LIA: A Virtual Assistant That can be Taught New Commands by Speech

Merav Chkroun and Amos Azaria
Department of Computer Science, Ariel University, Israel
meravgu@gmail.com, amos.azaria@ariel.ac.il

Abstract

In the imminent future, people are likely to engage with smart devices by instructing them in natural language. A fundamental question to ask is how might intelligent agents interpret such instructions and learn new tasks. In this paper we present the first speech based virtual assistant that can be taught new commands by speech. A user study on our agent has shown that people can teach it new commands. We also show that people see great advantage in using an instructable agent, and determine what users believe are the most important use cases of such an agent.

1 Introduction

Most machine learning based methods base their learning on large data-sets labeled manually, usually involving tremendous amounts of human labor. Labeling large data-sets may not be practical for a user wanting to teach her agent how to perform a specific task. Humans, on the other hand, usually learn by obtaining instructions in natural language. For example, a manager can teach her human assistant how to perform a very specialized task, just by *explaining* in natural language how one should perform the task, possibly accompanied by showing a single example. The human assistant will learn not only how to perform the new task, but also to perform any similar task.

Virtual assistants which have always been considered the future of AI, have shown great growth in the past several years, with the appearance of Apple's Siri, Google Now and Microsoft's Cortana. More recently we witness the emergence of home assistants such as Amazon's Echo and Google Home. These assistants are operated solely by voice commands, and can be operated from a distance. Voice is also used as a principal user interface in smart devices such as smart-watches and smart-glasses.

From the above we may infer that the ability to teach virtual assistants new commands by speech, should be an integral part of the future of AI. In this paper we present the first speech based virtual assistant that can be taught new commands by speech. Our Learning by Instruction Agent (LIA) is composed of many AI components, such as a learning and generalization mechanism, text summarization, and other components common in dialog systems such as voice activity detection, speech recognition, natural language understanding, natural language generation and a text to speech mechanism

[26, 6]. LIA uses our recently developed method for command execution and for learning new commands, as described in [3]. However, the agent described in [3] (which we will refer to as AKM), was text based, was not deployed on a mobile phone, and was extremely limited in the tasks it could perform (and thus was not useful for actual users).

LIA supports sending and receiving emails (by connecting to actual email accounts), as well as fetching news about a specific category or according to given keywords. LIA also supports many common virtual assistants' commands such as playing music, answering factoid questions, saying something, spelling out words, and setting a timer, an alarm or a reminder. If LIA does not understand a user utterance (e.g. "forward to Bob"), the user is asked whether she would like to teach LIA how to perform that task. The user can then explain in natural language, step after step, how to perform the task, and LIA collects all these steps and associates them with the given command. LIA also identifies what words might serve as parameters (e.g. the word "Bob") and generalizes the taught command to other similar commands (e.g. "forward to Tom").

To summarize, our main contribution in this paper is the development of an open sourced virtual assistant, that can be taught new commands by natural language speech. Our agent can be fully operated from a distance. We also ran a user study evaluating the performance of our agent as well as user reception towards an agent that can be taught new commands and determine what users believe are the most important use cases of such an agent.

2 Related Work

The idea of a virtual assistant that can be operated by natural language speech has long-lived in science fiction literature, such as HAL 9000 from "2001: A Space Odyssey" (1968), the Ship's computer in Star-Trek (1967), and The Brain from Escape! (1945). However, the first commercial virtual assistant did not emerge until 2011, when Apple came up with Siri [4] that could perform some tasks by speech commands. Siri, was an offshoot from the CALO project starting in 2003 [18] and is based on many years of research in the dialog system community [26]. Soon later, additional virtual assistants have appeared, including Google Now and Google Assistant, Microsoft's Cortana and Amazon's Echo. While some of these virtual assistants support APIs allowing developers to extend these virtual assistants capabilities, for the lame users these assistants are rigid and cannot be extended to supporting new tasks that they may require. This problem becomes more pronounced for commands that are very specific to a user, which developers may never support (due to the minimal demand).

In natural language programming [5], a programmer can use natural language to develop software. In Inform 7 [22], for example, a programmer can create an interactive fiction program using English sentences. For example, the programmer can say "The kitchen is a room," "The kitchen has a stove," and "The description of the stove is: 'very dirty' ". However, despite being in natural language, statements in Inform 7, are required to be in a specific form. Needless to say, that neither of these programming languages support teaching an agent new commands based on speech. In [9] we allowed users to teach a chatbot new responses in natural language (by using a specific

form). Quirk et al. [21] have converted "recipes" written in natural language to if-then statements, in which the "if" part is a condition on a sensor (e.g. phone camera, mic. etc) or a cyber-sensor (e.g. weather, Twitter etc.), and the "then" part is bounded to a command (e.g. opening the camera app, or making a sound) [20]. Huang et al. [13] have developed a system that uses the crowd to compose if-then recipes for mobile phones.

Learning by demonstration, also known as, Programming by demonstration or Imitation learning, is commonly used by robots interacting with users. The common case study is with a person showing a robot how to lift or select a certain object, move an object or perform some other tasks [1]. In most studies the human teacher actually moves the robot's arms to perform the taught task [7], or controls the robot using a control peg, while in some other studies, the human teacher performs the task in front of the robot's cameras [19], or wears data gloves [15]. In many cases the robot can also generalize beyond the specific training scenario, to perform the task also in different conditions. For example, Calinon et al., [7], teach a robot by demonstration how to move a chess piece on a board of chess by moving the robot's arms. The robot can then generalize and move the same piece also when it is located in a different location. In recent work [17], we have combined speech commands with programming by demonstration to execute different commands on a mobile app. Thorne et al. [25] use a system based upon programming by demonstration as an alternative to spreadsheet programming. They run an experiment with human subjects, and describe the benefits and limitations of this system.

Some virtual assistants include embodied agents [8, 27]. However, embodied virtual agents require heavy resources both communication and computation wise. Herrmann [12] compares different machine learning based methods for virtual assistants.

In previous work [3], we have presented an agent (we will term AKM) that can learn how to execute new commands by interacting with a user in natural language. We have shown that users who were not required to teach the agent new commands, do so voluntarily, and save time by executing these taught commands. However, AKM was limited to interaction by text using a web-interface, and was only able to execute commands in the domain of mock emails.

3 LIA

LIA is composed of a client side running on Android, and a Java based server side. Figure 3 shows screen-shots of a user interacting with the Android app. LIA listens for a wake-up phrase ('Jessica'), and then lights up, waiting for the user to give a voice command. The recording of the command is sent to the server side which, using an ASR service, is converted to text. The command is then processed and the client receives the response, which can either be text that is spoken back to the user, or something more complex such as opening an app. The user may also communicate with LIA using text (rather than speech). See `https://tinyurl.com/LiaDemo` for a demonstration of a user interacting with LIA. In this demonstration, LIA answers a factoid question, plays a song, sets a reminder, and learns a new command. See `https://tinyurl.com/LiaDemo1` for a demonstration of LIA learning a com-

plex new command in the email domain, and based upon its generalization method, executing this learned command with a different argument.

