

Project Report

Name : Manav Jain (manavmessi25@gmail.com)

Title : Intelligent Customer Help Desk With Smart
Document Understanding

Category : Artificial Intelligence

Internship at smartinternz.com@2020

1	INTRODUCTION
	1.1 Overview
	1.2 Purpose
2	LITERATURE SURVEY
	2.1 Existing problem
	2.2 Proposed solution
3	THEORITICAL ANALYSIS
	3.1 Block diagram
	3.2 Hardware / Software designing
4	EXPERIMENTAL INVESTIGATIONS
5	FLOWCHART
6	RESULT
7	ADVANTAGES & DISADVANTAGES
8	APPLICATIONS
9	CONCLUSION
10	FUTURE SCOPE
11	BIBILOGRAPHY
	APPENDIX
	A. Source code
	B. Reference

1.Introduction

1.1 Overview

We will be able to write an application that leverages multiple Watson AI Services (Discovery , Assistant, Cloud function and Node Red). By the end of the project, we'll learn best practices of combining Watson services, and how they can build interactive information retrieval systems with Discovery + Assistant.

- Project Requirements: Python, IBM Cloud, IBM Watson
- Functional Requirements: IBM cloud
- Technical Requirements: AI,ML,WATSON AI,PYTHON
- Software Requirements: Watson assistant, Watson discovery.
- Project Deliverables: Smartinternz Internship
- Project Team: Manav Jain
- Project Duration:19 days

1.2 Purpose

The typical customer care chatbot can answer simple questions, such as store locations and hours, directions, and maybe even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn't valid or offer to speak to a real person.

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device's owners manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owners manual to help solve our customers' problems.

To take it a step further, the project shall use the Smart Document Understanding feature of Watson Discovery to train it on what text in the

owners manual is important and what is not. This will improve the answers returned from the queries.

1.2.1 Scope of Work

- Create a customer care dialog skill in Watson Assistant
- Use Smart Document Understanding to build an enhanced Watson Discovery collection
- Create an IBM Cloud Functions web action that allows Watson Assistant to post queries to Watson Discovery
- Build a web application with integration to all these services & deploy the same on IBM Cloud Platform 2.

2.LITERATURE SURVEY

2.1 Existing problem

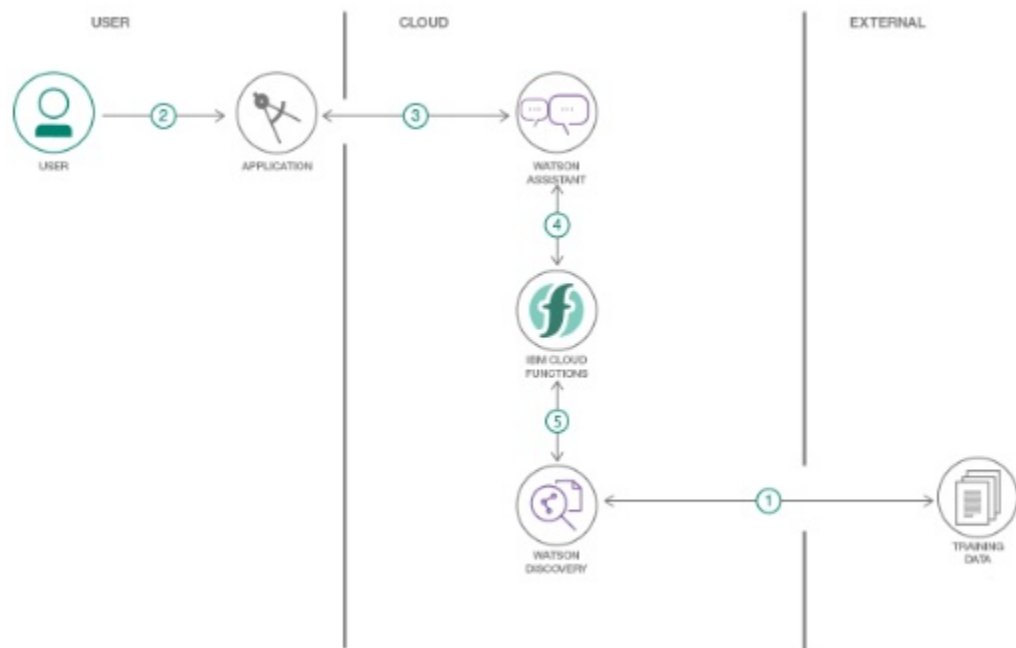
Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like “try again”, “I don’t understand”, “will you repeat again”, and so on... and directs customer to customer agent but a good customer Chatbot should minimize involvement of customer agent to chat with customer to clarify his/her doubts. So to achieve this we should include an virtual agent in chatbot so that it will take care of real involvement of customer agent and customer can clarifies his doubts with fast chatbots.

