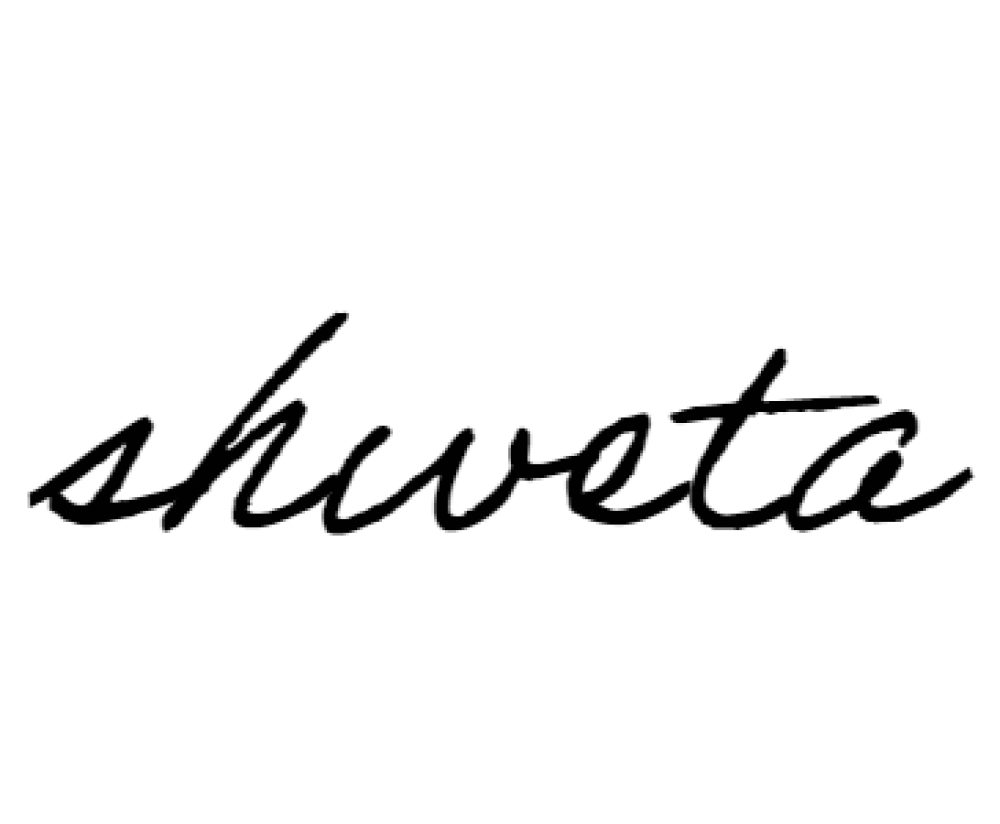
***VOICE ASSISTANT***

GUIDED BY : ER. SHWETA AGARWAL

******

|  |  |  |  |
| --- | --- | --- | --- |
| SUMIT | ADITYA | SHIVAM | BINIT |
| BE-CSE | BE-CSE | BE-CSE | BE-CSE |
| Chandigarh University | Chandigarh University | Chandigarh University | Chandigarh University |
| Gharuan , India | Gharuan , India | Gharuan , India | Gharuan , India |
| **sumbeckham@gmail.com** | **adityaarora742@gmail.com** | **badoniyashivam@gmail.com** | **binitsingh9903@gmail.com** |

***Abstract*—Nowadays Virtual Personal Assistants (VPA) become more and more popular. VPAs help to increase quality of life especially for elderly or disabled people. In this paper we develop an open source VPA and smart Voice Assistant or VA system as a 3- rd party extension to show the functionality of the assistant. The system is designed to use the VPA as a learning platform for engineers to provide them with the opportunity to create and test their own hypothesis. The VPA is able to recognize users’ commands in natural language and transform it to the set of machine commands that can be used to control different 3rd- party application. We use smart Voice Assistant or VA system as an example of such 3rd-party. We demonstrate that the system is able to control Voice Assistant appliances, like lights, or to display information about the current state of the Voice Assistant or VA, like temperature, through a dialogue between a user and the Voice Assistant or VA : Virtual Personal Assistant.**

***Index Terms*—Personal Assistant, Smart Voice Assistant or VA, Conversation Support, Command recognition**

1. INTRODUCTION

Nowadays we live in times when society has been changing the way of interaction with machines. Thanks to the achieve- ments in the ﬁeld of Python, Natural Language Processing (NLP) and Machine Learning (ML), computers can better and better recognize natural languages. One of the applications of this technology was found in a Vitual Personal Assistant.

