Results show variation in input length and lexical variety across users, suggesting a correlation between the two that can be studied further.

Background. Cooperative language building games allow the user and agent to jointly accomplish a shared task through language collaboration. Previous research in cooperative language building games has shown that success can be achieved through accommodation between the user and the agent, where both user and agent adapt their communication to adjust to the communicative needs of the other. The interaction can result in user language becoming "more consistent, less verbose, and more precise" as users adapt to the perceived abilities of the computer (Wang et al., 2016). Accommodation of user language supports findings in human-human interaction in which speakers tend to infer the linguistic and situational awareness of their interlocutors (Pickering and Garrod, 2004). Even low effort from the computer agent to cooperate with the user can lead users to believe common ground is established (Chai et al., 2014). Perceived common ground, or a common understanding of the shared environment, can be achieved through the goal of a shared task along with the agent's effort to make apparent its internal knowledge state.

Speech-to-Action Shared World Game. Apple Core-dination embodies the computer in a person-

[*] All authors have equal contribution.

ified agent that prompts the user to teach it new tasks through learning and cooperative interaction. The agent's internal knowledge of its progress is shared through linguistic and action-based feedback. We use the player's own speech (the input) to customize agent responses (the output). The resulting system allows us to analyze how individualized feedback affects the nature of user input. Game implementation code is available on GitHub[1]

Game Environment. The game begins with an introduction in which the user first learns to interact with the agent by naming it through speech. The goal of the game is to complete a set of tasks framed as **complex actions**. The user teaches the agent to achieve the given tasks through multi-step speech commands that build upon **basic actions** already known by the agent. A mid-session view of the game screen is shown in Figure 1. Additional game views are offered in Appendix A.

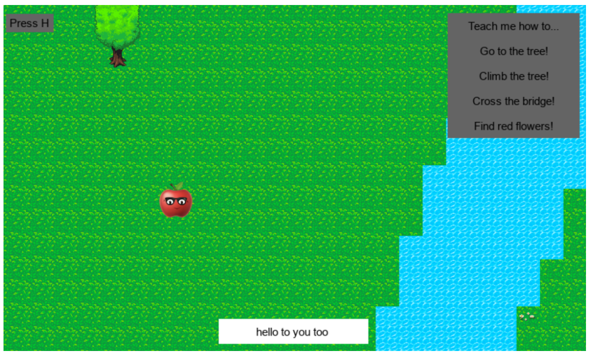


Figure 1: The game interface. The tasks are displayed on the right side and at the bottom appears the agent's feedback while executing a command.

The basic actions of the agent can be triggered by uttering corresponding commands. Basic actions include simple movement functions in all four directions (left, right, up, down), general movement (move), object destinations (e.g. tree, bridge, itself), repetition of previous actions, greeting the user, and reacting to positive and negative feedback such as *good job* or *no, not that*.

The agent can store new learned commands into its knowledge base via the `yes` function triggered by the word *yes*. Upon completion of a task, the `yes` function is silently called in order to add the the relevant task instructions to the agent's learned commands. A subsequent utterance of the learned task instruction (e.g. *climb the tree*) will directly access the `climb_tree` function. Although the

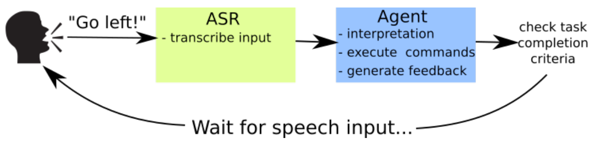


Figure 2: Description of the game loop. The user gives speech input which is transcribed by a speech-to-text model. The transcribed instruction is then interpreted by the agent and feedback is generated. If the instruction is in the agent's knowledge base, an action is executed. The game checks if the action led to the completion of a task and then waits for the next instruction.

agent can move to the location of objects in its knowledge base, it does not contain action functions to interact with the objects. While executing a command, the agent gives a response in the form of text feedback on the screen and movement to a destination when applicable.

Speech-to-Text. Our speech-to-text system provides users with the ability to play our game using their voice. We currently have two systems implemented: Google Cloud Speech API (Zhang, 2017) and SpeechBrain's pre-trained automatic speech recognition (ASR) model (Ravanelli et al., 2021). With both of these models, we can experiment to see the interactions between the users and the ability to choose between different speech recognition systems.