LIA uses the Combinatory Categorial Grammar (CCG) [24] parser to map user commands to logical forms, which can be handled and executed by the server. Using CCG is useful due to its tight coupling of syntax and semantics [28]. A CCG semantic parser is composed of a lexicon, a set of grammar rules, and a trained parameter vector. The lexicon maps words to syntactic categories, and species how each word can be combined with adjacent words and phrases during parsing to become both a new syntactic category and a logical form. The logical form represents the semantics of the sentence which later can be executed. For example, the word "set" is mapped to the syntactic category - ((S\PP StringV)/MutableField), the argument type appears on the right of the slash ("MutableField" in the example) followed by another argument to the left of the slash ("PP StringV" in the example) and the return type on the left ("S" in the example above). In addition, the word "set" is mapped to a logical form (lambda x y (setFieldFromFieldVal x y)), which defines the function that should be operated when parsing the word using the parameters from the previous map. The set of grammar rules correspond to standard function operations, such as application and composition. Our grammar also includes a small number of unary rules that represent common implicit conversions between types. The trained parameter vector (which was trained using machine learning) is used by the CCG parser to disambiguate multiple possible parses and decide which parse is the most relevant at a given context.

In order to support the learning of new commands, the semantic parser can parse any command into an "unknown command" (however, during the training process the parser learns not to parse commands into "unknown commands" if these commands can be parsed into any other logical form). When this happens, LIA asks the user whether she would like to teach LIA how to execute this new command. If the user acknowledges, LIA transfers to a learning state, and records all following commands provided by the user. When the user finishes the teaching process, LIA compiles all these commands into a logical form that is associated with the original command being taught. This logical form is added to the CCG lexicon. LIA includes a generalization method which allows the semantic parser to generalize the instruction to interpret other, similar commands, by identifying parts of the command which may be arguments. The algorithm learns which words in the taught command correspond to each part of the complete logical form. For example, if the user teaches LIA a command "Reply got it", LIA will understand that "got it" is an argument and later can execute the command "Reply see you then" correctly. The detailed logical form execution method and the lexicon induction algorithm (for learning new commands and generalize the commands) is described in [3]. In order to support all the new features, we have extended the original lexicon to 410 entries and the set of examples to 193 examples.

LIA supports the "undo" command. This was implemented using the Command design pattern and Java lambda expressions. Every time a command is executed, the inverse command is pushed into a stack (if applicable). If the user requests to undo the previous command, the inverse of the most recent command is popped out of the undo-stack and is executed. This allows the user to undo as many commands as she wishes. Tables 1 and 2 summarize the basic and complex commands currently supported by LIA, along with examples on how these commands may be used. Table 3 describes the

Feature	Example command
Playing YouTube songs	• "Play the piano man" • "Play the beatles"
Read news	• "Science news" • "Read news"
Set timer	• "Set a timer for 5 minutes with message go home" • "Tell me to go home in 5 minutes" • "Remind me to take the cake out of the oven in an hour"
Set alarm	• "Set an alarm for tomorrow at 7am"
Read timer/ alarm	• "Read timers" • "Read alarm"
Cancel timer/ alarm	• "Cancel timers" • "Cancel alarms" • "Cancel 6 am alarm"
Current time/date	• "What's the time?" • "What's the date?"
Factoid questions	• "Who is the president of the united states?" • "How many kilometers in a mile?"
Spell words	• "Spell elephant"
Undo previous command(s)	• "Undo"
Forget all learned	• "Forget everything"

Table 1: The basic commands currently supported in LIA.

ability to teach LIA new commands.

The detection of a wake-up phrase is based upon CmuSphinx [16]. It is important to note that LIA can listen to the wake-up phrase even when LIA is not the app in focus. Furthermore, LIA even listens for the wake-up phrase when the screen is off; the screen turns on as soon as LIA detects the wake-up phrase. This allows LIA to be operated from a distance. In addition, an external microphone was added to the phone running LIA, as well as an external speaker. Using both an external microphone and speaker required a mic-audio splitter (See Figure 1 for an image of LIA connected to both an external conference microphone and an external speaker, by the mic-audio splitter). These steps along with future development of basic commands such as the control of smart devices may allow LIA to serve as home assistant; LIA would be able to support smart home operation commands such as turning on and off the lights, controlling air conditions, electrical boiler etc. Furthermore, based upon these basic operations LIA

Feature	Example command
Read email	• "Read email"
Compose and send email	• "Compose a new email" • "Set the recipient to Dan's email" • "Set the body to Charlie is on his way" • "Send the email"
Say something	Useful when teaching new commands, for example: • "Thank you", "Say you are welcome" • "Bye", "Say goodbye, see you soon"
Defining new concepts, instances and fields	Users can define new concepts, create new fields, create new instances, get concept information. For example: • "Define concept person" • "Person has a favorite song" • "Becky is a person" • "Beckys favorite song is rolling in the deep" • "Play Beckys favorite song"

Table 2: Complex commands currently supported in LIA.

can learn to compose complex commands such as turning on the oven while increasing the air conditioning fan strength.

While, LIA allows the user to change the wake-up phrase to any phrase including a word (or several words) out of the CmuSphinx dictionary (approx. 133K words), finding the default wake-up phrase, turned out to be a complex task. We had to find a phrase that could be detected when called by users with different accents, not be invoked when not being called and sound like a female name (in order to match the text to speech component). Out of tens of wake-up phrases we have considered, 'Jessica' turned out to work best when tested against an initial group of four subjects with different accents (not included in the user study). When LIA asks the user a question (i.e. a sentence ending with a question mark), the user does not need to say the wake-up phrase again, but can naturally respond.

Unfortunately, Android allows only a single app to access the microphone. Since we want to have full access to the actual voice signal, our only option was to send a recording to our server and access an ASR service from our server (rather than accessing an ASR server directly from the phone). Another issue that came up due to this same exact limitation of Android, is, that while LIA is listening for a wake-up phrase, the user cannot use Skype or WhatsApp voice communication, or use Google Now. Fortunately, LIA listening for the wake-up phrase does not affect regular phone conversations. For this reason, we added a button on the GUI which causes LIA to release the microphone and not listen for the wake-up phrase. Even when LIA is not listening for the wake-up phrase, the user can still click on the talk button and give LIA a voice command.

