

Project Report

Topic: Intelligent Customer Help Desk with Smart Document Understanding.

Category: Machine Learning/Artificial Intelligence

Application ID: SPS_APL_20200000664

Project ID: SPS_PRO_99

Internship At: SmartInternz

Completed By: Debatra Chatterjee

Mail id: debatra.chatterjee.24@gmail.com

Contents:

1. INTRODUCTION

- a. Overview
- b. Purpose

2. LITERATURE SURVEY

- a. Present Problem
- b. Proposed Solution

3. THEORETICAL ANALYSIS

- a. Block diagram
- b. Hardware/Software requirements

4. PROCEDURAL ANALYSIS

5. FLOWCHART

6. RESULTS

7. ADVANTAGES AND DISADVANTAGES

8. APPLICATION

9. CONCLUSION

10. FUTURE SCOPE

11. BIBLIOGRAPHY

APPENDIX

Source Codes

1. INTRODUCTION

a. Overview

We design a customer help desk chatbot for Ecobee3. Ecobee3 is a smart thermostat falling under the category of smart home devices. This chatbot helps the user to gather information by raising a query. This chatbot accesses an unstructured document (user manual) of Ecobee3 and by smart understanding of the document it replies to any query related to Ecobee3.

b. Purpose

The main purpose of this project is to enhance the efficiency of the chatbot (Customer Help Desk) by incorporating Smart Document Understanding of Watson Discovery into it, so that both pre-defined and non-predefined queries, whose answers can be traced to the user manual, get an answer.

2. LITERATURE SURVEY

a. Present Problem

The usual chatbots are able to help customers with specific queries, like "What is the store location?", "What is the operation time?", "Book me an appointment."and so on. But, they are unable to give details about a particular device and its operations. So, whenever the query falls outside the pre-defined set, the chatbot allows us to initiate a conversation with a real person, a customer care representative.

b. Proposed Solution

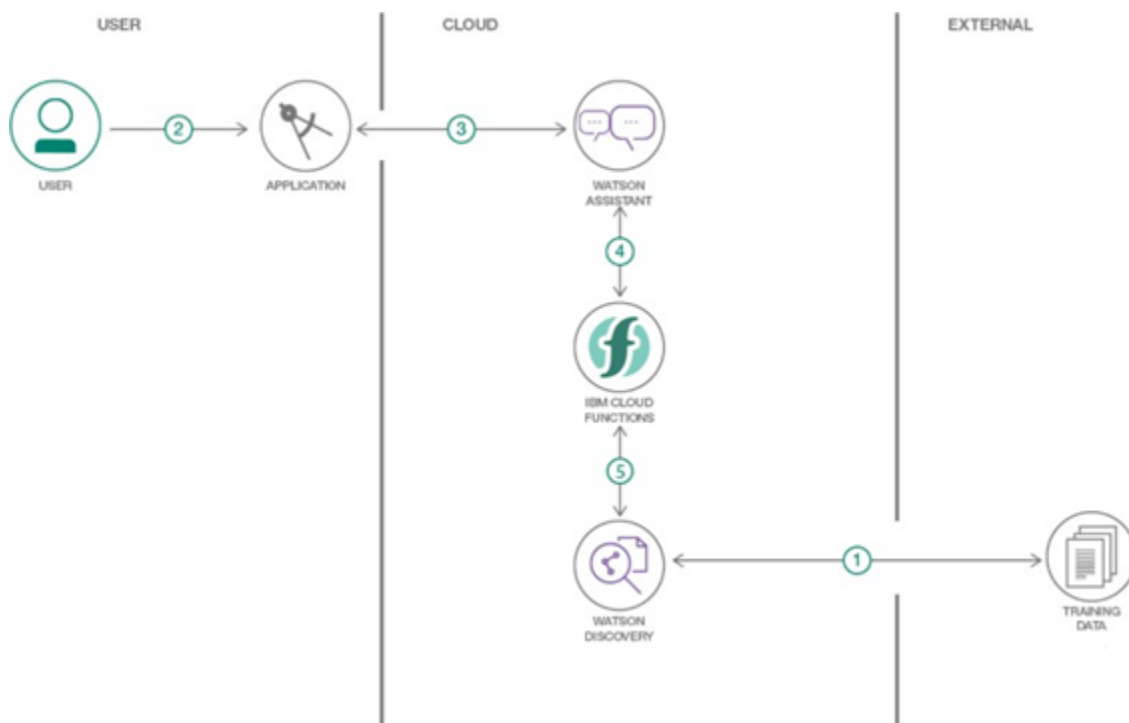
This project deals with those queries which are out of the set of pre-defined queries. One of the major part of these kind of queries are about adevice specification and handling assistance of the device.