Our speech-to-text pipeline is as follows: our users give speech input by pressing and holding a key (*M* or *SPACE*) that accesses the speech-to-text model. The input is recorded using Python's speech recognition library (Zhang, 2017) and transcribed using the selected model. A single .wav file is saved locally and rewritten for each utterance that uses the SpeechBrain model.

Agent Knowledge and Language Parsing. The agent embodies the language knowledge and learning mechanism of the game. The agent has access to a knowledge base, a **transcript** that serves as long-term memory, interpretation methods for parsing linguistic input, and internal properties that act as working memory. The agent's **knowledge** also contains **hidden actions**, or functions representing the complex game tasks. These functions are hidden in the sense that the agent cannot initially access them through their corresponding instructions. Figure 3 is referenced for explanation.

The knowledge base contains a **lexicon** of known words mapped to action functions, a dictionary of **learned phrases** that fills as the game

[1]https://github.com/ariabee/applecore

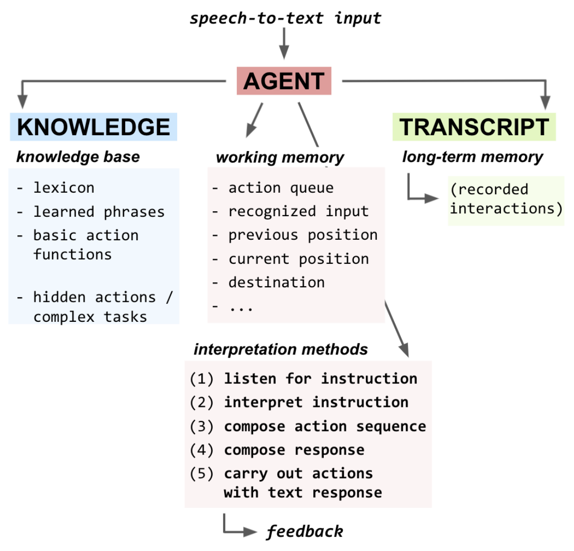


Figure 3: Model of the agent knowledge and language processing mechanism in Apple Core-dination.

progresses, a list of actions mapping indices to action functions, and the individual action functions which represent basic actions. Each piece of language that the agent recognizes is mapped to the related **action functions**. Knowledge also holds the hidden actions that represent the given complex tasks that the agent cannot yet access through mapped language, e.g. the tasks *climb the tree*, *cross the bridge*, and *find red flowers*.

The transcript serves the role of **long-term memory**, in case the agent needs to reference a previous instruction, action sequence, or response. The transcript updates and saves to a structured data file in every game loop. The file contains information about which of the two speech-to-text models was accessed, which instruction the user has given to the agent (output of the speech-to-text model), the response of the agent, and what sequence of actions was executed after the command. We track the progression of the game by storing whether the performed actions led to the completion of a task.

The agent has the ability to map both new and recognized utterances to actions. The function for confirmation and learning links the previous command to a successful sequence of actions performed by the agent. When an instruction is given, the agent parses and composes the utterance into an executable action sequence, or list of indexed action functions. Each function of the event is then executed during the game loop. Successfully completing a task will map the original task command to the hidden task function, adding the phrase-to-action mapping to the agent's learned phrases. Henceforth, input containing the task command will trigger the learned task function.

We implement a simplified parsing mechanism to compose input strings into meaningful combinations of mapped action functions. Parsing involves first searching for learned phrases, followed by checking the remaining string for recognized words that exist in the lexicon. Known words and phrases along with their corresponding actions are stored in the agent's **working memory**. The list of actions is then composed into a list of single actions and combined based on semantic meaning. The completed sequence is stored in an **action queue**. Each action contributes to building the agent's response by returning a string based on the input that triggered the action. Once the response is composed, each action in the queue is executed one at a time in the game loop.

Linguistic Feedback. When a user gives an instruction (e.g. *walk there to the left*), part of the utterance is incorporated into the response of the agent (e.g. *walking somewhere going left*). Feedback from the agent then becomes input for the user, while also informing the user about the nature of the agent's knowledge, or what it does or does not know about language meaning and its surrounding environment.