2.2 Proposed solution

For the above problem to get solved we have to put an virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should trained from some insight records based company background so it can answer queries based on the product or related to company. In this project I used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED

3.THEORITICAL ANALYSIS

3.1 Block/Flow Diagram



1. The document is annotated using Watson Discovery SDU
2. The user interacts with the backend server via the app UI. The frontend app UI is a chatbot that engages the user in a conversation.
3. Dialog between the user and backend server is coordinated using a Watson Assistant dialog skill.
4. If the user asks a product operation question, a search query is passed to a predefined IBM Cloud Functions action.
5. The Cloud Functions action will query the Watson Discovery service and return the results.

3.2 Hardware / Software designing

1. Create IBM Cloud services
2. Configure Watson Discovery
3. Create IBM Cloud Functions action
4. Configure Watson Assistant
5. Create flow and configure node
6. Deploy and run Node Red app.

4.EXPERIMENTAL INVESTIGATIONS

1.Create IBM Cloud services

Create the following services:

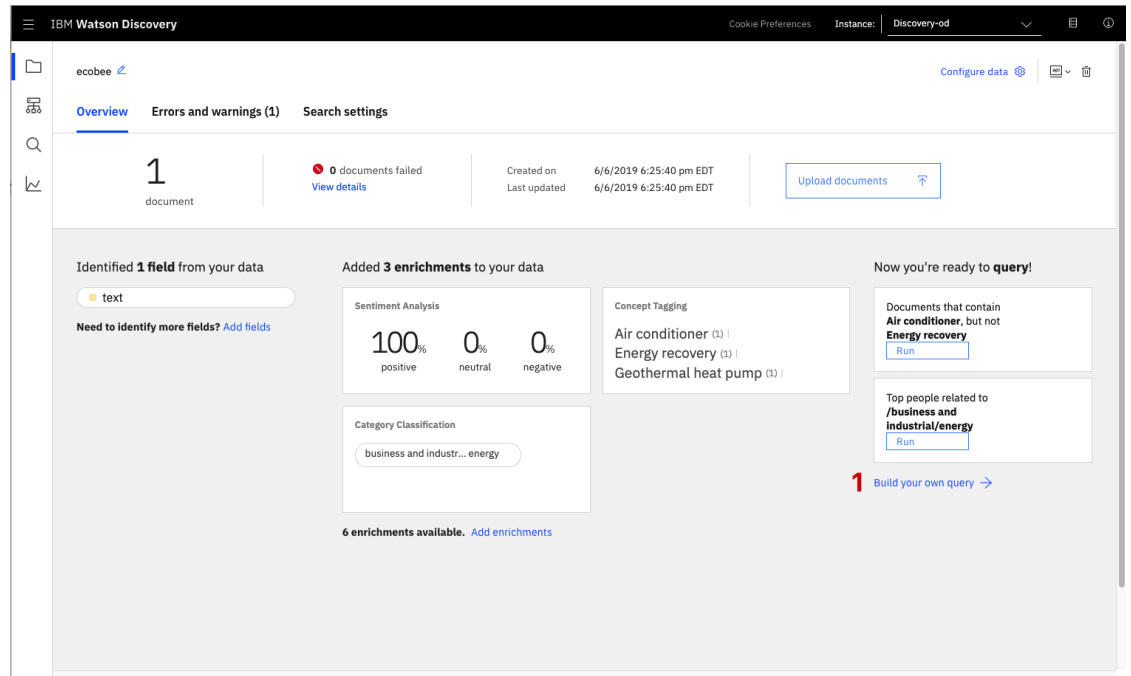
- Watson Discovery
- Watson Assistant
- Node Red