Usually, facts about operating adevice is available in the user manual.With the help of Watson Discovery (an IBM cloud service), we are able to smartly analyze the user manual to obtain the important parts of the manual and divide it subtitle-wise.

Therefore, any query regarding the operations of the device gets a reply from the Helpdesk after it has accessed the user manual using Smart Document Understanding.

3. THEORETICAL ANALYSIS

a. Block Diagram



b. Hardware/Software Requirements

We have to build an IBM Cloud account. By creating IBM Cloud account we are eligible to create various IBM Services. For this project, we are required to build Watson Discovery Service, IBM Cloud function action, Watson Assistant Service and Node-Red App. Also, a document of the user manual of Ecobee3 is needed, which is fed to Watson Discovery Service.

4. PROCEDURAL ANALYSIS

- **Create an IBM Cloud Account.**

To create an IBM Cloud account, go to <https://www.ibm.com/cloud>, then click on [Create an account](#).

If an account exists then log in with the IBMid and password.

Then we create the following services in IBM Cloud:

- **Create Watson Discovery.**

Open the catalog option in IBM Cloud account and select [Services](#).

There, under the Artificial Intelligence(AI) section, Discovery Service can be found.

Note the "url" and "apikey" value, these will be required in cloud function action.

- **Create Watson Assistant.**

Open the catalog option in IBM Cloud account and select [Services](#).

There, under the Artificial Intelligence (AI) section, Watson Assistant can be found. Note the "url" and "apikey" values, these will be required in the Node-Red flow section of this project.

We will create a few other services, which will be required, later.

- **Configure Watson Discovery**

Next, we will have to configure the Watson Discovery service.

After launching Watson Discovery, click on [Upload your own data](#) to upload your dataset. In this case, this dataset will be the Ecobee3 User Guide manual.

After opening the dataset, go to [Configure Data](#).

In [Identify fields](#), label the data on the basis of fields like "title", "subtitle", "text", "footer" and so on.

Then open the [Manage fields](#) tab, where we only ask to identify using subtitles and text field by turning the rest of the fields “off”. The document is also split in the [Manage Fields](#) section on the basis of "subtitle" in this case, by selecting [Split Document On Each Occurrence Of](#).

Then click [Apply Changes To Collection](#) and upload the original document to save the changes.

Now, in the [Overview](#) page, we see that the original document is split into 103 documents (this is in my case. The number may vary depending on your field labelling). Also, Sentiment Analysis is seen. [Build your own query](#) helps to test the accuracy.

Finally, note the api values of "Collection ID" and "Environment ID", these will be required in the cloud function action(parameter).

- **Create IBM Cloud Function Action**

Here, we will build the web action that will make queries against the Discovery collection.

Go to IBM Cloud Dashboard and click on [Create Resources](#) then select Functions.

Click on [Actions](#) on the left panel. Then select [Create](#) and provide a unique action name.

Click on the **Code** tab. A certain code is written which connects Function to the Discovery Service, makes the query against the collection and returns the result. This code is provided in the Source Codes section of this report under the name of WatsonDiscovery Code.

Next, click on [Parameters](#) tab. The values of following parameters, "url", "environment_id", "collection_id" and "iam_apikey" can be accessed from the Discovery Service. This helps the action to connect to the Discovery Service.

Next, click on [Endpoints](#) tab. Click on [Enable as Web Action](#). This generates an URL of cloud function action. Note this URL, this will be required in Watson Assistant as webhooks.

- **Configure Watson Assistant.**

After launching the Watson Assistant, go to [Skills](#) tab. For our project, we use the sample Customer Care skill provided. So go to [Use Sample Skill](#), and choose [Customer Care Sample Skill](#).