Voice Assistant or VA is a software that helps users to achieve goals by performing tasks and services on behalf of an individual. To be efﬁcient, a VPA usually combines user input with other implicit information that it can gather, such as information from online sources, location, and history of previous user behavior. There are a lot of different Personal Assistants that were created for the last ten years, the most popular ones are Siri, Cortana and Alexa.

The user can interact with a Personal Assistant via different interfaces, but most common ones are textual commands through chat and voice. Textual and voice commands require from developers to match user input to executable commands. It requires solving a huge list of NLP problems such as lemmatization, part of speech tagging, chunking, named entity recognition, intent extraction, support dialog and so on. After the extraction of the intent from user’s query, a VPA can delegate the task to other 3rd party services that will execute commands.

The creation of a complex VPA that is able to handle a huge variety of tasks is very hard and time-consuming and only a few big companies like Microsoft or Google are able to create such types of assistants . These assistant are commercial

products thus they do not provide any access to their source code, which means developers are limited in being able to modify them and test hypothesis.

One of the trading applications of Personal Assistant is the control of the smart Voice Assistant or VA environments. Controlling different properties of the laptop such as play Music, Brightness control, as well as some more complicated factors like identity recognition becomes easier every year, mainly because of decreasing price of hardware components and increasing power of microprocessors. These hardware components allowing us to control the environment and collect the data are usually referred as Internet of Things (IoT).

In this paper, we introduce our Voice Assistant or VA : The Virtual Personal assistant VPAs that is capable to understand user’s intent, extract parameters from utterances, transform them to machine commands and route the commands to automation module that handles the execution of these commands. The system can be easily extended by other modules developed by 3rd party developers. In the following chapters, we are going to provide the general background of existing work, describe the architecture of the system and its requirements. Then we will present results that we obtained and talk about the current and future

steps.

1. RELATED WORK
2. *Personal Assistant*

Virtual Personal Assistants are very popular nowadays, thanks to Google, Amazon and Apple more and more people start using VPAs in their everyday life. A modern Personal Assistant does not only performs actions on behalf of a user but also tries to predict the Voice actions and provide proactive assistance. Many early assistants were focused on speciﬁc tasks, but nowadays more and more systems can be extended to deal with a variety of different tasks. We can observe it in the examples of Google’s Assistant and Apple’s Siri. Such architecture makes a VPA more and more intelligent and user friendly.

But everything has a price. VPAs involve more and more computationally expensive methods of data processing in order to achieve human capabilities in understanding a text, speech and Manual Support. A team from the University of Michigan has made VPA Lucida with the aim

to evaluate requirements to future data centers which play the role of back system for processing VPA tasks . A team of engineers demonstrated that the voice query requires 164 times greater time than traditional query in a web searcher. Such time consumption imprints on the scalability of the system. As the information volume grows, the system spends more time on applying different NLP ﬁlters in order to choose the best an- swer of a question. Developers provide speciﬁc measurements of time consumption for a Question-Answering module which spends on average 85% of the cycles on three components: stemming, regular expression pattern matching and CRF. In the paper of J. Hauswald et al. demonstrate possible solutions related to the problems of resource consumption, but still Lucida can not be run on modern smartphones.

1. *Comparison of Personal Assistants*

Nowadays high level architecture of all VPAs resemble each other. We can highlight modules such as Speech-To- Text, Text Processing, Information extraction, Intent handlers and Text-To-Speech. Variability of actions which a VPA can perform depends on the plugged intent handlers. For this reason, comparing assistants by plugged modules does not reﬂect the real quality of them. The main advantages of a VPA are hidden in natural language processor modules. Companies that own proprietary assistants do not provide detail technical information about their implementation of the modules. In its turn, open source VPAs such as MyCroft, Open Assistant, Jasper and ADRIAN use open source libraries and some of the libraries can be integrated in several assistants, e.g. library Pocketsphinx is involved in MyCroft, Open Assistant and Jasper . For this reason, we should compare no entire VPAs to another one, but analyze each module separately. But this comparison is beyond the scope of our article.

