A Multi-stage Approach to Facilitate Interaction with Intelligent Environments via Natural Language

Zinovia Stefanidi, Asterios Leonidis(✉), and Margherita Antona

Institute of Computer Science (ICS), Foundation for Research and Technology – Hellas (FORTH), Heraklion, Crete, Greece
{zinastef,leonidis,antona}@ics.forth.gr

Abstract. Due to the proliferation of Internet of Things (IoT) devices and the emergence of the Ambient Intelligence (AmI) paradigm, the need to facilitate the interaction between the user and the services that are integrated in Intelligent Environments has surfaced. As a result, Conversational Agents are increasingly used in this context, in order to achieve a natural, intuitive and seamless interaction between the user and the system. However, in spite of the continuous progress and advancements in the area of Conversational Agents, there are still some considerable limitations in current approaches. The system proposed in this paper addresses some of the main drawbacks by: (a) automatically integrating new services based on their formal specification, (b) incorporating error handling via follow-up questions, and (c) processing multiple user intents through the segmentation of the input. The paper describes the main components of the system, as well as the technologies that they utilize. Additionally, it analyses the pipeline process of the user input, which results in the generation of a response and the invocation of the appropriate intelligent services.

Keywords: Conversational agent · Chatbot · Intelligent environment · Intelligent home · Natural language processing · Home automation

1 Introduction

Research in the area of Intelligent Environments is booming over the last several years. The evolution of Internet of Things (IoT) along with the emergence of Ambient Intelligence (AmI) technologies have led to a plethora of web-based services and devices, with which the user interacts on an everyday basis, especially in the context of the Intelligent Home.

In order to achieve a natural and intuitive interaction with the intelligent environment, conversational agents (i.e. "chatbots") can be employed that utilize natural language - in the form of speech or text - to interact with the user. Over the last couple of years, due to advancements in Machine Learning (ML) and Speech Recognition and Understanding (SRU), their capabilities have expanded and their usage has spread, becoming a part of millions of households (118.5 Million Smart Speakers in the US

https://doi.org/10.1007/978-3-030-30712-7_9

alone since December 2018[1]). Popular examples of conversational agents in the form of virtual assistants are Amazon's Alexa[2], Microsoft's Cortana[3], Apple's Siri[4] and Google Assistant[5].

Using a conversational agent to communicate with a smart environment is not a new concept. There are a number of systems that use chatbots for home automation and control, even as kitchen assistants. However, in spite of the continuous progress and advancements in this area, there are still some considerable limitations in existing approaches. In particular, such systems require either user configuration before use, or reprogramming when adding a new service. This is inefficient, time-consuming, prone to errors, and most notably not user-friendly. Furthermore, errors in case of wrong or missing information when communicating with the Chatbot are not optimally handled from a user-centered perspective, thus resulting into a lack of understanding of the user's intent. This can prove to be particularly problematic, considering that errors during a conversation are commonplace. Especially, when speech recognition is involved, noise can easily alter the users input. Additionally, when the user's request is complex (e.g. "*Turn on the oven for 45 min at 180 °C and turn on the air-conditioning for 30 min at 22 °C*"), the necessary information is easily omitted or wrongly provided. Moreover, previous approaches are unable to handle input containing more than one user intents. For instance, the message "*turn off the water heater and play relaxing music in the bathroom*", should be split into two separate commands, namely "*turn off the water heater*" and "*play relaxing music in the bathroom*" that should be handled consecutively.

The proposed system aims to provide a scalable software framework that can be used by conversational agents in order to facilitate user interaction with any of the available services of the intelligent space (e.g. home, classroom, greenhouse) in a natural manner. To that end, the framework:

- **automatically integrates new services** based on their formal API specification without the need for reconfiguration or user action
- **incorporates fundamental error handling**, by posing a series of follow up questions to the user, in order to acquire the necessary missing information and
- **handles user input containing multiple intents** by splitting it into separate sentences, which are then processed sequentially.