2. Configure Watson Discovery

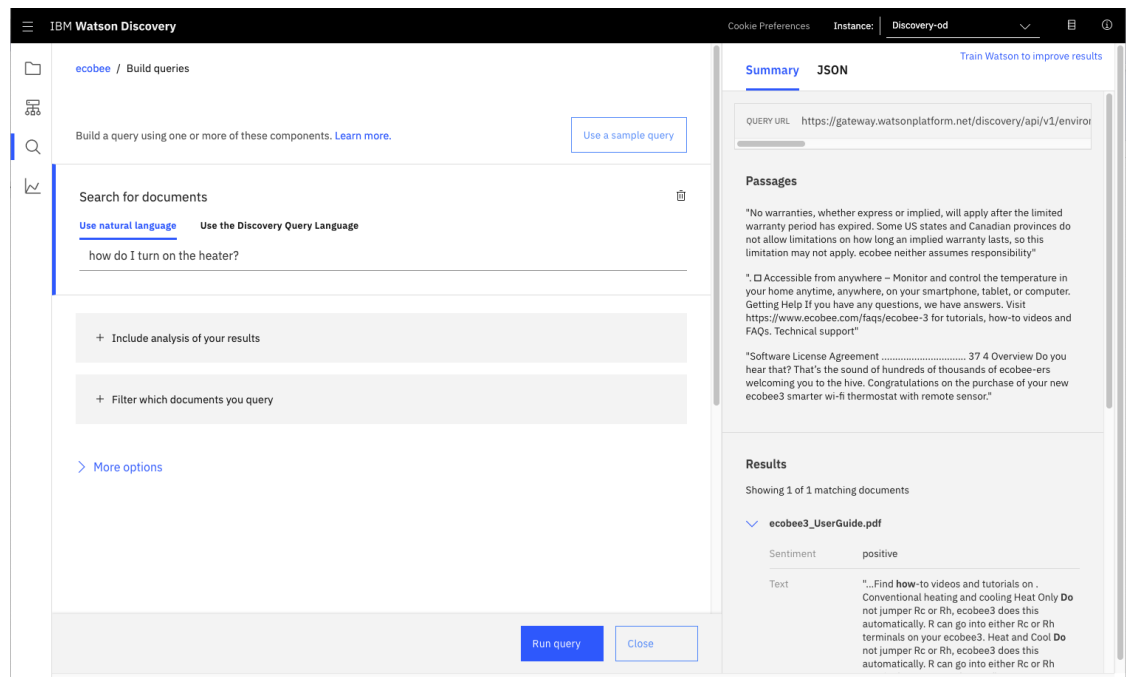
Import the document Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the ecobee3_UserGuide.pdf file located in the data directory of your local repo.

The Ecobee is a popular residential thermostat that has a wifi interface and multiple configuration options.

Before applying SDU to our document, lets do some simple queries on the data so that we can compare it to results found after applying SDU.



Click the Build your own query [1] button.