The nature of individualized feedback is dependent on each basic action function. The part of the utterance that triggered each action is sent as an argument when processing the function for response-only output. Constructed feedback from each mapped utterance is concatenated to form the agent's response. For instructions containing learned phrases, the entire learned string is returned as feedback. For example, the instruction *hop up to the tree* will initially result in the response, *going up going to the tree*. When the phrase is learned, the agent will respond instead with the recognized learned phrase *hop up to the tree* that maps to moving up and to the position of the tree. An example instruction with resulting linguistic feedback is provided in Appendix B.

The agent also provides feedback for fully unrecognized input; e.g. *climb the tall plant* would result in the agent response: *how do I climb the tall plant?*, where the unrecognized phrase is incorporated into agent feedback. Upon learning this new command, the agent would respond: *yes, I learned to climb the tall plant*.

Pilot Study. We tested our game in an exploratory pilot study of five English-speaking volunteers who were only instructed to play the game but not obligated to complete all tasks in the game. Participants were given instructions for using each speech-to-test system, but not directed to use either or both. We evaluate the interaction between agent and user with the agent's long-term memory, the transcript. The transcript files include information about which key they pressed to give an instruction, the instruction transcribed by the speech-to-text model, the actions triggered by the command, the linguistic feedback given by the agent and whether the command led to the completion of a task.

In completed transcripts, the instructions were manually annotated by one of the authors as *clean* if the speech-to-text model correctly transcribed the instruction. Only cleaned instructions were used for analysis, as incorrect model output does not indicate a successful interaction between user and agent. However, if the intended action was triggered even though one or more words were transcribed wrongly, the instruction was still marked as clean. The users were asked to anonymously send us or upload their transcript files. We gathered 219 commands in total, of which 182 were annotated as clean transcriptions.

Our main interest is how the user interacts with the agent. The usage of two speech-to-text models enables us to evaluate their performance based on the preference of the users.

Results. We examine two lexical factors of the instructions: *quantity* and *quality* of the command. Quantity in this context refers to the length of the phrase (how many words were used), and quality is a measure of how large the user's vocabulary is (how many different words were used). Table 1 shows relevant statistics regarding the instructions by each user.

Instruction lengths and vocabulary size were obtained from the cleaned transcripts (only utterances that the speech-to-text model transcribed correctly). We found that the range of the instruction length is quite wide (one to 11 words), but the average instruction length of 3.7 words suggests that users tend to use single short commands. The agent can perform multiple actions, but the players rarely made use of this. This observation is in agreement with studies by Marge et al. (2020). The quality of a user's instructions is defined by their vocabulary size, which was determined by counting the number of unique words uttered by each user. The average vocabulary size is 35.4 words. User 4, who uttered the shortest instructions on average, also has the smallest vocabulary size (20 words), and User 2 has the largest vocabulary (53 words), but their instruction length is below average. Future work will more precisely examine the interaction between quality and quantity of user input and whether it corresponds to agent feedback or individual participant differences.

	User1	User2	User3	User4	User5	average
instruction length (min-max)	1-5	1-7	3-11	1-5	1-11	1.4-7.8
average instruction length	3.1	3.3	5.8	2.3	3.8	3.7
vocab size	28	53	38	20	38	35.4
M key (clean output)	44 (27)	0	0	4 (1)	0	9.6
SPACE bar (clean output)	4 (3)	52 (47)	19 (19)	46 (42)	50 (43)	34.2

Table 1: Relevant interaction data between each user and the agent: Shortest and longest instruction, average instruction length, vocabulary size (number of unique words) and number of times *M* (SpeechBrain) or *SPACE* (Google Speech) was pressed (and how many times the speech was transcribed well).

Task	User1	User2	User3	User4	User5	average
Go to the tree	1	2	2	5	2	2.4
Climb the tree			9		2	5.5
Cross the bridge		10		7	21	12.7
Find red flowers				7	15	11

Table 2: Instructions needed by each user until a task was completed.

The linguistic feedback helped the players to communicate successfully with the agent. They often repeated their previous command when the speech-to-text model did not correctly transcribe their utterance.

Table 2 shows how many instructions the users uttered until a task was completed. The first task (*go to the tree*) can be completed with one command since the agent has it already in its knowledge base. This task acts as positive reinforcement for the user to continue completing tasks and marks a common point across users where utterances become more task-directed. It was also the only task that all users completed: User 1 achieved it after one command, User 4 needed five commands. Only a single player completed all four tasks, which could mean that it is not clear how to complete some of the tasks or that the game is not engaging enough.