2 Related Work

Nowadays, Conversational Agents are becoming an integral part of our daily lives. A steadily increasing number of applications utilize them to achieve a more natural and seamless interaction between the user and the system. Notable applications that

[1] https://www.nationalpublicmedia.com/wp-content/uploads/2019/01/Smart-Audio-Report-Winter-2018.pdf.

[2] https://developer.amazon.com/alexa-voice-service/sdk.

[3] https://developer.microsoft.com/en-us/cortana.

[4] https://developer.apple.com/siri/.

[5] https://developers.google.com/assistant/sdk/.

incorporate Chatbots can be found in numerous fields, such as medicine [1, 2] and education [3–6]. Particularly in Intelligent Environments, populated by multiple heterogeneous devices and different IoT ecosystems, a single chatbot can serve as a common interface [7]. According to [7], this approach can address technological as well as human-centric challenges of IoT systems.

In the context of the Intelligent Home, there have been a number of applications that employ a chatbot or voice commands for the automation and control of the house [8–12]. Some of them accept as input simple commands such as “Turn on” and “Home” [8], while others understand natural language and engage in a conversation with the user [9, 10]. Some systems particularly focus on the Smart Kitchen, developing a conversational kitchen assistant that provides cooking recipes and nutrition information [13, 14]. In [13], the conversational agent can also reason about dietary needs, constraints and cultural preferences of the users, whereas in [14], it can guide the user throughout the cooking process.

For the development of conversational agents, different technologies and frameworks are employed, such as IBM’s Watson[6], Google’s DialogFlow[7] and Facebook’s Messenger Platform[8]. The majority of those technologies rely on intent classification and intent extraction of the user input, using Natural Language Processing (NLP) methods. This entails training a Machine Learning model with multiple examples for each user intent. Another technique used to process user input utilizes keyword and action lists, where the former contains all the possible keywords relevant to the system (e.g. light, TV, temperature) and the latter contains all possible actions (e.g. open, close, increase).

3 System Objectives

The proposed system aims to facilitate Human - Computer Interaction (HCI), in the context of an AmI environment, by utilizing the Natural Language Interaction paradigm. It incorporates a Conversational Agent in the form of a Virtual Assistant with whom the user can interact, not only through text messages, but also through speech. The components of an AmI environment are exposed as services to the system, enabling the user to communicate with the environment through the Conversational Agent in a natural and intuitive manner. In particular, the system’s objectives are threefold: (a) provision of information regarding the intelligent environment, (b) execution of commands that affect the intelligent environment, and (c) programming the behavior of the surrounding intelligent environment.

Provision of Information. An integral part of the system is the provision of information about the environment using natural language. For instance, in the context of the Smart Greenhouse, the user can inquiry about the condition of the crops or the environmental conditions inside the greenhouse. The system provides timely

[6] https://www.ibm.com/watson.

[7] https://dialogflow.com/.

[8] https://developers.facebook.com/docs/messenger-platform/.

information by communicating with the appropriate service. Consequently, the user can be kept informed and up-to-date about the environment, even remotely.

Execution of Commands. Another essential part of the system is to execute commands issued in natural language. For example, in the context of the Smart Kitchen, the user can turn on the coffee machine, or turn off the oven by expressing that intent. The system understands the task the user wants to perform and calls the appropriate function of the corresponding service. Therefore, the user can perform even complex actions instantly and intuitively.

Programming of the Surrounding Environment. Apart from acquiring information and executing actions, the user can program the environment by defining automations in the form of if-then statements. Through the trigger-action paradigm, users can define triggers that initiate specific actions when their conditions are met. For instance, in the context of the Smart Greenhouse, a trigger could be "*if humidity falls below 50%*", with the resulting action being "*turn on the sprinklers*". Thus, common operations in the user's environment are automated using natural language.