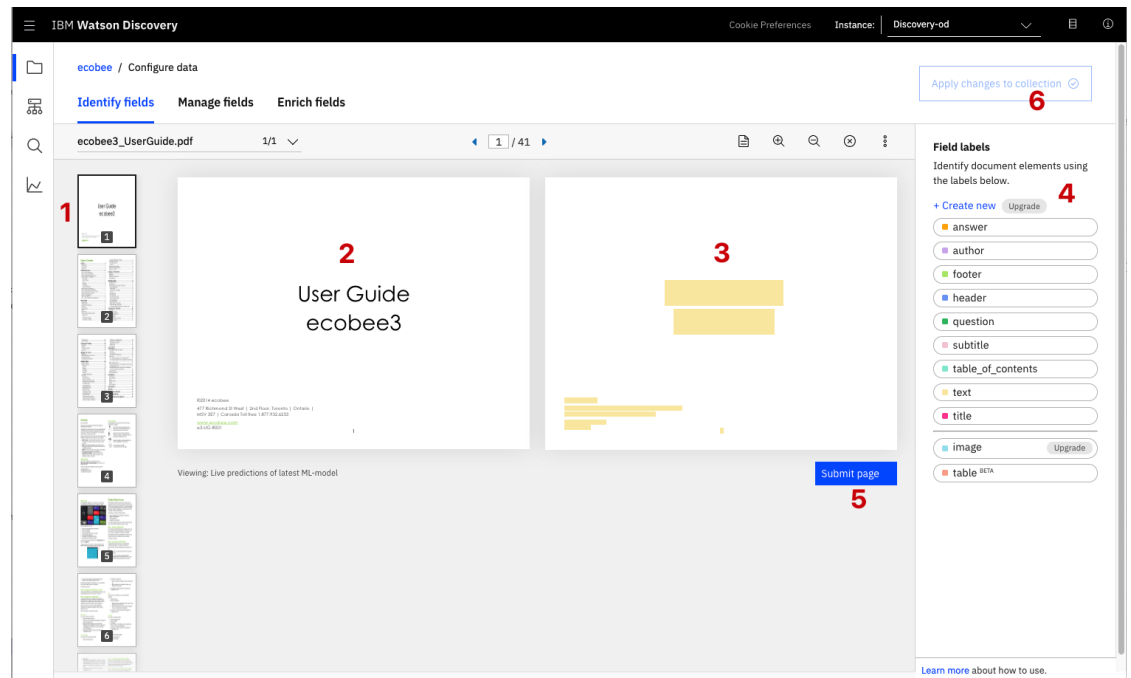


Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

Annotate with SDU

Now let's apply SDU to our document to see if we can generate some better query responses. From the Discovery collection panel, click the Configure data button (located in the top right corner) to start the SDU process.

Here is the layout of the Identify fields tab of the SDU annotation panel:



The goal is to annotate all of the pages in the document so Discovery can learn what text is important, and what text can be ignored.

[1] is the list of pages in the manual. As each is processed, a green check mark will appear on the page.

[2] is the current page being annotated.

[3] is where you select text and assign it a label.

[4] is the list of labels you can assign to the page text.

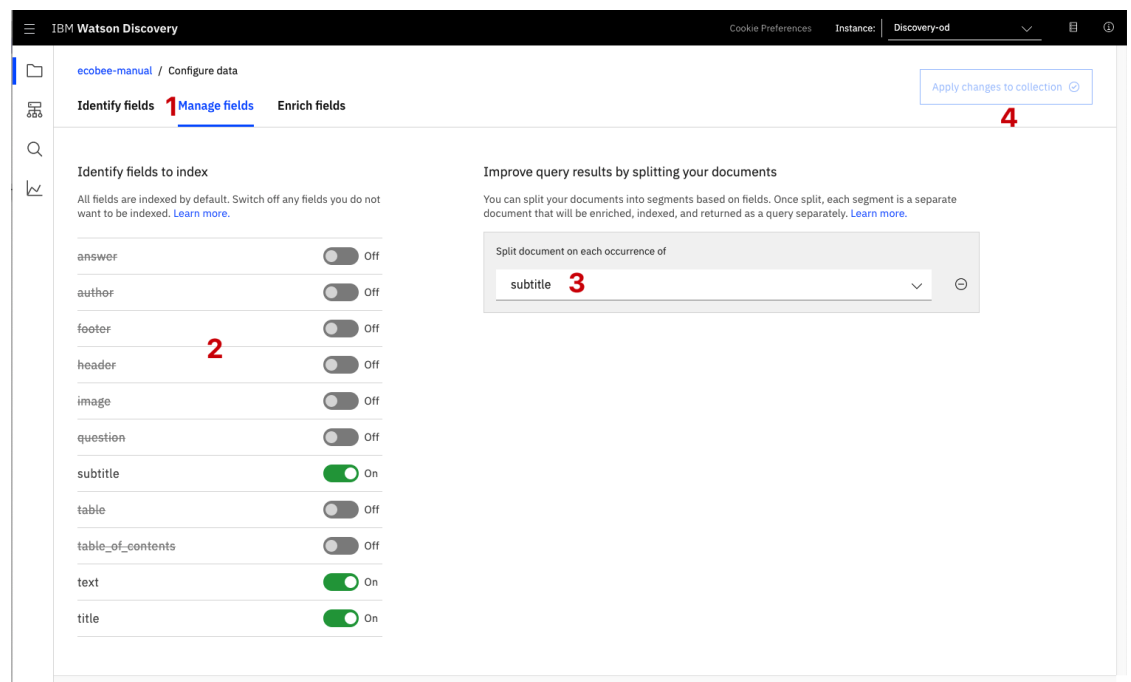
Click [5] to submit the page to Discovery.