While listening for a command, LIA needs to decide when the user finished giving

There are two options for teaching LIA new commands:

1. Say something LIA doesn't know how to execute and then say yes when she asks if you would like to teach her.
2. Say, for example "I want to teach a new command", and then give the command.

- In the following example a user teaches LIA to greet a person followed by playing his or her favorite song (assuming the person has a favorite song defined):
  - User: "Greet Caroline"
  - LIA: "Would you like to teach me?"
  - User: "Yes"
  - LIA: "What should I do when you say for example 'Greet Caroline'?"
  - User: "Say hello Caroline, here is your favorite song"
  - LIA: "What should I do next?"
  - User: "Play Carolines favorite song"
  - LIA: "What should I do next?"
  - User: "That's it"
  - ...
  - User: "Greet John"
  - LIA: "Hello John, here is your favorite song"
  - LIA: [Playing John's favorite song]

Table 3: Learning by natural language.

a command. In order for LIA to decide that the user has said a complete utterance, we have determined several threshold which turned out to work well in practice. LIA requires at least 200 milliseconds of speech, and an ending silence of at least 320 milliseconds. If LIA detects additional speech before it gets the response from the ASR, it will revoke the ASR request and continue to listen to the user until it detects another 320 milliseconds of silence. This allows LIA to have a fast response if the user has actually completed her utterance, while still listening for the full utterance. While detecting whether the user is talking or not is a classical voice activity detection (VAD) problem [23], in practice, all VAD methods we have tested require at least several seconds of a baseline signal. Unfortunately, the whole command is only a few seconds long, and therefore VAD methods we have tested did not seem to perform well. Instead we apply a low-pass-filter (with a window of 500) on the signal, and treat any sound that has an amplitude less than 1.4 times the lowest sound amplitude in the last 0.5 seconds plus some constant, as silence. We use a slightly higher threshold for determining speech.

LIA uses the Google ASR (Automatic Speech Recognition) API to convert the

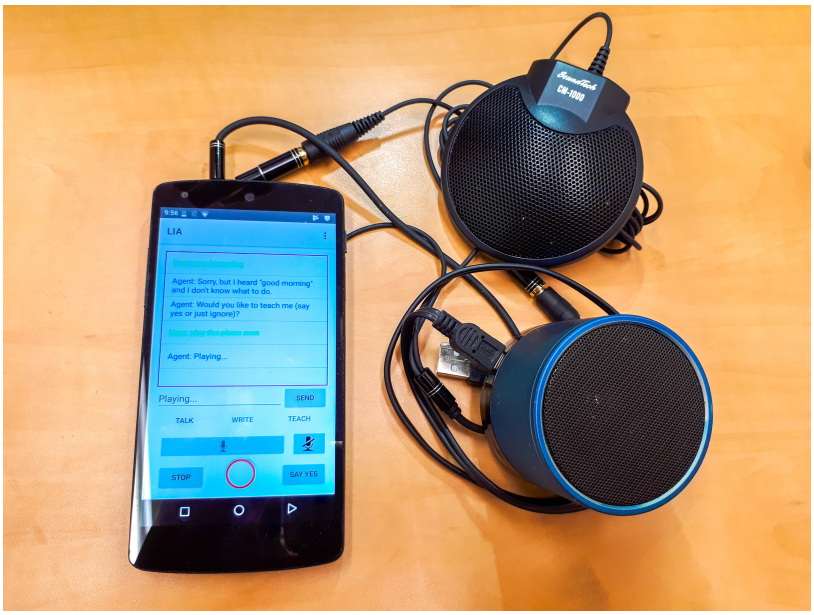

Figure 1: A photograph of LIA connected to the microphone and speaker.

users speech into text. Once the user has finished to speak, the ASR processes the sentence and returns text which it believes is a best fit to what has been said. The ASR also returns a list of candidates (alternatives), that were given a lower score, but might still match the user's speech. For example, given a user speech command, the ASR returned the following response structure: "get the weather", confidence:0.96787101. alternatives: "check the weather", "what the weather", "the weather", "I get the weather". LIA's server side first tries to parse the initial response obtained from the ASR. If this parse returns a command, it is executed, however, if this parse returns the "unknown command", LIA tries to parse the following results in the order of their appearance in the returned ASR candidate list. If LIA can parse any of these candidates to a command (other than the "unknown command") this command is executed. However, if all parses result in an "unknown command", the command is considered to be unknown, and the user is asked whether she would like to teach LIA how to execute the command, using the initial response obtained from the ASR.

LIA uses several APIs to provide content. LIA uses the YouTube API to find a requested video or song, the guardian API to provide news, the Aylien API to summarize this news and the Worlfram Alpha API to answer factoid questions. LIA also support composing and reading real emails. Email capabilities in LIA can be invoked on-demand, that is, the first time the user tries to compose an email or read an email, LIA replies that this action requires obtaining the user's email and password, and the user is prompted to provide their user email and password. The password is securely saved on the server using the AES algorithm [10], based upon the user's phone identifier and a random number (known as salt). The next time the user requires email

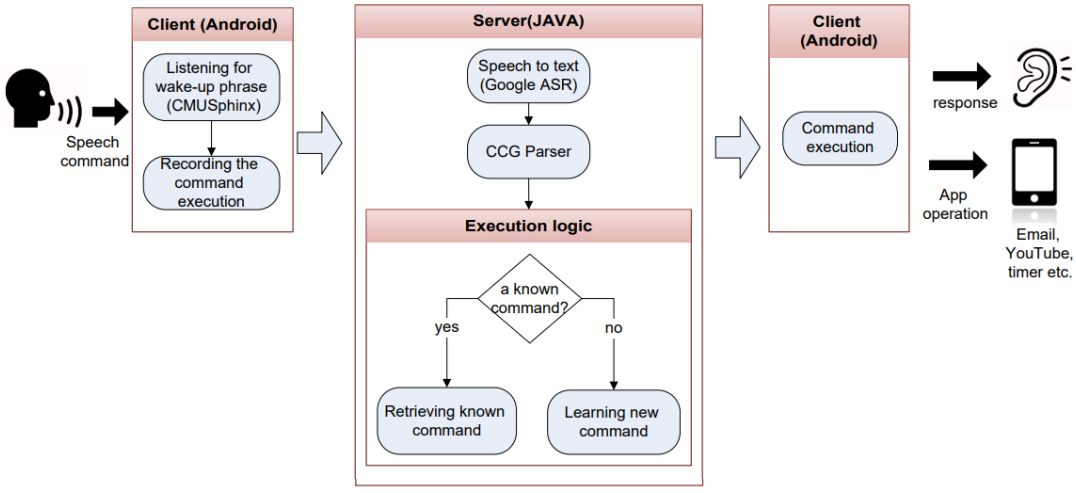

Figure 2: A diagram of the whole process LIA go through.

capabilities the user is not required to provide her password again, but the server uses the user's phone identifier in order to obtain the user's email address and password.