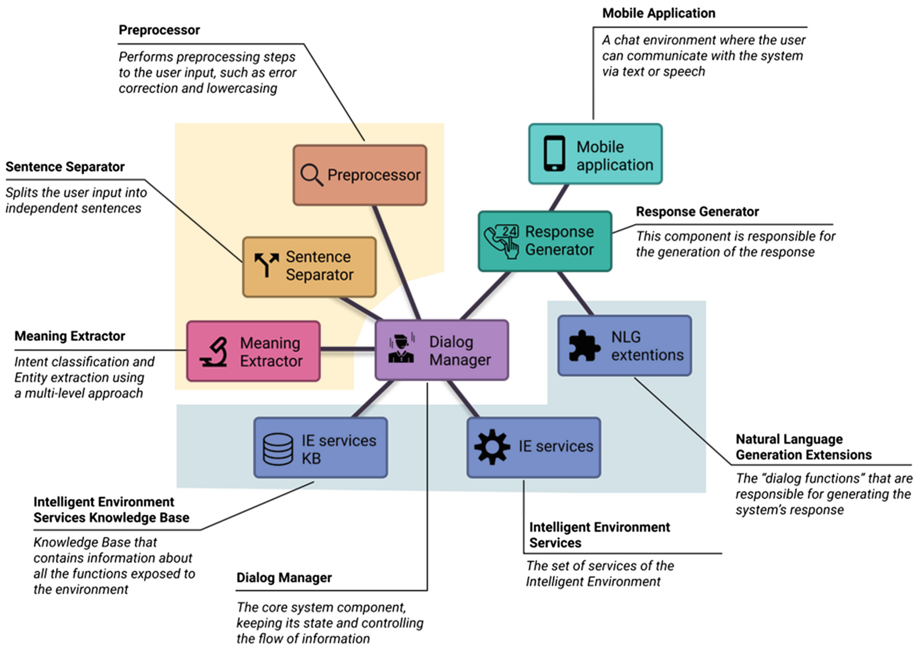

Preprocessor
Performs preprocessing steps to the user input, such as error correction and lowercasing
Mobile Application
A chat environment where the user can communicate with the system via text or speech
Sentence Separator
Splits the user input into independent sentences
Preprocessor
Mobile application
Response Generator
Response Generator
This component is responsible for the generation of the response
Sentence Separator
Meaning Extractor
Intent classification and Entity extraction using a multi-level approach
Meaning Extractor
Dialog Manager
NLG extentions
IE services KB
IE services
Natural Language Generation Extensions
The "dialog functions" that are responsible for generating the system's response
Intelligent Environment Services Knowledge Base
Knowledge Base that contains information about all the functions exposed to the environment
Intelligent Environment Services
The set of services of the Intelligent Environment
Dialog Manager
The core system component, keeping its state and controlling the flow of information

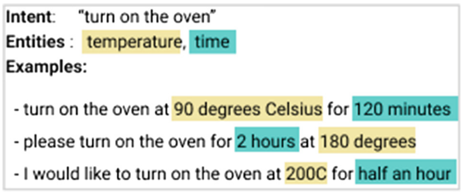
Fig. 1. The overall architecture of the proposed system.

4 System Architecture

As Fig. 1 illustrates, the system comprises of three main categories of components namely: (a) components that process user input aiming to extract its meaning, (b) components that interact with the services of the intelligent environment, and (c) components that manage the conversation flow and communicate with the user.

Preprocessor. It processes the user input before sending it to the Sentence Separator and performs various actions (e.g. lowercasing, lemmatization and error correction) to streamline the subsequent steps of the analysis pipeline.

Sentence Separator. Splits the user input into independent sentences. For example, the input "*turn on the light and the TV*" is split into the sentences "*turn on the light*" and "*turn on the TV*". This is achieved through a heuristic approach, which incorporates the Dependency Parsing and Part-of-speech (POS) Tagging facilities of the SpaCy[9] framework, along with custom algorithms that aim to generate complete sentences by filling-in any implicitly defined data.