Click [6] when you have completed the annotation process.

As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit

[5] to acknowledge it is correct. The more pages you annotate, the better the model will be trained. For this specific owner's manual, at a minimum, it is suggested to mark the following: The main title page as title The table of contents (shown in the first few pages) as table_of_contents All headers and sub-headers (typed in light green text) as a subtitle All page numbers as footers All warranty and licensing information (located in the last few pages) as a footer All other text should be marked as text. Once you click the Apply changes to collection button [6], you will be asked to reload the document. Choose the same owner's manual .pdf document as before.

Next, click on the Manage fields [1] tab.



- [2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.
- [3] is telling Discovery to split the document apart, based on subtitle.
- Click [4] to submit your changes.

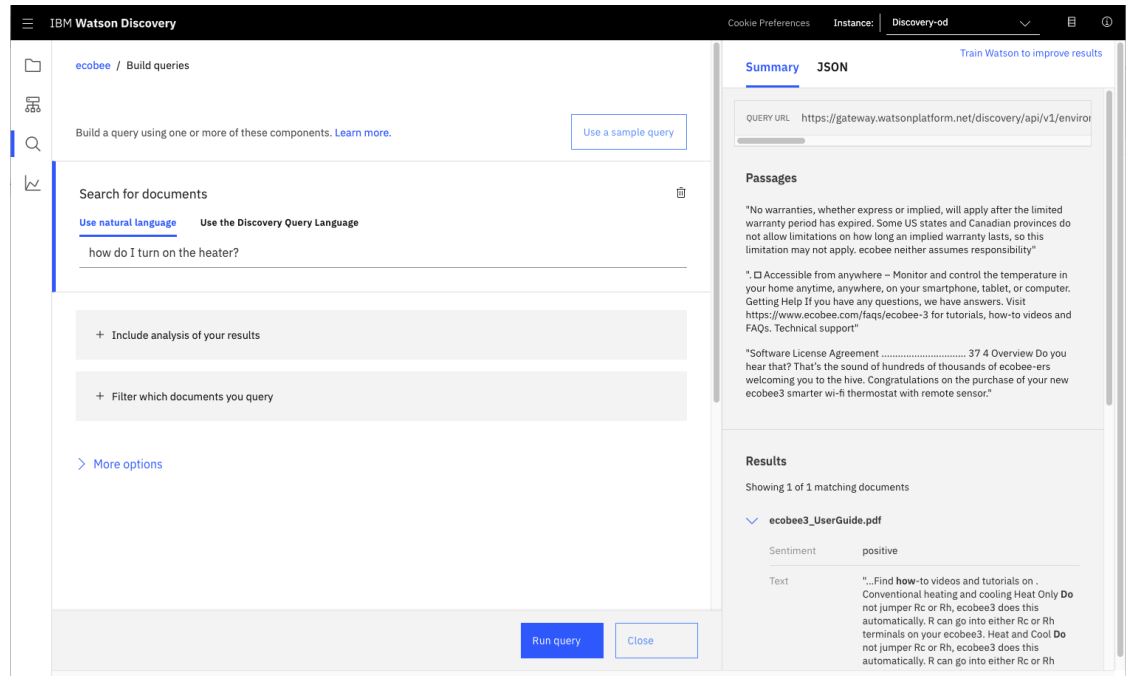
Once again, you will be asked to reload the document.