Our initial efforts of using Olympus [6] as a dialog framework were not very successful, as we had to replace many of the components in Olympus, and eventually, found it simpler to write the whole system ourselves. LIA is open source and all code of both the client and the server is available on GitHub. LIA is deployed as a beta application on Google Play. Figure 2 presents an overview of LIA.

4 Experimental Evaluation

In order to evaluate LIA we recruited 15 subjects, of which 6 were male and 9 where female. The subjects' ages ranged from 18 to 49, with a mean of 35.8 and median of 36. All subjects live in Israel and have a good level of English. We hypothesized that the subjects would be able to interact with LIA by natural language speech and, more importantly, teach LIA new commands. Another purpose of experimenting with human subjects was to evaluate whether human subjects appreciate the importance of generalization of taught commands.

Each subject received a brief description of LIA (in a few sentences), and got a short read-me file describing the abilities of LIA. The subject then received a mobile phone with LIA installed on it and was asked to complete 18 tasks. The first task was to ask LIA to forget everything she has learned from previous subjects (to allow the subject to operate on a fresh version of LIA). The following is the list of tasks that the subjects were required to execute with LIA:

1. Ask LIA to forget everything.

2. Ask LIA to Play “strawberry fields forever”.
3. Think of something you want to hear or watch and ask LIA to Play it.
4. Ask LIA: “who is the president of the united states?”
5. Ask LIA another factual question.
6. Ask LIA for the time or date.
7. Ask LIA to remind you something in 30 seconds.
8. Ask LIA to remind you something else.
9. Ask LIA to read your email.
10. Define a new concept “contact” with fields “phone” and “email”:
    (a) “define concept contact”.
    (b) “a contact has an email”.
    (c) “a contact has a phone number”.
11. Define a new instance of the concept contact, for example:
    (a) “*Kate* is a contact”
    (b) “*Kate’s* email is kate@kate2.com”
    (c) “*Kate’s* phone number is 123456789”
12. Ask LIA what is the email of the instance you created.
13. Send an email using LIA.
14. Teach LIA that when you say “good morning”, she should say “have a wonderful morning”.
15. Tell LIA “good evening”.
16. Tell LIA “good girl”.
17. Teach LIA something you want her to do / say.
18. Check if LIA succeeded to generalize the command you taught her.

The tasks presented to the subjects were composed such that they gradually build up task complexity, to allow the subjects a gradual learning curve. For each of LIA’s abilities the tasks start out very specific, with later tasks being more general, with some tasks including examples which make redundant the need to use specific tasks. All examples were already a part of LIA’s instructions (readme file). For example, task 2, “Ask LIA to Play ‘strawberry fields forever’ ”, was specific, therefore, the following task, “Think of something you want to hear or watch and ask LIA to Play it”, was general. Table 4 summarize the different types of tasks the subjects were requested to complete. Note that the subjects were introduced to LIA’s ability to learn new commands and generalize them in tasks 14 through 18 (5 tasks in total), therefore, they had a good level of understanding the concept of generalization when performing task 18.

Upon completion of all the tasks, the subjects were asked to complete a four parts questionnaire regarding their experience with LIA. On the first part they were asked to rate each and every feature they tried (playing music on *YouTube*, asking factual questions, setting an alarm or a timer, defining new concepts and instances, reading

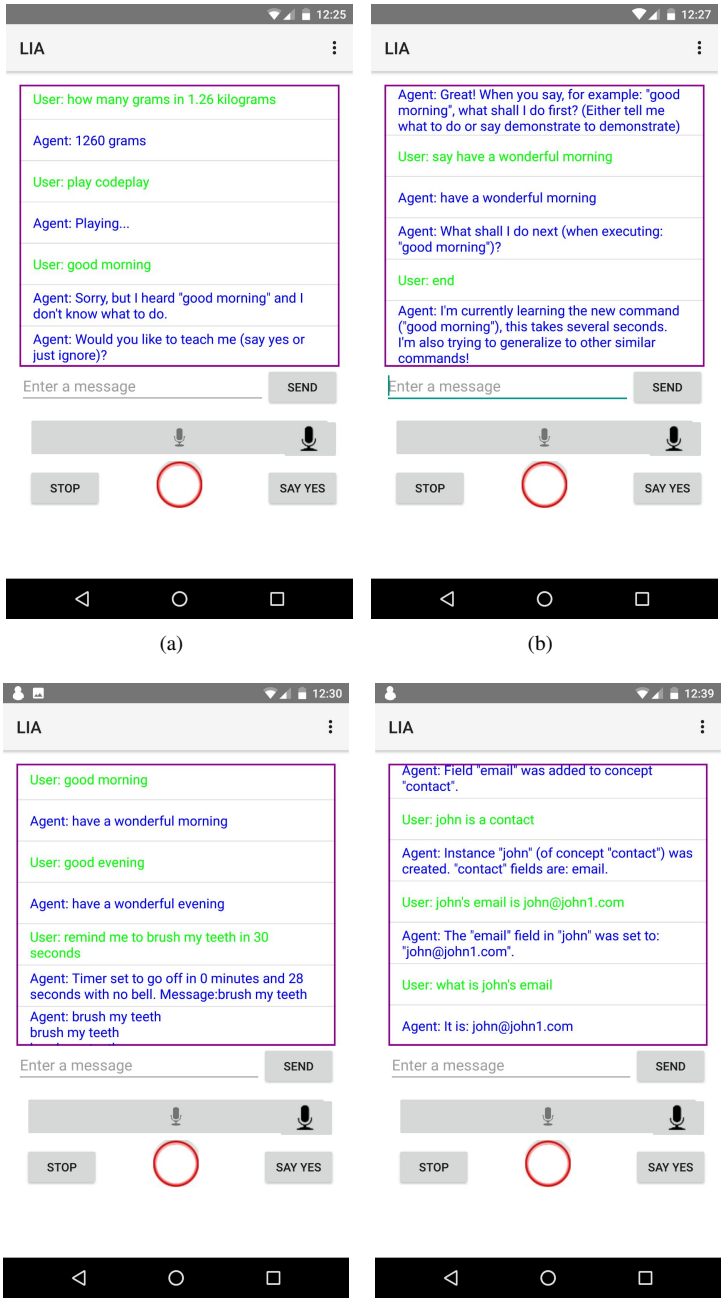

Figure 3: A screen-shot of LIA during a conversation with a user: (a) The user asks a factoid question and gets an answer; asks to play a YouTube video; and gives a command that LIA is not familiar with. (b) The user teaches LIA the new command. (c) The user uses the generalized newly taught command, and asks to set a reminder. (d) The user defines a new concept with the field 'email'; defines a new instance (John); and asks to retrieve John's email.