[Intents](#) is a goal or purpose of user's questions. [Entities](#) is a specific detail of questions and statements. [Dialog](#) puts intent and entity together to provide interaction.

In the Intents tab, we will add one more intent to the intents already provided. We name this Product Information, and this intent will be used by the chatbot to respond to any question outside its scope using the User Manual uploaded in Watson Discovery. For this project, we give three example questions to this intent.

In the [Dialog](#) tab, we add another node to the ones already provided. In this Dialog node, we open the Customize section where we turn webhooks “on”. In this node, we give condition that if assistant recognizes product information intent, then it will give it name input. Then in the webhooks parameter section, we set key to “input” and value to “<?input.text?>”.

In the [Options](#) tag, we click webhooks and in the url section, we paste the url of Cloud Function Action and add .json to the end of the url and save it.

- **Build Node-RED flow to integrate all services together.**

Go to [Create Resource](#) and search for [Node-RED App](#). Open the [Node-RED App](#).

Provide a unique name for the app or accept the default name. Provide IBM Cloud API key or generate a new one. Also provide memory per instance (in this case 256Mb).

Select region to create the DevOps toolchain and select [Create](#).

Initially, the [Status](#) in the Delivery Pipelines will show "In Progress". The Deploy stage will take a few minutes to pass. Once the Deploy stage is passed, the status will turn to "Success" and will have a green tick mark beside it.

The newly created Node-RED App will be listed under the [App Section](#) in [Resource List](#). Also a corresponding entry under the [Cloud Foundry Apps](#) section can be seen.

Open the app from the Cloud Foundry App section in Resource list. From there, click [Visit App URL](#) to open the Node RED editor.

- **Build a web dashboard using Node-Red.**

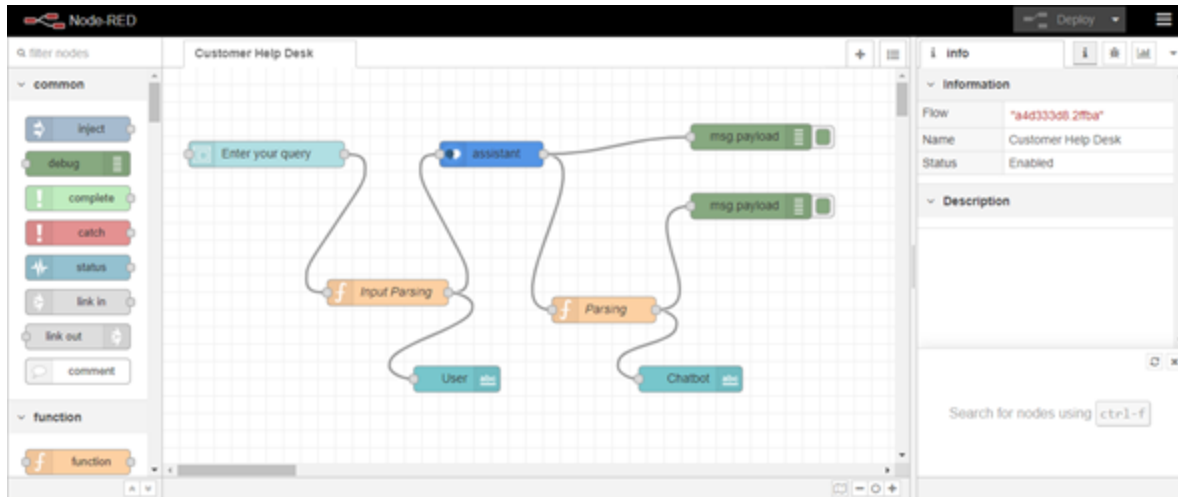
Go to [Settings](#) and then [Install](#) and search for [node-red-dashboard](#). Install the [node-red-dashboard](#).

Then we add the following nodes –

One form node, two text nodes, two debug nodes, two function nodes and an assistant node.

The "assistant" node is used to link the flow with the assistant (Customer Help Desk). The WorkspaceID ,Service end point and API key are the Skill ID, URL and API key respectively of the Watson Assistant.

Taking various other nodes from pallette, the desired web dashboard is built.



The "form" node helps to built the query section for the user.

One "text" node is named as "User" which displays the query asked by the user.