Now, as a result of splitting the document apart, your collection will look very

different:

The screenshot displays the IBM Watson Discovery user interface. At the top, the header includes 'IBM Watson Discovery', 'Cookie Preferences', 'Instance: Discovery-01', and navigation icons. The left sidebar contains icons for document management, search, and analysis. The main content area is titled 'ecobee-manual' and shows '130 documents'. It includes tabs for 'Overview', 'Errors and warnings (130)', and 'Search settings'. A status bar indicates '0 documents failed' and provides creation and update timestamps. An 'Upload documents' button is present. The interface is divided into three main sections: 'Identified 5 fields from your data' (listing footer, subtitle, table_of_contents, text, and title), 'Added 4 enrichments to your data' (showing Entity Extraction, Sentiment Analysis, Concept Tagging, and Category Classification results), and 'Now you're ready to query!' (offering options to run queries on negative sentiment, specific terms like 'Heat' and 'Internet', or top entities). A 'Build your own query' link is at the bottom right.

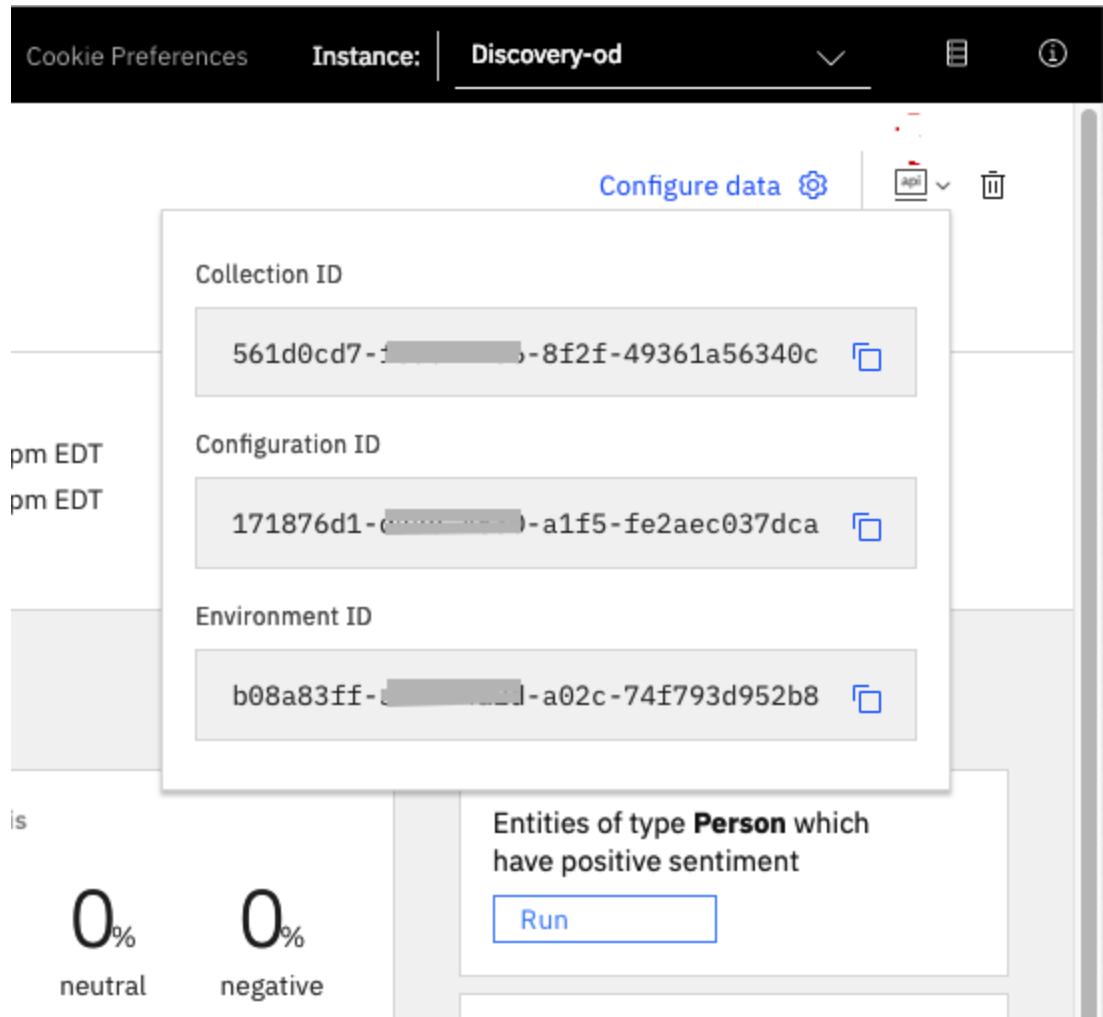
Return to the query panel (click **Build your own query**) and see how much better the results are.