1. *Natural language processing*

For the last few years, there has been a huge growth of Internet Of Things . There are a lot of deﬁnitions for, but it is usually used to refer to a set of objects that are connected to the internet and can possibly communicate with each other via different protocols. These can be various sensors, actuators, embedded electronics and etc. At the moment there are around 22 billion IoT devices all over the world. Internet Of Things paradigm enables us to create a huge amount of different communication scenarios, where all IoT devices are supposed to communicate and cooperate. Different scenarios could be used in different ﬁelds of our life, such as in Voice Assistant or VA automation, Health, Transportation and Logistics and more .

A smart Voice Assistant or VA consists of a set of IoT devices communicat- ing with each other to achieve the goals of Voice Assistant or VA automation and monitoring. The market has many different IoT devices, but unfortunately, not all of them are compatible with each other and with a Digital Personal Assistant. Managing the communication between the devices is a big challenge when building Voice Assistant or VA automation

Another problem is that most of IoT devices are not capable of managing computationally intensive tasks, thus centralized control is required in order to build complex Automation system.

Smart Voice Assistant or VAs are automated environments that provide users with access to sophisticated monitoring and control over the house environment. The idea of building smart Voice Assistant or VA appeared in 1970s ; but the costs of automation were high and required signiﬁcant time and effort. Usually smart Voice Assistant or VAs operate with light, temperature, air conditioning and security, but they can be extended to control any part of the house like multimedia and even kitchen appliances and etc. Since prices and sizes of micro-controllers signiﬁcantly have been decreasing for the last 10 year we can automate almost anything that people can think of. Voice Assistant or VA automation can be divided into three layers by the tasks they are intended to solve:

* Sensing - provides a system with information from all the different sensors deployed in the Voice Assistant or VA. These data provide an understanding of the current state of thehouse.
* Reasoning - data collected from sensors then used by the reasoning layer to make decisions about what action the system should perform.
* Acting - is used to control the environment byperforming speciﬁc actions like turning on/off the light or increasing the temperature in the room.

In recent years it has become a common practice to combine Voice Assistant or VA automation systems with digital personal assistants.

Many big companies have presented their solutions to the

market like Google with Google Voice Assistant or VA, Apple with Voice Assistant or VAKit and Amazon with Alexa. All these projects use natural lan- guage queries from a user to control the Voice Assistant or VA automation. Amazon Alexa is a good example of a Personal Assistant that can have its tasks extended using different modules such as Voice Assistant or VA automation, news or video handling and more. It provides Alexa Smart Voice Assistant or VA Skill API for developers to setup automation system and extend the functionality [. All that developer needs to do is to add new functionality to specify what type of actions are possible and deﬁne the actions which can be invoked using natural language.

* 1. REQUIREMENTS

The following subsection covers the list of requirements that our prototype of VPA with Smart Voice Assistant or VA automation extension should satisfy. These requirements were chosen by comparing the already existing Digital Personal Assistance solutions to those of our VPA to identify the minimal, most common set of actions allowing the user to start using the system. The requirements presented in tables FR1 to FR7 are supported by the system while the creation of custom action scenarios is still under development and left as future work.

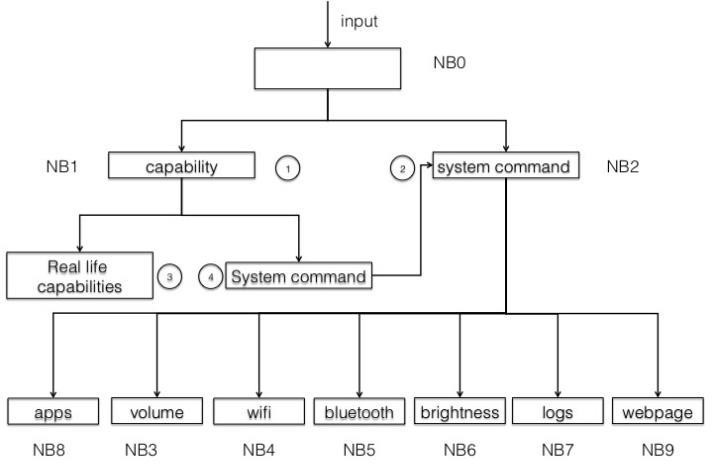
III. ARCHITECTURE

|  |  |
| --- | --- |
| **Requirement ID** | FR-01 |
| **Title** | Recognition of actions and questions |
| **Description** | The system should understand a user’s query and  distinguish between command and question |
| **Priority** | Mandatory |
| **Risk** | Critical |