Task type	Tasks numbers
Specific tasks	1, 2, 4, 7, 14, 15, 16
General tasks	3, 5, 6, 8, 9, 12, 13, 17, 18
Tasks with examples	10, 11

Table 4: Different types of tasks the subjects were requested to complete.

and sending emails and teaching LIA new commands). The rating score ranged from 1 to 5 on a Likert scale (1- Very disappointed, 2-Disappointed , 3-Neither disappointed nor satisfied , 4-satisfied ,5-very satisfied).

The second part of the questionnaire focused on the ability of LIA to learn new commands. The subjects were asked to determine what their motives for teaching LIA new commands were .They were asked to choose the most appropriate statement from the list below:

1. I don't think I would be teaching LIA any new commands.
2. I would teach LIA new commands just to check if and how it works.
3. I would teach LIA some fun commands.
4. I would teach LIA new commands in order to give short aliases to longer commands.
5. I would teach LIA new commands in order to automate longer tasks (composed of several commands).
6. I would teach LIA new command in order to giving aliases to commands that can only be typed in (e.g. commands in a foreign language).

The subjects also had to indicate whether LIA succeeded to generalize the commands they have taught. Two more questions in this part checked the subjects level of willingness to share new commands they had taught the agent with others, as well as their willingness to use commands that LIA learned from others.

On the third part of the questionnaire, subjects were asked to assign their level of agreement with each of the following statements on a 7-point Likert scale (strongly disagree, disagree, slightly disagree, neither agree nor disagree,slightly agree, agree, strongly agree):

1. LIA is smart
2. LIA understood me
3. I would like having LIA on my phone
4. An agent that can be taught new commands is useful.

The first 3 statements (LIA is smart, LIA understood me and I would like having LIA on my phone) are identical to the statements that appear in [3], which were evaluated on a 7-point Likert scale as well. In order to fully understand the attitude the users had towards generalization of commands, we asked them to rate two additional statements:

Criterion	LIA	AKM
Completion rate	**80%**	41%
Parse failure	28%	**15.4%**
Execution error	5.4%	5.4%
"LIA is smart"	5.0	5.0
"LIA understood me"	4.33	**5.5**
"I would like to have LIA on my phone"	**4.73**	4.6

Table 5: A comparison between LIA (this paper) and the text based agent from [3] (AKM).

(1) Generalizing commands to other similar commands is an important feature (2) I only care about the execution of the command I taught the agent (and don't care about other similar commands).

In the last part of the questionnaire the subjects were asked about their acquaintance with programming. The subjects could choose one of the following: None or very little; Some background from high-school or equivalent; Some background from college/university or equivalent; Bachelor (or other degree) with a major or minor in software, electrical or computer engineering or significant knowledge from other sources. We assigned numbers to each of the options, resulting in a numeric measure of acquaintance with programming from 1 to 4.

5 Results

Most of the subjects (80%) completed the full list of tasks described above (in comparison to 41% with AKM). Two subjects encountered difficulty in the tasks related to concepts: defining a concept and creating an instance of a concept. Another subject failed the last two tasks of the experiment, and did not teach LIA a new command. The average completion time was slightly under 33 minutes (32.6). The fastest subject has completed the experiment in only 20 minutes while the longest experiment lasted 62 minutes. Table 5 presents a comparison on different criteria between LIA and AKM.

As for the success rate of the parsing and execution, out of 798 commands, 28% (228 commands) were parsed to an unknown command, which in most cases is a parse failure (in comparison to 15.4% in AKM). An additional 5.4% (43 commands) resulted in an execution error. For example, when defining a concept that already exists or setting an email to a non-email value (in comparison to 5.4% in AKM)

We evaluate the performance of LIA on the following five categories:

1. General features (common in current virtual assistants): Playing music on *YouTube*, asking factual questions and setting an alarm or a timer.
2. Concepts: Defining new concepts, defining fields and creating new instances.
3. Complex features: Reading and sending emails (these features may require additional definitions such as defining a concept named contact, and creating an instance of this contact).

4. New commands: Teaching LIA new commands including the method used to generalize these commands, and sharing commands with other users.
5. Overall user satisfaction: The user rating as obtained from the third part of the questionnaire.

Feature	Result
Play music on YouTube	4.6
Ask LIA factual questions	4.13
Set alarm or timer	4.13
Define new concept	2.93
Read and Send emails	3.07
Teach LIA new commands	3.67

Table 6: User scores of different features of LIA.

Table 6 presents the average scores given by the subjects to the different features of LIA. LIA was very successful in terms of General features, with playing music on *YouTube* getting 4.6 points on average , asking LIA factual questions getting 4.13 points on average, and setting an alarm or a timer achieved 4.13 points on average. Furthermore, 80% of the subjects found the feature of playing music on *YouTube* one of the most useful feature. 60% of the subjects found the feature of setting an alarm one of the most useful feature. 47% of the subjects found the feature of asking factual questions one of the most useful feature (each subject could chose more then one feature as the most useful feature).

In the category of Concepts and that of complex features, LIA did not perform as well. Defining new concepts, defining fields and creating new instances obtaining only 2.93 points on average. This was the only feature with an average suggesting that the subjects were somewhat disappointed. Reading and sending email receiving only 3.07 points on average. In addition, not even a single subject mentioned defining new concepts as one of the most useful features of LIA, and only 26% of the subjects noted reading and sending emails as one of the most useful features.

LIA has shown moderate success in the most important category of teaching LIA new commands, with an average user score of 3.67 points. We note that only 13% of subjects said that if they would use LIA on daily basis they don't think they would be teaching LIA any new commands; with the rest of the subjects, 87%, saying that they would teach LIA new commands for some purposes: 27% said that they would use LIA to automate longer tasks, 20% would teach LIA some fun commands, another 20% would teach new commands in order to give aliases to commands that can only be typed (for example, commands in a foreign language), and the rest, 20%, would only to check if and how teaching new commands works.

Another important feature (not currently supported by LIA) is sharing new commands between different users of LIA. The subjects have shown great interest in sharing commands they taught LIA with other users, giving such a feature a score of 4.4 points. In addition, the subjects were interested in using commands that others have taught LIA and gave such a feature a score of 4.53 points.