We used multiple speech-to-text models to evaluate which model was preferred by our users in terms of user interaction. Table 1 shows that the

SPACE bar was pressed more often than the *M* key, meaning that the Google model was accessed more often. Only User 1 used the *M* key in the majority of speech commands to the agent and three out of the five users did not use the second model at all. This could be due to preferred usability (the *SPACE* bar is easier to press than the *M* key), that one model produces cleaner and faster transcriptions compared to the other model, or simply that users forget there is a second model option.

Transcription accuracy from either model can affect feedback, user-agent interaction, and game progression in several cases of interest. In one example, the Google model transcribed the instruction *go up* as **co-op*, which resulted in the response *how do I co-op?* and no movement from the agent. For User 1 who accessed both models, the SpeechBrain model transcribed the instruction *go left* as *go *lift*, which resulted in the response of *going somewhere*, since the agent recognized only the word *go* from its knowledge base. Both examples exhibit failed instructions from transcription errors, since the input did not lead to successful parsing of the intended actions. However, only the former reveals the incorrect transcription. In the latter case, the user is not notified that *left* was absent from the feedback because it was transcribed as *lift*. In cases of misalignment due to transcription error, the user could attempt to clarify the command, if simple clarification seems possible, or opt for a new speech-to-text model altogether. For future experiments, we will encourage users to try either system to see if they have a preference, especially when it is clear that a command was not properly interpreted.

Future Work. Results from user-agent interactions in the pilot study inform improvements that can be made to the game environment. These include expanding the agent's baseline knowledge, increasing the complexity in agent parsing abilities, and more finely incorporating user input into feedback responses. More grammatically formed feedback for learned phrases could be processed with the addition of a semantic parser. The response for a learned phrase *dance left and slide right* would appear more realistic and sophisticated if it occurred from the agent in the gerund verb form as $\underline{\textit{dancing}}$ *left and* $\underline{\textit{sliding}}$ *right*.

We plan to carry out a larger comparative study of user input in Apple Core-dination. Players will be divided into two groups: one where the agent gives individualized linguistic feedback and one where it gives action-only feedback. By providing a clear choice of speech-to-text models, we can better assess the interaction between model accuracy, agent response, and user behavior to determine whether the user's preference for a model becomes a factor in resolving agent understanding.

Our current results show variation in quantity and quality of input across users, which will be investigated further. The user's language adjustment over time may also vary when the agent responses are shown to increasingly accommodate user input. Individualized linguistic feedback should better inform the user of how the agent processes input and adapts to new knowledge from the user. It should also allow users to speak to the agent through preferred utterances, and help them to resolve communication errors when they struggle to complete a task.

Acknowledgments. We would like to thank the anonymous reviewers for their valuable feedback. We also thank our anonymous participants for their help, and Dr. Lucia Donatelli for her insightful comments and suggestions throughout.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

Joyce Y. Chai, Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson. 2014. Collaborative effort towards common ground in situated human-robot dialogue. HRI '14, page 33–40, New York, NY, USA. Association for Computing Machinery.

M. Chita-Tegmark, M. Lohani, and M. Scheutz. 2019. Gender effects in perceptions of robots and humans with varying emotional intelligence. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 230–238.

Lisa Fan, Matthias Scheutz, Monika Lohani, Marissa McCoy, and Charlene Stokes. 2017. Do we need emotionally intelligent articial agents? first results of human perceptions of emotional intelligence in humans compared to robots. In *Proceedings of the Seventeenth International Conference on Intelligent Virtual Agents*.

Matthew Marge, Felix Gervits, Gordon Briggs, Matthias Scheutz, and Antonio Roque. 2020. Let's do that first! a comparative analysis of instruction-giving in human-human and human-robot situated dialogue. In *Proceedings of the 24th Workshop on the Semantics and Pragmatics of Dialogue - Full Papers*, Virtually at Brandeis, Waltham, New Jersey. SEMDIAL.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(2):169–190.

Mirco Ravanelli, Titouan Parcollet, Aku Rouhe, Peter Plantinga, Elena Rastorgueva, Loren Lugosch, Nauman Dawalatabad, Chou Ju-Chieh, Abdel Heba, Francois Grondin, William Aris, Chien-Feng Liao, Samuele Cornell, Sung-Lin Yeh, Hwidong Na, Yan Gao, Szu-Wei Fu, Cem Subakan, Renato De Mori, and Yoshua Bengio. 2021. Speechbrain. `https://github.com/speechbrain/speechbrain`.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2368–2378, Berlin, Germany. Association for Computational Linguistics.