* + 1. *General architecture*

Our prototype ecosystem consists of the VPA and 3-D party applications. The goal of the VPA is to receive, recognize and extract information from users request in natural language and then send it in machine understandable format to particular 3- D party app. In its turn, this app has to execute the users intent and provide the result of its job.

|  |  |
| --- | --- |
| **Requirement ID** | FR-07 |
| **Title** | Excitability |
| **Description** | The VPA must allow third-party developers to extend  its functionality via their devices and applications |
| **Priority** | Mandatory |
| **Risk** | Critical |

A 3-D party programmer can easily integrate his/her ap- plication with the VPA by adding description (Listing 1) of intents which the app can handle and what intents’ parameters are expected. Important to note is that the 3rd party system can be fully autonomous and work independently from the VPA.

|  |  |
| --- | --- |
| **Requirement ID** | FR-02 |
| **Title** | Information extraction |
| **Description** | The VPA should extract numbers, dates and string  corresponding to pattern. |
| **Priority** | Mandatory |
| **Risk** | Critical |

|  |  |
| --- | --- |
| **Requirement ID** | FR-03 |
| **Title** | Dialog support |
| **Description** | The VPA should ask a user additional question if the  information provided by the user is not enough for the execution of a query. |
| **Priority** | Mandatory |
| **Risk** | Critical |

* 1. USE CASES

|  |  |
| --- | --- |
| **Requirement ID** | FR-04 |
| **Title** | Sensor information |
| **Description** | Information, captured by any sensor must be shown  in the app. |
| **Priority** | Mandatory |
| **Risk** | Critical |

Use cases deﬁne the interaction with the VPA for achieving speciﬁc goals such as turn on/off activators, display measure of sensors and change of temperature.

|  |  |
| --- | --- |
| **User Case ID** | UC-01 |
| **Title** | Extension |
| **Description** | I as a developer can integrate with the VPA without  changes in VPAs code so that I can provide extensions of the functionality of the VPA. |

1. *VPA architecture*

We are developing the VPA with the idea that any program- mer can add easily support of his/her native language. For this purpose, we developed API for a language model. Each language is a module which has to handle particular language and provide information about the users request in uniform format for all languages. The format includes information about separating text on tokens and their description such as part of speech, named type and so on.

In the ﬁrst stage, we implemented the support of English. The language model for it is implemented with using CoreNLP project. CoreNLP separates text on tokens and does PoS and NER tagging.

The platform can extract three data type from a text. They are date, number and string which correspond to a pattern. At the current stage, the VPA expects that a request does not contain more than one value for each data type. This restriction allows to solve a slot ﬁlling problem in a straightforward way. After parsing the user’s utterance, the VPA begins to look for suitable intent. First of all, the assistant calculates the WMD for the incoming request and all samples of all intents. If a minimal distance for a sample is lower then the predeﬁned threshold then it stops the search. If the search by samples can not ﬁnd a close sample then the assistant tries to ﬁnd intent by key phrases. If even after it the VPA does not ﬁnd anything,

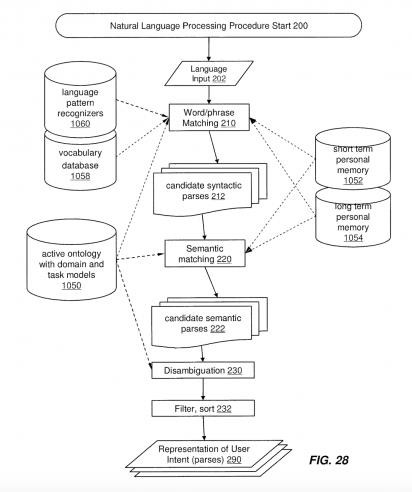
|  |  |
| --- | --- |
| **User Case ID** | UC-02 |
| **Title** | Dashboard information |
| **Description** | I as a user can request information about particular  sensor so that I can know the current status of the environment. |

|  |  |
| --- | --- |
| **User Case ID** | UC-03 |
| **Title** | Activator control |
| **Description** | I as a user can turn on and off a presence in the system  activators so that I can manage the work of activators. |

we notify the user that we do not recognize his/her request.

In case of success search, the VPA takes meta information about

expected arguments for the intent and based on it, tries to ﬁll slots of intent by values found in the user request. If the request does not contain a value for an obligatory slot, the VPA asks predeﬁned additional question for the slot. When all obligatory slots will be ﬁlled they will be passed to intent handler. After obtaining the handler execution result, the assistant displays it to user.