Intent: "turn on the oven"
Entities : temperature, time
Examples:

- turn on the oven at 90 degrees Celsius for 120 minutes
- please turn on the oven for 2 hours at 180 degrees
- I would like to turn on the oven at 200C for half an hour

Fig. 2. Part of the definition of the "turn on the oven" intent. Rasa NLU relies on such detailed definitions to understand user input.

Meaning Extractor. This component uses Rasa NLU[10], an open source Python library for intent classification and entity extraction. In particular, three machine learning models are used, which are trained using the training examples that every intelligent service has registered in the IE Service Knowledge Base, as seen in Fig. 2.

- *Level-1* model: a general model that aggregates indicative examples from all the connected services
- *Level-2* service-specific models: they describe which intents a service can accommodate (i.e. the functions that if offers)
- *Level-3* function-specific models: they define in detail the arguments that a specific function of a certain service can have.

In more detail, the Level-1 model is mainly used for deciding the service with which the user wants to communicate, whereas the Level-2 model is primarily used for

[9] https://spacy.io: A library for advanced natural language processing in Python.
[10] https://rasa.com.

firstly deciding the function of the service that needs to be called and then extracting its arguments. Finally, Level-3 models are used for extracting the missing or wrong arguments of the initial user input, in a follow-up clarification dialog, when needed. This hierarchical approach is used to improve the accuracy of intent classification as already confirmed in [15]. Moreover, common user intentions such as "greet" and "help" are also incorporated and recognized from these models, with their semantics being model-dependent. For instance, the treatment of the "help" intent differs between the generic Level-0 model and a specific Level-1 model; in the former case the system should provide a general help message to the user, whereas in the latter case, the system should deliver context-sensitive instructions with respect to the given service.

Intelligent Environment Services and Knowledge Base. Each AmI service should provide an API that contains information about all the functions it exposes to the environment (Fig. 3). Concretely, for each function, its definition should contain the function arguments, their type, and their range or accepted values. In addition, it should include training examples of user input that correspond to the specific function being called. These examples are used to train the model that determines which service function needs to be called for a given user input. The set of all the services' formal specifications populate the Intelligent Environment Services' Knowledge Base.

Natural Language Generation (NLG) Extensions. Every service should provide a Natural Language Generation (NLG) extension providing information or the dialogue. Specifically, this extension should supply, for every function in the service, "dialog functions" that are responsible for generating in natural language the system's response upon successful execution or in different failure cases (e.g. when arguments are missing, when an argument is wrong), so as to correctly produce the response that will communicate the outcome to the user (e.g. provide a summary to the user with respect to the lock state of the home's doors, windows and shutters).

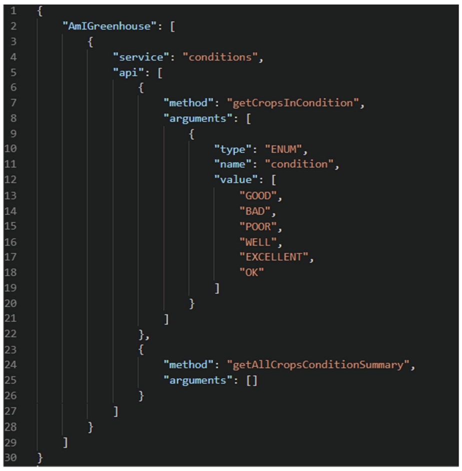
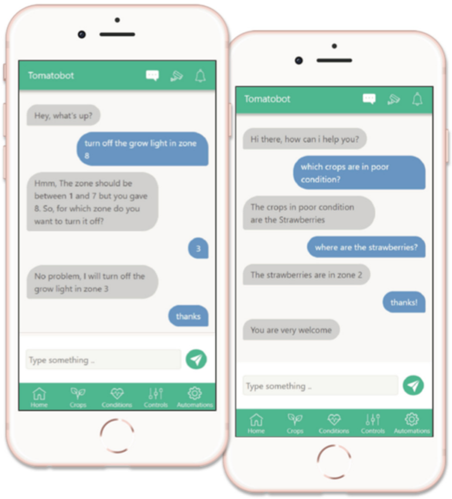

```
{
    "AmIGreenhouse": [
        {
            "service": "conditions",
            "api": [
                {
                    "method": "getCropsInCondition",
                    "arguments": [
                        {
                            "type": "ENUM",
                            "name": "condition",
                            "value": [
                                "GOOD",
                                "BAD",
                                "POOR",
                                "WELL",
                                "EXCELLENT",
                                "OK"
                            ]
                        }
                    ]
                },
                {
                    "method": "getAllCropsConditionSummary",
                    "arguments": []
                }
            ]
        }
    ]
}
```