The other text node is named as "Chatbot" which displays the answer of the query by the bot.

The "assistant" node links the flow with the Watson Assistant (Customer Help Desk) to feed questions and obtain answer from it.

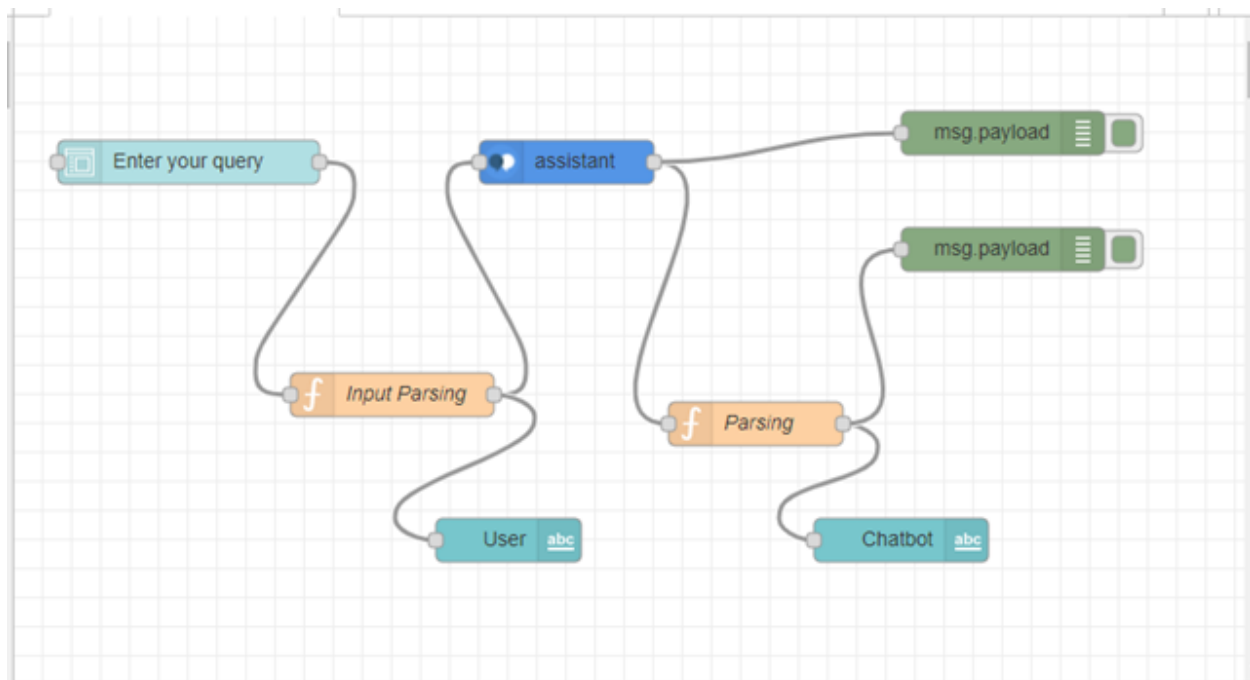
There are two "function" nodes, one named as "Input parsing" for feeding the question text to the assistant and the other named as "parsing" which converts output of assistant from json to text format.

The two "debug" nodes are used to help the designer debug the outputs.

The two "function" nodes have to be coded in order to extract text from json. The code is given in Source Code(Appendix).

Deploy the app after adding and connecting all nodes. To see the UI, select dashboard in the right column and click on .