In order to interact with the VPA, the user just needs to type a message to it thought Telegram messanger.

1. *Our hardware and software requirements*

Constraints Creating : An AI that can answer every single question is not possible to implement with current technology and within the duration of the project, so the system will be able to answer questions about limited topics. The system will only support questions in standard English. Each feature will be assigned an importance value. The project will be complete if all the features of Priority 1 and at least 50% of features of Priority 2 are implemented. No Priority 3 requirements are necessary.

Priority Meaning

1 Speech to text input, Web surfing 2 Reminders, Alarms

3 Casual and random talks . In our implementation we used:

Hardware Required :

* 1. A touch screen or any other pointing device is required by the user in order to interact with the system.
  2. A microphone is required to implement speech to text.
  3. Speakers are required to receive audio outputs.
  4. Keyboard is required to Input search queries and other text inputs.

Software Required :

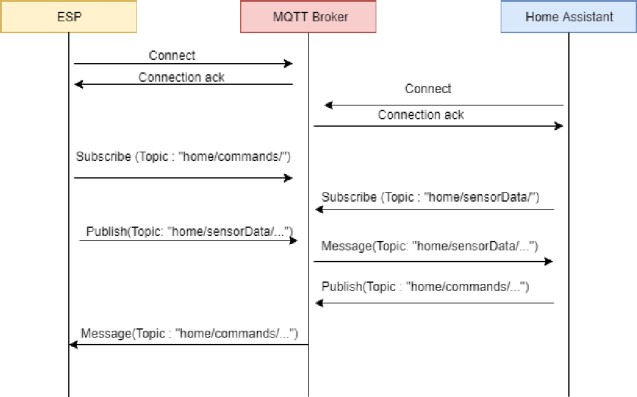
* + 1. Libraries such as numpy, pandas etc.
    2. Python 3
    3. PYTHON IDE such as spyder or PyCharm.
    4. System Features.

Most of the search engines today, like Google, use a system (The Page rank Algorithm) to rank different web pages. When a user enters a query, the query is interpreted as keywords and the system returns a list of highest ranked web pages which may have the answer to the query. Then the user must go through the list of webpages to find the answer they are looking for. Voice Assistant or VA Chatbot, however, will try to understand the query and provide a definitive answer. There will be four main units to the system working together to understand the question and return an appropriate answer:

* Generic question construction - capable of taking a natural language question and making it more generic.
* Generic answer construction - capable of taking a generic question template and providing a generic answer template.
* Generic answer population - capable of taking a generic answer template and populating it with information from the database to form an answer.
* Information extraction - capable of finding information through structured or unstructured websites, and storing that information in a database.

1. *Operating Enviornment*

Voice Assistant or VA is developed in open source and user-friendly environment of python with machine learning implementation. Baic libraries such as numpy, pandas are also used. Source code has been coded in python which supports the application as well as database.



[MQTT messaging sequence diagram]

There are two main functions in the communication between the Assistant and devices. The ﬁrst is collecting information from sensors and the second is to send commands to actuators. Initially, both ESP and the Assistant are connected to the broker. After acknowledgements of the con- nections, ESP which controls all the

devices, subscribes to a speciﬁed topic named ”Voice/commands” which is used by the Assistant to issue the commands. The personal Assistant on the other hand subscribes to a topic ”voice/sensorData” to which ESP will send information from the sensors. Now, when ESP is ready to send data, it publishes the message to the broker, using a pre- deﬁned topic (”Voice/sensorData/.values.”). After publishing the message, Broker takes all the responsibility to transfer the message to all Subscribers of a given topic, it attempts to deliver the message until it is successfully delivered to all subscribers. The same procedure happens when the Voice Assistant or VA Assistant is ready to send the command to ESP.

|  |  |
| --- | --- |
| **Expected Result** | 1. Setting I/O port to HIGH should switch relay and   turn on statusled.   1. Setting I/O port to LOW should switch relay and turn off status led. |
| **Status** | Success |

|  |  |
| --- | --- |
| **Test Case ID** | TC03 |
| **Description** | To verify that MQTT broker works properly and ESP  can successfully send sensor information through MQTT |
| **Reqs** | FR04,FR05,FR06 |
| **Test Procedure** | 1. Upload test program to ESP that will connect to   MQTT broker and send data from sensors every 2 seconds using ”Maya/sensorData” topic.   1. Open terminal on PC running MQTT and connect to MQTT broker. 2. Subscribe to topic ”Maya/sensorData” |
| **Expected Result** | 1.Every 2 seconds the terminal should show new data  from the Screen. |
| **Status** | Success |