Anthony Zhang. 2017. Speech recognition (version 3.8).

A Example User-Agent Interaction.

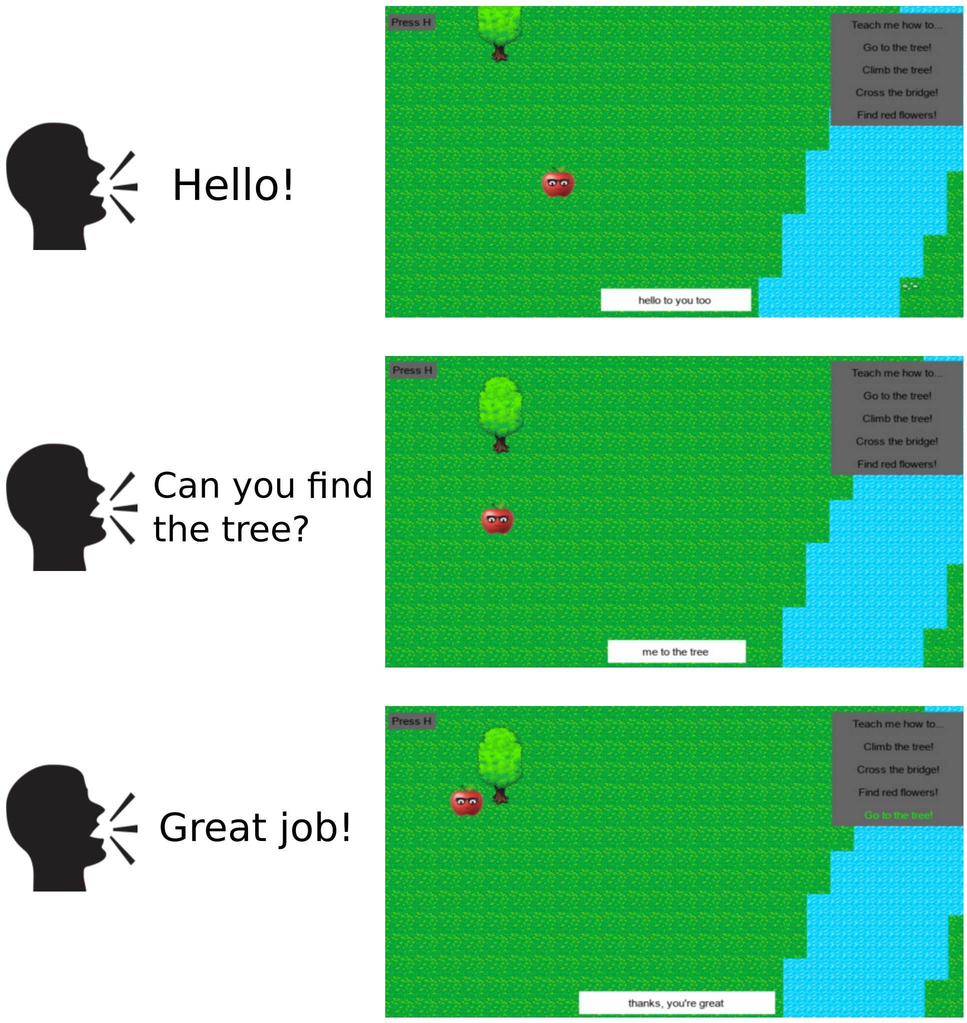
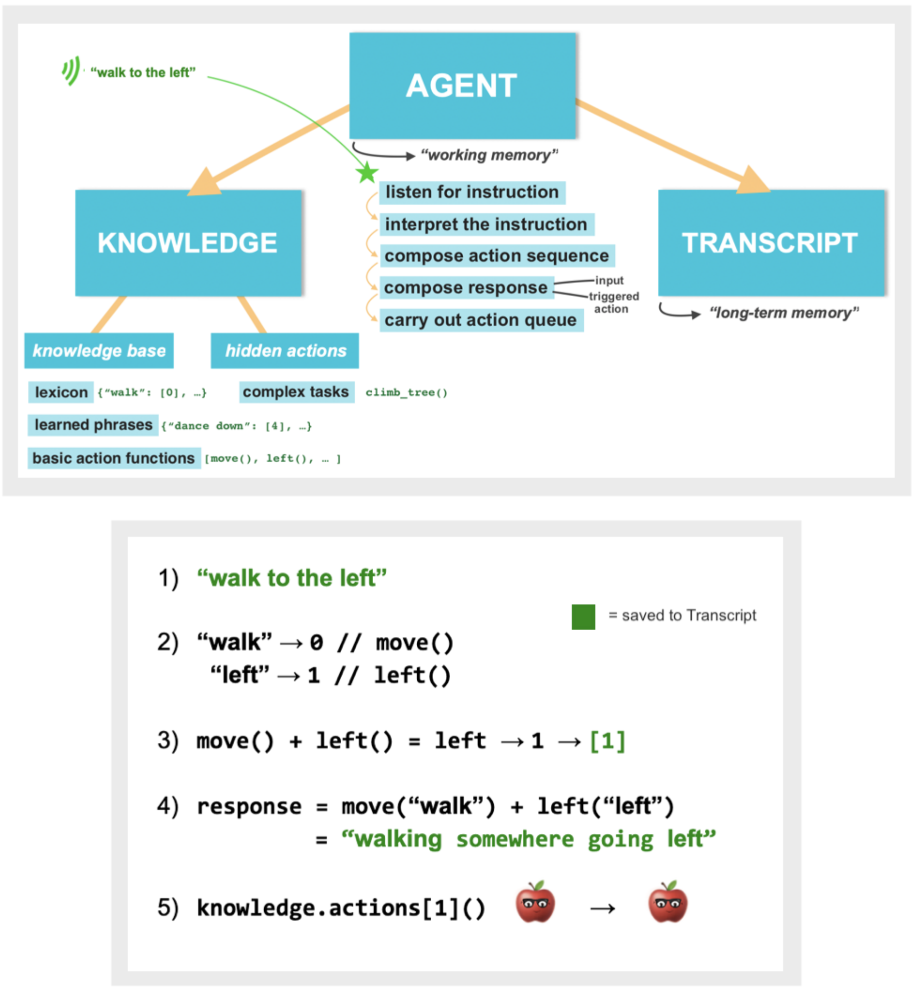