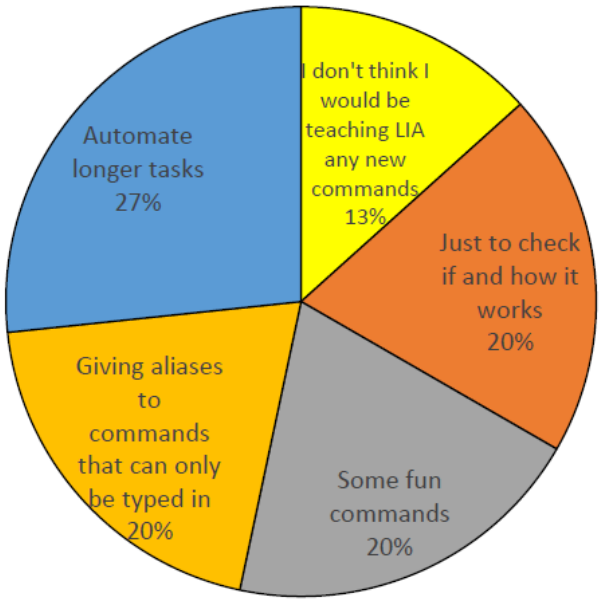

Figure 4: Users' different motives for teaching LIA new commands.

As for the overall user satisfaction questions (which used a $1-7$ Likert scale) the subjects seemed to slightly agree that LIA was smart, giving it 5.0 on average (interestingly, this same question got an average score of 5.0 with AKM as well); the subjects gave LIA 4.33 with respect to whether they thought that LIA understood them (in comparison to 5.5 with AKM). The subjects gave LIA a score of 4.73 with regards to whether they would like to have LIA on their phone (in comparison to 4.6 with AKM). The subjects seemed to agree that an agent that can be taught new commands is useful, giving this question a very high score of 6.13 on average. The subjects also expressed the importance of generalization by agreeing with the statement on generalization and giving it a score of 5.8 while disagreeing with the counter statement (which claims that they only care about the command that they taught and not about other similar commands) and giving that statement only 2.87.

	Average time per single task	Standard deviation	% of all tasks	Normalized % of all tasks
Specific Tasks	1.33	0.80	28.0%	19.1%
General Tasks	1.80	1.46	48.7%	25.7%
Tasks with examples	3.87	0.28	23.3%	55.3%

Table 7: Average completion time per single task, standard deviation of the average completion time of the tasks in a given type, the time each task type took in proportion to the total time of the experiment, and this proportion when accounting for the number of tasks in each type.

	Average time	Standard deviation	% of all tasks
Task 1	2.57	1.80	7.7%
Task 2	1.03	0.89	3.1%
Task 3	0.80	0.63	2.4%
Task 4	0.90	0.46	2.7%
Task 5	0.88	0.87	2.6%
Task 6	0.84	0.49	2.5%
Task 7	0.98	0.92	2.9%
Task 8	1.04	1.38	3.1%
Task 9	2.01	1.29	6.1%
Task 10	4.07	2.89	12.2%
Task 11	3.67	2.92	11.0%
Task 12	0.98	0.84	2.9%
Task 13	4.80	2.60	14.5%
Task 14	2.33	0.98	7.0%
Task 15	0.43	0.42	1.3%
Task 16	1.04	0.79	3.1%
Task 17	3.67	3.54	11.0%
Task 18	1.16	0.71	3.5%

Table 8: Average completion time per task, standard deviation of completion time of each of the subjects, and the time each task took in proportion to the total time of the experiment.

Table 7 summarizes the average time it took the subjects to complete the tasks of each of the three types (that is, specific tasks, general tasks, and tasks with examples). Table 8 provides additional details on each of the tasks. As expected, the general tasks took longer, on average, than the specific tasks. However, the tasks with examples took even longer to complete. This is because the tasks with examples were harder in the first place and therefore were accompanied with examples in order to slightly simplify them. The high standard deviation of the general task type, suggests that some of the general tasks took much less than others. For example, while task 3 (Think of something you want to hear or watch and ask LIA to play it) took less than one minute on average, task 13 (composing and sending an email) which is of type general task as well, took nearly 5 minutes on average. The first task, despite being a very specific task (ask LIA to forget everything), took much longer than average. This is expected, as being the first task the subjects had to perform, this task required them to become acquainted with LIA as well as activating her by the wake-up phrase. We would also like to note the very large standard deviation of task 17, which was teaching LIA new commands, as two subjects took over 10 minutes to complete this task, while five subjects took only slightly over one minute.

Table 9 compares measures of subjects who have completed the entire experiment successfully with those of subjects that have failed at least one of the tasks. All differences between the two groups are small and not statistically significant. As can be seen in the table the average completion time for subjects who have successfully completed all tasks is slightly lower than those who did not complete all tasks. However,

these differences are not statistically significant, as there were 4 types of subjects. The first type included subjects that when they encountered any communication difficulties, gave up and moved on to the next task. The second type of subjects, did not give up that soon and spent time trying to accomplish the tasks (some of these subjects have eventually succeeded). The third type of subjects communicated fluently with LIA and tried to complete the experiment as fast as possible. The last type, which is in fact the most interesting, includes subjects that did not encountered any difficulties in communicating with LIA, but still took longer to complete the experiment. These subjects were very enthusiastic with the experiment and enjoyed trying more commands, and therefore took longer.

We found a weak correlation (-0.24) between the average rating of the three overall user satisfaction questions ("LIA is smart", "LIA understood me", "I would like having LIA on my phone") and the subject completion time. This weakly suggests that subjects that took less time to complete the experiment, rated LIA higher. We believe that, as mentioned above, this correlation is not very large because some subjects who enjoyed interacting with LIA did so for a longer period of time. We found nearly no correlation (-0.05) between the subject completion time and the average feature scores. However, when comparing the scores that the subjects gave each feature with the time it took each subject to complete tasks related to that specific feature, we found a moderate correlation of -0.37. This suggests that, in general, subjects that were faster at completing tasks of a specific feature, gave that features a higher score.

	Succeeded (80%)	Failed (20%)
Average total time (Standard deviation)	31.67 (8.72)	36.33 (22.52)
Average rating tasks types	3.83	3.72
Average rating LIA	4.64	4.89

Table 9: A comparison between subjects who had failed at least one of the tasks of the experiment with subjects who had completed the entire experiment successfully.

Recall that we had two hypotheses:

1. The subjects would be able to interact with LIA by natural language speech and, more importantly, teach LIA new commands.

2. Human subjects appreciate the importance of generalization of taught commands.

The results presented above confirm our hypotheses. All subjects have interacted with LIA by natural language speech, 80% of the subjects have completed the full task list and 93.3% of the subjects have successfully taught LIA new commands (which they have came up with themselves). The subjects also highly agreed that learning new commands is useful. The results also indicate that subjects realize that generalization of new commands is important and useful.

We would like to mention a subject that was very enthusiastic about the experiment. This subject taught LIA a complex new command. She taught LIA that when she says 'goof' LIA should do all the following: (1) Play a song from *YouTube* ; (2) Set a reminder to take a break in 30 seconds; (3) Read an email; and (4) Say 'have a

wonderful day'. After defining this command, the subject tried it out and was excited to see it all works. Teaching this command took her much longer than the average subject (8 minutes). This duration is reasonable considering the level of the complexity of the command she had taught. While all subjects have taught LIA new commands, most other subjects did not combine several abilities of LIA in their new command.