1. *External Interface Requirements*

User Interfaces Application GUI Framework used:- TKINTER

* 1. The GUI will have a textbox that will accept inputs from a keyboard.
  2. Text box will originally contain a suggestive text question, to guide the user to the format of an appropriate question.
  3. The GUI will have a “Send” button which sends text from the textbox to the API when clicked.
  4. The GUI will have a chat window displaying questions sent to the system and responses from the API.
  5. The chat window will contain all questions and answers from the current session, with a scroll bar if all messages can’t fit on the screen.
  6. If there is a network issue, the chat window will display an error message.
  7. There will be a designated phone number that users can send texts to.
  8. Texts sent to that number will be sent to the API, then the system will reply to the user with the answer as another text message.
  9. If a question is not understood by our API, the system will send a text containing an example question after the text with the API response.

|  |  |
| --- | --- |
| **Test Case ID** | TC04 |
| **Description** | To verify that the VPA successfully delegates com-  mands execution to the Voice Assistant or VA Assistant and these commands are successfully performed |
| **Reqs** | FR01,FR02,FR03,FR05 |
| **Test Procedure** | 1. Connect the VPA with the Voice Assistant or VA Assistant 2. Input query ”Voice Assistant or VA, turn on the Music Player” in the VPA 3. Input query ”Voice Assistant or VA, turn off the Music Player” |
| **Expected Result** | 1. Relay speciﬁed for the Player should be switched on after the ﬁrst query. 2. Relay speciﬁed for the Player should be switched off after the second query |
| **Status** | Success |

Examples in tables EX1 to EX4 show what kind of queries the system is able to recognize and show the expected results of entity recognition for each case.

**CONCLUSION**

In this paper, we have introduced our prototype of an open source VPA. The system consists of two main components:

|  |  |
| --- | --- |
| **Example ID** | EX01 |
| **Dialog** | User: Maya, turn off  VPA: What should I turn off? User: the computer |
| **Expected JSON** | *{*”AppName”: ”Maya”, ”Intent”:”Turn off”, ”ob-  ject”: ”the computer”*}* |
| **Status** | Success |

|  |  |
| --- | --- |
| **Test Case ID** | TC01 |
| **Description** | To verify that users queries are recognized correctly |
| **Reqs** | FR01,FR02,FR03 |
| **Test Procedure** | 1.Input query ”Maya, turn on the Music Player” in VPA |
| **Expected Result** | 1.Resulting JSON ﬁle should specify that the Maya  Assistant was chosen as a delegate, Music player is the entity that should be affected, and turn on is the  commands. |
| **Status** | Success |

|  |  |
| --- | --- |
| **Example ID** | EX02 |
| **Dialog** | User: Calendar, remind 16th of November to meet  Sahil |
| **Expected JSON** | *{*”AppName”: ”Calendar”, ”Intent”:”Create remind”,  ”Subject”: ”meet Sahil”, ”Date”: ”16th of Novem- ber”*}* |
| **Status** | Success |

|  |  |
| --- | --- |
| **Test Case ID** | TC02 |
| **Description** | To verify that ESP successfully controls relays |
| **Reqs** | FR05 |
| **Test Procedure** | 1. Connect ESP with sensors to PC 2. Open Serial Monitor to send commands to ESP 3.Send command to change I/O port connected to relay to HIGH.   4.Send command to change I/O port connected to relay to LOW. |

<http://jasperproject.github.io/documentation/api/standard/>

|  |  |
| --- | --- |
| **Example ID** | EX03 |
| **Dialog** | User: Calendar, remind  VPA: When should I remind you? User: on Monday  VPA: What should I remind you? User: meet Sahil on the airport |
| **Expected JSON** | *{*”AppName”: ”Calendar”, ”Intent”:”Create remind”,  ”Subject”: ”meet Sahil on the airport”, ”Date”: ”Monday”*}* |
| **Status** | Success |

1. R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J.-P. Robichaud,
   1. Celikyilmaz, Y.-B. Kim, A. Rochette, O. Z. Khan, X. Liu *et al.*, “An overview of end-to-end language understanding and dialog management for personal digital assistants,” in *Spoken Language Technology Workshop (SLT), 2016 IEEE*. IEEE, 2016, pp. 391–