5. FLOWCHART



6. RESULTS

Customer Help Desk

Default

User

How do I turn on heater

Chatbot

You can customize the brightness of your ecobee3's screen. The brightness for both the active and standby screens can be configured independently. You can also configure the screen to automatically sleep (i.e. turn off) whenever your ecobee3 enters the Sleep activity period. For example, if your thermostat is located in a bedroom, you may want to blank the screen when you are sleeping, whereas if the thermostat is in a hallway, you may want the screen displayed all the time. On Thermostat: 1. Select Main Menu > Settings > Preferences 2. Select Screen brightness. 3. Adjust the values of the Active and Standby screen brightness. 4. Select Screen sleeps when I sleep if you want to make the screen blank during the Sleep activity period. If you have a furnace or boiler installed: 1. Select the heating menu. 2. Configure the heater type: ☐ Furnace: Optimizes ecobee3 for systems using forced air ☐ Boiler: Optimizes your ecobee3 for systems using radiators or in-floor heat. 3. Touch Next. You will be returned to the Equipment configuration menu. The HVAC System settings depend on the type of system you have. Depending on your system, one or more the following options are shown: ☐ Cool: Turn on the air conditioner when the current temperature rises above the set temperature. ☐ Heat: Turn on the heat when the current temperature drops below the set temperature. ☐ Auto: Activate the heating or cooling system as required to keep your home within the configured range of set temperatures. ☐ Aux: Only use the auxiliary or backup heat source to maintain the heat set point temperature. This option only appears if auxiliary heat is configured in the Equipment menu. ☐ Off: Turn the system off. When the system is off, only the current temperature will be displayed on the Home screen. On Thermostat and Mobile: Select Main Menu > System > HVAC System On Web: Select System tile > HVAC

7. ADVANTAGES AND DISADVANTAGES

Advantages:

- a. Quick responses to complex queries.
- b. Multiple customer handling is possible.
- c. Reduction in workload of customer care representative.
- d. 24*7availability.

Disadvantages:

- a. Lessaccurate.
- b. Limited fields ofreply.

8. APPLICATION

This chatbot, Customer Help Desk, is mainly applicable for any query relating to the operation of the device. Instead of searching for details about the operations of a device from its user manual or waiting for reply from customer care representatives, the chatbot helps to find answers by just raising a query about any specific topic about the operation of the device.

9. CONCLUSION

Thus, by using the Smart Document Understanding feature of the Watson Discovery, we have been able to improve the functionality of the regular chat-bot. By training the Watson Discovery to compartmentalize various sections of the document provided, we not only provided the chatbot

with an ability to respond to questions that it was not trained to answer, but also greatly improve its accuracy. This can still be improved by providing more examples in the intent created and by training more meticulously.

10. FUTURE SCOPE

Device operation related answers are obtained but there is limited field of reply. Feeding higher number of documents (data) can lead to more accuracy. A personalized or more user friendly Chatbot is seen as the future scope to this system.

11. BIBLIOGRAPHY

Reference links:

1. <https://github.com/IBM/watson-discovery-sdu-with-assistant>
2. <https://www.youtube.com/watch?v=-yniuX-Poyw&feature=youtu.be>
3. <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
4. <https://drive.google.com/file/d/1pKM2lt793hv9RIBAWl4-VGihMXCTF4vI/view>
5. <https://drive.google.com/file/d/15s07ymOgBMInOf7mabqla5mLiAtlVJ31/view>

6. <https://www.youtube.com/watch?v=Jpr3wVH3FVA&feature=youtu.be>
7. <https://cloud.ibm.com/>

APPENDIX

SOURCE CODES-

1. WatsonDiscovery_code -

```
/*  
 * @param {object} params  
 * @param {string} params.iam_apikey  
 * @param {string} params.url  
 * @param {string} params.username  
 * @param {string} params.password
```



```

* @param {string} params.environment_id
* @param {string} params.collection_id
* @param {string} params.configuration_id
* @param {string} params.input
*
* @return {object}
*
*/

```

```

const assert = require('assert');
const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');

```

```

/**
 *
 * main() will be run when you invoke this action
 *
 * @param Cloud Functions actions accept a single parameter, which must be a
JSON object.
 *
 * @return The output of this action, which must be a JSON object.
 *
*/

```

```

function main(params) {
  return new Promise(function (resolve, reject) {

```

```

    let discovery;

```

```

    if (params.iam_apikey){
      discovery = new DiscoveryV1({
        'iam_apikey': params.iam_apikey,
        'url': params.url,

```

```

        'version': '2019-03-25'
    });
}
else {
    discovery = new DiscoveryV1({
        'username': params.username,
        'password': params.password,
        'url': params.url,
        'version': '2019-03-25'
    });

    discovery.query({
        'environment_id': params.environment_id,
        'collection_id': params.collection_id,
        'natural_language_query': params.input,
        'passages': true,
        'count': 3,
        'passages_count': 3
    }, function(err, data) {
        if (err) {
            return reject(err);
        }
        return resolve(data);
    });
});