6 Discussion

Since no other system supports teaching new commands by natural language speech, there was no system that could have been directly compared to LIA. Therefore, we compare the results in this paper with the results from [3] (AKM), in which users could teach the system new commands by text. This comparison has its limitations, as the communication with LIA is by speech, while communication with AKM is by text. Another major difference between the two systems is that the AKM supported only email related commands (and thus the experiment focused solely on email communication), while LIA also supports common virtual assistants' commands such as playing music, answering factoid questions, saying something, spelling out words, and setting a timer, an alarm or a reminder (and thus the experiment focused on each of these tasks). Nevertheless, the groups of subjects in the two experiments appear similar in terms of gender (46.7% males and 53.3% females in LIA; 51.2% males and 48.8% females in AKM), and age (with an average of 35.58 in LIA and 35.1 in AKM; $p = 0.8$ in a T-Test comparison).

As can be seen in Table 5 LIA outperformed AKM in both the completion rate and the user rating for the statement of "I would like to have LIA/the agent on my phone". This shows that LIA is more user-friendly than AKM. Interestingly, LIA and AKM had the exact same execution error. We believe that keeping the same rate of execution error in LIA as in AKM, despite the additional complexity level added by dealing with actual speech and additional domains presents the robustness of the system. However, AKM outperformed LIA in the parse failure rate and in the user rating for the statement "LIA/the agent understood me". We believe that this is a result of the additional complexity level added by dealing with actual speech, rather than only text in AKM, for example, there were many cases in which the ASR has failed, mixing 'as' with 'has' or 'John's email' with 'Jones email' etc. There were also failures caused by a false wake-up phrase detection, or the users saying the wake-up phrase more than once. These results may also be attributed to the significantly broader set of tasks LIA can handle, in comparison to AKM which was limited to the email domain, making it much easier for AKM's parser to understand the user, and for the *user* to learn how to effectively communicate with AKM.

Interestingly, there was no correlation (-0.098) between the level of acquaintance with programming and the time it took each subject to complete all tasks. In [3], where a weak correlation was found, we hypothesized that such a correlation may be attributed to typing speed. Not finding this correlation in current work (in which interaction was done by speech), strengthens this hypothesis. There was, however, a weak negative correlation (-0.28) between the level of acquaintance with programming and the ratings that the subjects gave LIA. We believe that this is because subjects with a

higher level of acquaintance with programming may be less tolerant towards bugs and may expect a flawless system.

As stated in the results section, creating concepts, fields and instances received a relatively low rating. We believe that this is because this feature appeared to be more complex for the subjects to master. This could be because that understanding the idea of concepts, fields and instances is non-trivial. We believe that had the subjects received additional examples and explanation on how to use these concepts and fields, they would have rated this feature higher. Note that teaching new commands obtained a much higher ranking, likely because it seems more intuitive to the subjects.

One might be concerned by the fact that in the experiment, the experimenter used specific words and phrases when instructing the subjects to communicate with LIA. We would like to note that many of the virtual assistants like Siri, Google Now, Microsoft's Cortana and Amazon's Echo show the users example commands and suggest new commands. For example, users of amazon echo get a newsletter including a list of commands the user can teach (about 20 examples commands). Nevertheless, LIA can understand a variety of expressions and word order of commands given by users. For example, all the following expressions "Set subject to hello", "Put hello in the subject" , "The subject is hello", "Fill the subject with hello" result in the same execution, that is, setting the subject of an email to the value "hello". Furthermore, if the user phrases a command differently (in a way that LIA does not understand), the user can teach LIA that this is a paraphrase of a command LIA already knows, and next time the user will be able to use her own phrasing of the command.

Currently it seems that the main limitation of LIA is that it requires acquaintance with the agent to become useful, and for the user to feel that LIA actually understands her. One of the subjects explicitly commented, that the experiment should have been much longer in order to allow her to fully understand LIAs capabilities and take advantage of them. In an attempt to mitigate this issue and reduce the required learning curve we intend to implement a conversational interface [14]. Conversational interface is a combined UI that allows users to interact with the system by speech integrated with graphical UI elements such as buttons, images, menus, videos, etc. This combination allows users to obtain a more holistic experience of the system. We therefore intend to add a text interface that allows users to complete blanks in pre-written commands, for example, "play _____", "set _____ to _____", "set a _____ minutes timer", add "yes" and "undo" buttons etc. This way, the users will see some of the commands currently supported by LIA, and should be able to execute them easily by text. Hopefully, they will slowly migrate to using voice commands.

It is hard to overestimate the importance of this work. If LIA gains control of many additional actions, LIA would allow anyone to contribute to a global repository of commands taught all through the world. People could use LIA assuming that it can execute any command, but in a case that LIA fails, the user may simply teach it how to execute that command.

7 Conclusions

In this paper we present the first speech based virtual assistant that can learn new commands by instruction. We build on an earlier version of our text based agent from [3], and have added support to a speech interface, as well as many different features, some of which are common to virtual assistants (such as setting a timer and playing music). A user study that we have ran shows that people are excited about a personal agent that can learn new commands, and about the idea of sharing these taught commands with others. We also determine what users believe are the most important use cases of such an agent. We have deployed LIA to a beta community on Google Play, and have 10 active users. We intend to release LIA to production in the near future. We intend to add support to sharing commands and learn what type of commands users actually teach LIA and share with each-other.

8 Future Work

LIA deployment opens up many significant opportunities for future work.

Conditional execution: We are considering adding to LIA the ability to learn conditional execution. For example, a user may want to teach LIA the command "safe send", with an expectation from LIA to first read the email and then ask if it should send the email (sending the email only if the user has replies "yes"). This requires LIA to support different execution threads while learning, depending on the user response. For example, after teaching LIA to read aloud the email, LIA may ask what should be done next. The user may then say "ask me if I'm sure", LIA would then ask "what should I do if you say yes?", and the user will give the final command. In run-time, LIA will execute the final command only if the user replies "yes". LIA would support not only yes/no conditions, but also requests like "ask whom should I forward this email to", LIA would then reply "suppose you answer John, what should I do next?" and then use "John" as an argument. Adding support for conditional execution in LIA would turn LIA into a platform for creating dialog systems in natural language.