397. [Online]. Available: [https://www.microsoft.com/en-us/research/wp- content/uploads/2016/12/CortanaLUDialog-FromSLTproceedings.pdf](https://www.microsoft.com/en-us/research/wp-%20content/uploads/2016/12/CortanaLUDialog-FromSLTproceedings.pdf)

|  |  |
| --- | --- |
| **Example ID** | EX04 |
| **Dialog** | User: Voice Assistant or VA, turn off Music Player |
| **Expected JSON** | *{*”AppName”: ”Voice Assistant or VA”, ”Intent”:”Turn off”, ”ob-  ject”: ”Music Player”*}* |
| **Status** | Success |

* Conversational Interface that is able to translate natural human language to a set of machine commands.
* Smart Voice Assistant or VA agent that executes the commands issued by the user to control Voice Assistant or VA appliances as well as to provide the possibility to monitor data from the sensors deployed in a house.

We have demonstrated the functionality of the TEXT-TO- SPEECH and the Smart Voice Assistant or VA by showing their interaction with a real person. As future work, we will extend the Conversational interface to include a dialog support system, introduce new language modules and improve the slot ﬁlling functionality. Regarding the Smart Voice Assistant or VA agent, we intend to work on the possibility to create custom scenarios, improve data analysis, and concentrate on the safety aspects, to make the system more intelligent and safe. Future work will have to take into account the functioning of Personal Assistants in different scenarios, for example automotive [25], and how to re-engineer it in order to be deployed in a ﬂexible and continuous fashion, for example following the microservice paradigm .

REFERENCES

1. “Action on google,” feb 2018. [Online]. Available: https://developers.google.com/actions/
2. (2018) Sirikit. [Online]. Available: https://developer.apple.com/sirikit/
3. (2018) Mycroft skills. [Online]. Available: https://mycroft.ai/documentation/mycroft-software-hardware/#mycroft- skills
4. (2018, feb) How to create a standard jasper module. [Online]. Available:

I. B. M. Nalin and M. Mazzara, *A holistic infrastructure to support el- derlies independent living. Encyclopedia of E-Health and Telemedicine*. IGI Global, 2016.

1. K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, and N. Mavridis, “Microservice-based iot for smart buildings,” in *Advanced Information Networking and Applications Workshops (WAINA), 2017 31st Interna- tional Conference on*. IEEE, 2017, pp. 302–308.
2. “The virtual digital assistant market will reach $15.8 billion worldwide by 2021,” *Tractica*, aug 2016. [Online]. Avail- able: [https://www.tractica.com/newsroom/press-releases/the-virtual-](http://www.tractica.com/newsroom/press-releases/the-virtual-) digital-assistant-market-will-reach-15-8-billion-worldwide-by-2021/
3. J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khu- rana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang *et al.*, “Sirius: An open end-to-end voice and vision personal assistant and its implications for future warehouse scale computers,” in *ACM SIGPLAN Notices*, vol. 50, no. 4. ACM, 2015, pp. 223–238.
4. (2018, feb) Open assistant tools. [Online]. Available:<http://openassistant.org/tools/>
5. (2018, feb) Wake word detection. [Online]. Available: https://mycroft.ai/documentation/mycroft-software-hardware/#wake- word-detection
6. (2018, feb) Choosing an stt engine. [Online]. Available: [http://jasperproject.github.io/documentation/conﬁguration/#stt](http://jasperproject.github.io/documentation/conï¬guration/#stt)
7. M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, “From word embed- dings to document distances,” in *International Conference on Machine Learning*, 2015, pp. 957–966.
8. “Iot platforms market report,” *IoT Analytics*, jan 2016.
9. J. Santos, J. J. Rodrigues, J. Casal, K. Saleem, and V. Denisov, “Intelligent personal assistants based on internet of things approaches,” *IEEE Systems Journal*, 2016.
10. L. Tan and N. Wang, “Future internet: The internet of things,” in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 5. IEEE, 2010, pp. V5–376.
11. P.-V. Mekikis, A. Antonopoulos, E. Kartsakli, A. S. Lalos, L. Alonso, and C. Verikoukis, “Information exchange in randomly deployed dense wsns with wireless energy harvesting capabilities,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 3008–3018, 2016.

.