```

2. Skill.json- This will be provided in the Git Repository.

3. Input_Parsing.txt –

```
msg.payload=msg.payload.text;
return msg;
```

4. Parsing.txt -

```
msg.payload.text="";
if(msg.payload.context.webhook_result_1){
  for(var i in msg.payload.context.webhook_result_1.results){

    msg.payload.text=msg.payload.text+"\n"+msg.payload.context.webhook_result_1.results[i].text;
  }
  msg.payload=msg.payload.text;
}
else
msg.payload = msg.payload.output.text[0];
return msg;
```

5. flows.json

```
[{"id":"a4d333d8.2ffba","type":"tab","label":"Customer Help Desk","disabled":false,"info":""},{
  "id":"92b64bbe.059d78","type":"ui_form","z":"a4d333d8.2ffba","name":"","label":"Enter your query","group":"3472749f.3e774c","order":0,"width":27,"height":1,"options":{"label":"","value":"text","type":"text","required":true,"rows":null},"formValue":{"text":""},"payload":"","submit":"submit","cancel":"cancel","topic":"","x":120,"y":100,"wires":[["898b5acc.453378"]]},
  {"id":"898b5acc.453378","type":"function","z":"a4d333d8.2ffba","name":"Input Parsing","func":"msg.payload=msg.payload.text;\nreturn msg;","outputs":1,"noerr":0,"x":260,"y":260,"wires":[["c315575b.2a7f48","ec294bd6.741438"]]},
  {"id":"c067b3b6.f4f74","type":"function","z":"a4d333d8.2ffba","name":"Parsing","func":"msg.payload.text=\"\\\";\\nif(msg.payload.context.webhook_result_1){\\n  for(var i in msg.payload.context.webhook_result_1.results){\\n    msg.payload.text=msg.payload.text+\"\\\"\\n\\\"+msg.payload.context.webhook_result_1.results[i].text;\\n}\\n\""}]
```

```
msg.payload=msg.payload.text;\n}\n\nelse\nmsg.payload =  
msg.payload.output.text[0];\nreturn  
msg;";outputs":1,"noerr":0,"x":510,"y":280,"wires":[["7ad872a3.a4bb7c","6babfa52.a  
49da4"]],{"id":"ec294bd6.741438","type":"ui_text","z":"a4d333d8.2ffba","group":"34  
72749f.3e774c","order":1,"width":27,"height":1,"name":"","label":"User","format":  
{{msg.payload}}","layout":"col-center","x":340,"y":360,"wires":[]},{"id":"7ad872a3.a4  
bb7c","type":"ui_text","z":"a4d333d8.2ffba","group":"3472749f.3e774c","order":2,"wid  
th":27,"height":8,"name":"","label":"Chatbot","format":{{msg.payload}}","layout  
":"col-center","x":610,"y":360,"wires":[]},{"id":"5c85071.f15f1f8","type":"debug","z":"a  
4d333d8.2ffba","name":"","active":true,"tosidebar":true,"console":false,"tostatus":  
false,"complete":"payload","targetType":"msg","x":680,"y":80,"wires":[]},{"id":"6ba  
bfa52.a49da4","type":"debug","z":"a4d333d8.2ffba","name":"","active":true,"toside  
bar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg  
","x":680,"y":160,"wires":[]},{"id":"c315575b.2a7f48","type":"watson-conversation-v1  
","z":"a4d333d8.2ffba","name":"","workspaceid":"c095dcec-77bd-4732-b647-7e407c  
90c1c0","multiuser":false,"context":false,"empty-payload":false,"service-endpoin  
t":"https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/1d1e6d57-fb4e-  
418e-8c7c-626f63f8e869","timeout":"","optout-learning":false,"x":380,"y":100,"wire  
s":[["c067b3b6.f4f74","5c85071.f15f1f8"]],{"id":"3472749f.3e774c","type":"ui_group",  
z":"","name":"Default","tab":"c37fbb08.7c0d78","order":1,"disp":true,"width":27,"c  
ollapse":false},{"id":"c37fbb08.7c0d78","type":"ui_tab","z":"","name":"Customer  
Help Desk","icon":"dashboard","disabled":false,"hidden":false}].
```