Fig. 3. Part of the formal API specification of an AmI Greenhouse's service.

Response Generator. This component uses ChatScript[11], which is a "next Generation" Chatbot engine with various advanced features and capabilities, in order to generate the responses to be communicated to the user. It invokes the appropriate dialog function from the service's NLG extension, depending on the state of the conversation, to produce the response. For instance, when an argument of a function is missing, it calls the corresponding dialog function which asks the user for that missing argument (e.g. "*Which window do you want to open?*"). The Response Generator also produces the responses to user intents that are not directed to a specific service, but refer to a more general context (e.g. when a user says "thank you" or "hello").

Dialog Manager. It is the core component of the system, keeping the system's state and controlling the flow of information. It communicates with the Meaning Extractor to discover the appropriate service and function and extract any provided arguments. Comparing the currently extracted data with the data that the discovered service requires, it deduces the system's state (e.g. wrong or missing arguments, successful extraction of all required arguments) and delegates control to the Response Generator for the generation of the appropriate response. In addition, provided that the state indicates that an intelligent service has to be invoked and all the required data are in place, the Dialog Manager is responsible of executing the call and forwarding the result to the Response Generator for further processing.

Mobile Application. This component is a chat environment where the user can communicate with the Conversational Agent via text messages or speech through a smartphone, as depicted in Fig. 4.


Tomatobot

Hey, what's up?

turn off the grow light in zone 8

Hmm, The zone should be between 1 and 7 but you gave 8. So, for which zone do you want to turn it off?

3

No problem, I will turn off the grow light in zone 3

thanks

Tomatobot

Hi there, how can i help you?

which crops are in poor condition?

The crops in poor condition are the Strawberries

where are the strawberries?

The strawberries are in zone 2

thanks!

You are very welcome

Fig. 4. Sample conversations between the Chatbot and the user in the context of a Smart Greenhouse.

[11] http://chatscript.sourceforge.net.

5 The Analysis Pipeline

The input is processed in consecutive steps in order to understand the user's intentions, invoke the appropriate intelligent service, and generate the response (Fig. 5). The analysis pipeline is used for all three types of user intentions in the context of the intelligent environment, namely the acquisition of information, the execution of commands and the programming of the environment's behavior.

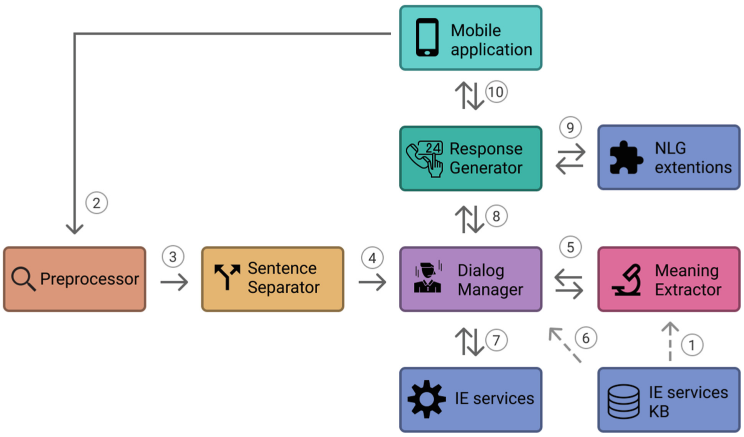


Fig. 5. A high-level view of the analysis pipeline.