LIA-light: Our experiments indicate that some of our users find the most value in teaching LIA fun commands. For example teaching it that when asking who is the smartest girl in the world, it should answer "Alison". Therefore, we are working on a light version that only deals with responses that aren't tied to execution commands. We want to allow users to teach such simple interaction commands by saying a simple sentence, instead of the current teaching process. For example, if the user would like to teach the command above, she could simply say: "when I say who is the smartest girl in the world, then answer Alison". We believe that LIA-light can be useful not only as an app on a mobile phone, but also be placed at the core of a toy, such as a robot toy or an artificial parrot etc. that can be taught new responses by using natural language (without a need for a graphical user interface). Sharing taught commands will be a core feature of LIA-light (and would be much simpler to implement than in LIA). Sharing and deletion of specific commands should be simpler in LIA-light.

Concept learning by dialog: We are exploring a method for learning concepts (such as spam mail) by dialog. After the user gives some explanation on the concept,

if the agent cannot fully understand the whole explanation, it may identify some parts of the sentence that it does not understand and ask follow-up questions only on that part. Once it gets an answer to this follow-up question it may either decide to further drill-down into a part of the follow-up answer, or move back up to the higher level. We would like to show that by using a dialog, the agent can learn better from human teachers.

Implicit feedback: We would like to improve LIA's performance without requiring the user to explicitly mark whether an execution is correct or not. We consider two different methods of detecting a user-feedback and two different mediums. The first method is by analyzing the current command, and the second method is by comparing the current command to previous command. Each of these options can be done on verbal (actual words) or voice (acoustics). Namely, we consider the following four options: 1. Verbal, current sentence: for examples commands like: "No, set the subject to hello". 2. Verbal, sentence comparison: for example, if the first sentence is "set the recipient to mom", and then the second sentence is "set the recipient to mom's email", the distance between these two sentences should be small. The distance between sentences can be measured using different word embedding methods (e.g. word2Vec). 3. Voice, current sentence: we believe that when one is not satisfied with LIA's response, she will react with a different tone of voice. 4. Voice, sentence comparison: we will use different methods to compare the current signal with the previous signal. This can be especially useful in cases where the wrong command was executed due to a fault in the ASR. (e.g., "set subject to Johnny", "set subject to join me"). We can use dynamic time warping for detecting this distance. Some work on obtaining implicit feedback from speech was performed in [11].

9 Acknowlegments

This work was a part of the InMind project for the creation of a smart personal assistant [2].

Authors' Biographies

Merav Chkroun holds a degree in computer science engineering from JCT - Lev Academic Center, Israel. She is a PhD student in the computer science department in Ariel University, Israel and works there as a lecturer. Her main research interest is human agent interaction such as chatbots and virtual assistants.

Amos Azaria is a senior lecturer at CS, Ariel University, Israel. He received his PhD from Bar Ilan University, Israel in 2015 and was a post-doctoral fellow at CMU, Pittsburgh PA. Azaria has co-authored over 40 papers and won the Victor Lesser distinguished dissertation award for 2015.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[2] A. Azaria and J. Hong. Recommender systems with personality. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 207–210. ACM, 2016.

[3] A. Azaria, J. Krishnamurthy, and T. M. Mitchell. Instructable intelligent personal agent. In *AAAI*, volume 4, 2016.

[4] J. R. Bellegarda. Spoken language understanding for natural interaction: The siri experience. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 3–14. Springer, 2014.

[5] A. W. Biermann. *Natural language programming*. Springer, 1983.

[6] D. Bohus, A. Raux, T. K. Harris, M. Eskenazi, and A. I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *Academic and industrial research in dialog technologies*, pages 32–39, 2007.

[7] S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(2):286–298, 2007.

[8] J. Cassell. Embodied conversational interface agents. *Communications of the ACM*, 43(4):70–78, 2000.

[9] M. Chkroun and A. Azaria. "Did I say something wrong?": Towards a safe collaborative chatbot. 2018.

[10] J. Daemen and V. Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[11] M. Gilbert, I. Arizmendi, E. Bocchieri, D. Caseiro, V. Goffin, A. Ljolje, M. Phillips, C. Wang, and J. G. Wilpon. Your mobile virtual assistant just got smarter! In *INTERSPEECH*, pages 1101–1104, 2011.

[12] J. Herrmann. Different ways to support intelligent assistant systems by use of machine learning methods. *International Journal of Human-Computer Interaction*, 8(3):287–308, 1996.

[13] T.-H. K. Huang, A. Azaria, and J. P. Bigham. Instructablecrowd: Creating if-then rules via conversations with the crowd. In *2016 CHI*, pages 1555–1562. ACM, 2016.

[14] L. C. Klopfenstein, S. Delpriori, S. Malatini, and A. Bogliolo. The rise of bots: A survey of conversational interfaces, patterns, and paradigms. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, pages 555–565. ACM, 2017.

[15] K. Kuklinski, K. Fischer, I. Marhenke, F. Kirstein, D. Solvason, N. Kruger, T. R. Savarimuthu, et al. Teleoperation for learning by demonstration: Data glove versus object manipulation for intuitive robot control. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2014 6th International Congress on*, pages 346–351. IEEE, 2014.

[16] P. Lamere, P. Kwok, W. Walker, E. B. Gouvêa, R. Singh, B. Raj, and P. Wolf. Design of the cmu sphinx-4 decoder. In *INTERSPEECH*, 2003.

[17] T. J.-J. Li, A. Azaria, and B. A. Myers. Sugilite: creating multimodal smartphone automation by demonstration. In *CHI'17*, pages 6038–6049. ACM, 2017.

[18] W. Mark and R. Perrault. Calo: a cognitive agent that learns and organizes, 2004.

[19] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, and K. Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *The International Journal of Robotics Research*, 26(8):829–844, 2007.

[20] C. Quirk, R. Mooney, and M. Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL-15)*, pages 878–888, Beijing, China, July 2015.

[21] C. Quirk, R. J. Mooney, and M. Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *ACL (1)*, pages 878–888, 2015.

[22] A. Reed. *Creating Interactive Fiction with Inform 7*. Cengage Learning, 2010.

[23] J. Sohn, N. S. Kim, and W. Sung. A statistical model-based voice activity detection. *IEEE signal processing letters*, 6(1):1–3, 1999.

[24] M. Steedman and J. Baldridge. Combinatory categorial grammar, 2011.

[25] S. Thorne, D. Ball, and Z. Lawson. Reducing error in spreadsheets: Example driven modeling versus traditional programming. *International Journal of Human-Computer Interaction*, 29(1):40–53, 2013.

[26] W. Wahlster and A. Kobsa. User models in dialog systems. In *User models in dialog systems*, pages 4–34. Springer, 1989.

[27] T. Watanabe, M. Okubo, M. Nakashige, and R. Danbara. Interactor: Speech-driven embodied interactive actor. *International Journal of Human-Computer Interaction*, 17(1):43–60, 2004.

[28] L. S. Zettlemoyer and M. Collins. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, 2005.