Figure 1: Example interaction.

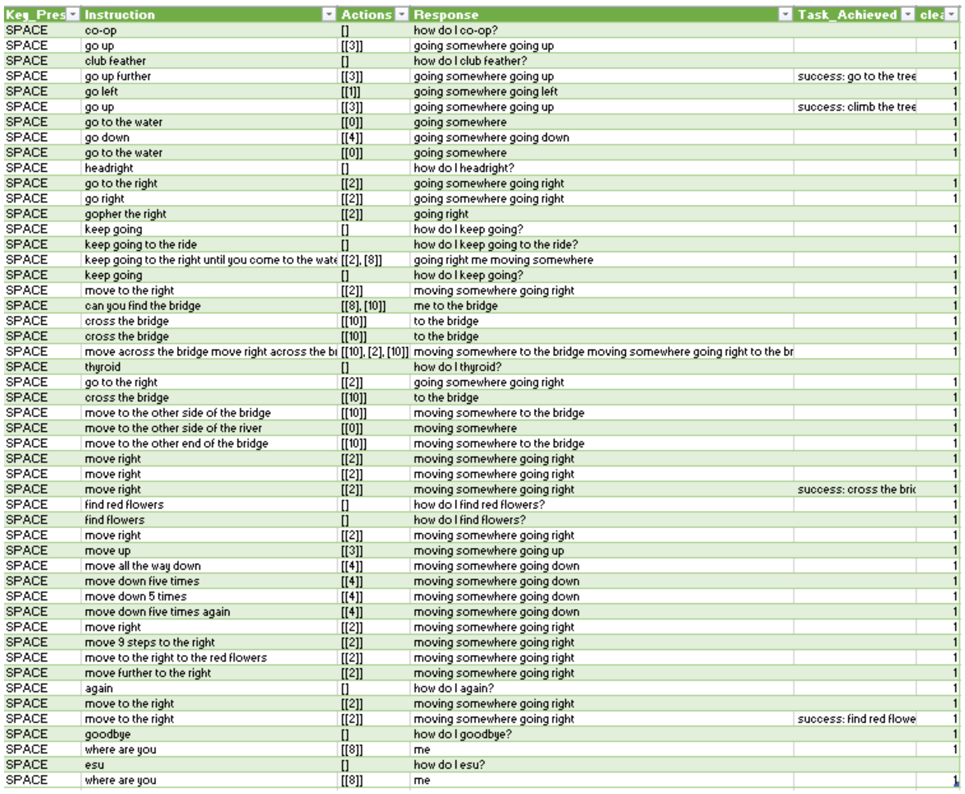
B Example Instruction Processing.



```
1)  "walk to the left"

2)  "walk" → 0 // move()
     "left" → 1 // left()

3)  move() + left() = left → 1 → [1]

4)  response = move("walk") + left("left")
                = "walking somewhere going left"

5)  knowledge.actions[1]()   →
```