Step 1: Initially, the Entity Extractor dynamically trains at run-time its internal recognition mechanisms with the appropriate model(s), based on the current dialog state; the training models are retrieved from the IE Services' KB. In particular, at the beginning the Entity Extractor loads the general Level-1 model that collects indicative examples from all the available Level-2 models (i.e. available Intelligent Environment Services), so as to be able to determine the service that the user most likely refers to. As soon as the desired service is detected (see Step 5), then the service-specific Level-2 model is used for training to facilitate the recognition of the desired function. Finally, if the conversation's state indicates that a number of arguments are missing or are incorrect, then a Level-3 model that corresponds only to the selected function is automatically generated and loaded to aid the extraction of the missing/incorrect data in a follow-up dialog.

Step 2: The user input is forwarded from the UI to the Preprocessor, where it is adapted appropriately.

Step 3: The adapted input is propagated from the Preprocessor to the Sentence Separator, which will split it into sentences if needed.

Step 4: Each distinct sentence is dispatched to the Dialog Manager, where further processing begins to understand the user's intentions and act accordingly.

Step 5: The input is forwarded to the Meaning Extractor component (whose internal recognition mechanisms have been prepared during step 1) to firstly discover which IE service the user wants to use (e.g. Light Service, HVAC Service, Cooking Assistance Service), and subsequently decide to which function of that service the user refers to (i.e. extract the user's intent which uniquely identifies a specific function). During this step, possible entities that correspond to the arguments of the desired function are also extracted. This information is sent back to the Dialog Manager for further processing.

Step 6: Since the desired function is detected, the system knows the exact number of arguments and types that it should anticipate. Subsequently, if any arguments have been extracted during the previous step (i.e. step 5), their types and values are compared with the expected ones; if any mismatches are found (e.g. missing arguments, incorrect types, values outside of the permitted bounds), the dialog's state changes accordingly and the Dialog Manager is notified to act accordingly (i.e. start a follow-up dialog to address these issues).

Step 7: If the system has all the necessary information to execute the function (i.e. no missing or wrong arguments exist), then the actual call to the IE service is carried out. As soon as the remote call returns, the Dialog Manager incorporates any results into the state and forwards control to the Response Generator.

Step 8: The Response Generator examines the dialog's state and schedules the generation of the appropriate response (see Step 9).

Step 9: If the user refers to a specific service, the Response Generator, depending on the state of the conversation, calls the appropriate dialog function from the service's NLG extension in order to produce the response to be sent to the user, namely: (a) ask for a missing argument, (b) notify that a value of an argument is out of range, (c) report the success of a function call along with any returning messages, or (d) report the failure of a function call and any possible error messages; for the two latter cases, the Response Generator retrieves any data posted by the Dialog Manager at step 7 that correspond to the value(s) that the function returned when invoked. For instance, if an argument of a function is wrong, it calls the dialog function that informs the user about the mistake, and asks for the missing argument (e.g. "*The zone number should be between 1 and 7 but you gave 9. So, in which zone do you want to turn on the water pump?*"); on the contrary, if a function call executed correctly, it uses the dialog function that reports the success message to the user (e.g. "*The alarm is set for tomorrow morning at 6:45 AM*"). If on the other hand the user's intent is not directed to a specific service, but belongs to a conversation topic of general interest, then an internal built-in model is used to generate the answer without having to consult any NLG extension.

Step 10: Finally, the response is communicated back to the user via the UI.

6 Future Work

A significant future advancement of the system will be the integration of context awareness. Contextual information, such as the location of the user, his profile, his current activity, as well as the time the conversation is taking place, will further enhance the system's user-friendliness and efficiency. Additionally, the syntactic structure and lexical analysis of the user input is going to be utilized for the improvement of service disambiguation and intent classification. Another future improvement could be the semi-autonomous generation of training examples for the NLU JSON APIs of the services. This will increase the number of the training examples and reduce human effort, while also potentially improving the accuracy of intent classification. Furthermore, the system's sentence separation will be enhanced, in order to deal with more complex cases, where attributes are involved. For example, the input "*turn on the bedroom's lights and TV*" should be split into "*turn on the bedroom's lights*" and "*turn on the bedroom's TV*", with the attribute "*bedroom's*" being included in both sentences. The system will also undergo user-based evaluation in the setting of simulated intelligent environments.