Figure 2: Agent processing of the instruction *walk to the left*.

C Transcripts.



Key_Pres	Instruction	Actions	Response	Task_Achieved	clea
SPACE	co-op	[]	how do I co-op?		
SPACE	go up	[[3]]	going somewhere going up		1
SPACE	club feather	[]	how do I club feather?		
SPACE	go up further	[[3]]	going somewhere going up	success: go to the tree	1
SPACE	go left	[[1]]	going somewhere going left		1
SPACE	go up	[[3]]	going somewhere going up	success: climb the tree	1
SPACE	go to the water	[[0]]	going somewhere		1
SPACE	go down	[[4]]	going somewhere going down		1
SPACE	go to the water	[[0]]	going somewhere		1
SPACE	headright	[]	how do I headright?		
SPACE	go to the right	[[2]]	going somewhere going right		1
SPACE	go right	[[2]]	going somewhere going right		1
SPACE	gopher the right	[[2]]	going right		
SPACE	keep going	[]	how do I keep going?		1
SPACE	keep going to the ride	[]	how do I keep going to the ride?		
SPACE	keep going to the right until you come to the wate	[[2], [8]]	going right me moving somewhere		1
SPACE	keep going	[]	how do I keep going?		1
SPACE	move to the right	[[2]]	moving somewhere going right		1
SPACE	can you find the bridge	[[8], [10]]	me to the bridge		1
SPACE	cross the bridge	[[10]]	to the bridge		1
SPACE	cross the bridge	[[10]]	to the bridge		1
SPACE	move across the bridge move right across the bı	[[10], [2], [10]]	moving somewhere to the bridge moving somewhere going right to the br		1
SPACE	thyroid	[]	how do I thyroid?		
SPACE	go to the right	[[2]]	going somewhere going right		1
SPACE	cross the bridge	[[10]]	to the bridge		1
SPACE	move to the other side of the bridge	[[10]]	moving somewhere to the bridge		1
SPACE	move to the other side of the river	[[0]]	moving somewhere		1
SPACE	move to the other end of the bridge	[[10]]	moving somewhere to the bridge		1
SPACE	move right	[[2]]	moving somewhere going right		1
SPACE	move right	[[2]]	moving somewhere going right		1
SPACE	move right	[[2]]	moving somewhere going right	success: cross the bri	1
SPACE	find red flowers	[]	how do I find red flowers?		1
SPACE	find flowers	[]	how do I find flowers?		1
SPACE	move right	[[2]]	moving somewhere going right		1
SPACE	move up	[[3]]	moving somewhere going up		1
SPACE	move all the way down	[[4]]	moving somewhere going down		1
SPACE	move down five times	[[4]]	moving somewhere going down		1
SPACE	move down 5 times	[[4]]	moving somewhere going down		1
SPACE	move down five times again	[[4]]	moving somewhere going down		1
SPACE	move right	[[2]]	moving somewhere going right		1
SPACE	move 9 steps to the right	[[2]]	moving somewhere going right		1
SPACE	move to the right to the red flowers	[[2]]	moving somewhere going right		1
SPACE	move further to the right	[[2]]	moving somewhere going right		1
SPACE	again	[]	how do I again?		1
SPACE	move to the right	[[2]]	moving somewhere going right		1
SPACE	move to the right	[[2]]	moving somewhere going right	success: find red flowe	1
SPACE	goodbye	[]	how do I goodbye?		1
SPACE	where are you	[[8]]	me		1
SPACE	esu	[]	how do I esu?		
SPACE	where are you	[[8]]	me		1

Figure 3: Transcript file of a completed game.