Acknowledgements. This work is supported by the FORTH-ICS internal RTD Program "Ambient Intelligence and Smart Environments".

References

1. Comendador, B.E.V., Francisco, B.M.B., Medenilla, J.S., Mae, S.: Pharmabot: a pediatric generic medicine consultant chatbot. J. Autom. Control Eng. **3** (2015)
2. Madhu, D., Jain, C.J.N., Sebastain, E., Shaji, S., Ajayakumar, A.: A novel approach for medical assistance using trained chatbot. In: 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, pp. 243–246. IEEE (2017)
3. Song, D., Oh, E.Y., Rice, M.: Interacting with a conversational agent system for educational purposes in online courses. In: 2017 10th International Conference on Human System Interactions (HSI), Ulsan, South Korea, pp. 78–82. IEEE (2017)
4. Heller, B., Proctor, M., Mah, D., Jewell, L., Cheung, B.: Freudbot: an investigation of chatbot technology in distance education. In: EdMedia + Innovate Learning, pp. 3913–3918. Association for the Advancement of Computing in Education (AACE) (2005)
5. Kerlyl, A., Hall, P., Bull, S.: Bringing chatbots into education: towards natural language negotiation of open learner models. In: Ellis, R., Allen, T., Tuson, A. (eds.) International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 179–192. Springer, London (2006). https://doi.org/10.1007/978-1-84628-666-7_14
6. Ranoliya, B.R., Raghuwanshi, N., Singh, S.: Chatbot for university related FAQs. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, pp. 1525–1530. IEEE (2017)
7. Kar, R., Haldar, R.: Applying chatbots to the internet of things: opportunities and architectural elements. Int. J. Adv. Comput. Sci. Appl. **7** (2016). https://doi.org/10.14569/ijacsa.2016.071119

8. Parthornratt, T., Kitsawat, D., Putthapipat, P., Koronjaruwat, P.: A smart home automation via Facebook chatbot and raspberry Pi. In: 2018 2nd International Conference on Engineering Innovation (ICEI), Bangkok, pp. 52–56. IEEE (2018)
9. Baby, C.J., Khan, F.A., Swathi, J.N.: Home automation using IoT and a chatbot using natural language processing. In: 2017 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, pp. 1–6. IEEE (2017)
10. Baby, C.J., Munshi, N., Malik, A., Dogra, K., Rajesh, R.: Home automation using web application and speech recognition. In: 2017 International Conference on Microelectronic Devices, Circuits and Systems (ICMDCS), Vellore, pp. 1–6. IEEE (2017)
11. Zhu, J., Gao, X., Yang, Y., Li, H., Ai, Z., Cui, X.: Developing a voice control system for zigbee-based home automation networks. In: 2010 2nd IEEE International Conference on Network Infrastructure and Digital Content, pp. 737–741. IEEE (2010)
12. Baig, F., Beg, S., Khan, M.F.: Controlling home appliances remotely through voice command. arXiv Prepr. arXiv12121790. (2012)
13. Angara, P., et al.: Foodie fooderson a conversational agent for the smart kitchen. In: Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, Riverton, NJ, USA, pp. 247–253. IBM Corporation (2017)
14. Rystedt, B., Zdybek, M.: Conversational agent as kitchen assistant (2018)
15. Jenset, G.B., McGillivray, B.: Enhancing domain-specific supervised natural language intent classification with a top-down selective ensemble model. Mach. Learn. Knowl. Extr. **1**, 630–640 (2019). https://doi.org/10.3390/make1020037