Store credentials for future use

In upcoming steps, you will need to provide the credentials to access your Discovery collection. The values can be found in the following locations.

The `Collection ID` and `Environment ID` values can be found by clicking the dropdown button [1] located at the top right side of your collection panel:



For credentials, return to the main panel of your Discovery service, and click the `Service credentials` [1] tab:

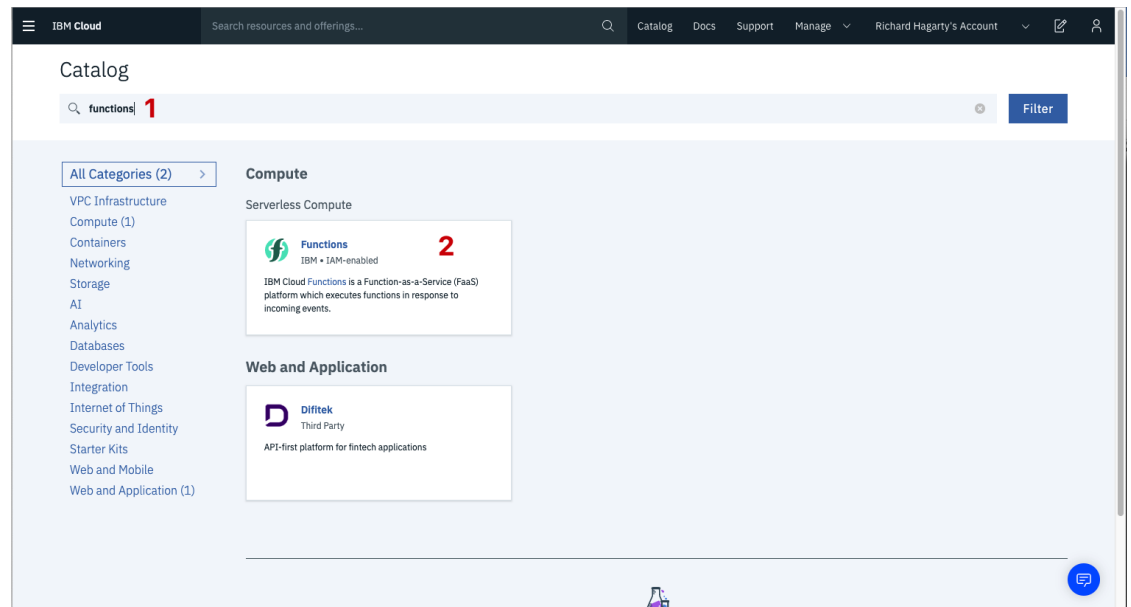
Click the `view credentials` [2] drop-down menu to view the IAM `apikey` [3] and `URL` endpoint [4] for your service.

3. Create IBM Cloud Functions action

Now let's create the `web action` that will make queries against our Discovery collection.

Start the `IBM Cloud Functions` service by selecting `Create Resource` from the `IBM Cloud` dashboard. Enter `functions` as the filter [1], then select the

Functions card [2]:



From the Functions main panel, click on the Actions tab. Then click on Create.

From the Create panel, select the Create Action option.

On the Create Action panel, provide a unique Action Name [1], keep the default package [2], and select the Node.js 10 [3] runtime.

Click the create button [4] to create the action.

IBM Cloud Search resources and offerings...

Functions

Getting Started

Actions

Triggers

APIs

Monitor

Logs

Namespace Settings

Create Action

Actions contain your function code and are invoked by events or REST API calls.

[Learn more about Actions](#)

[Learn more about Packages](#)

Action Name: disco-action-2

Enclosing Package: (Default Package)

Runtime: Node.js 10

Looking for Java, .NET or Docker? [Docker](#) Actions can be created with the [CLI](#)

Cancel Previous Create

Once your action is created, click on the `code` tab [1]:

disco-action Web Action

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Code Node.js 10

Change Input Invoke

```

1- /**
2-  *
3-  * @param {object} params
4-  * @param {string} params.lam_apikey
5-  * @param {string} params.url
6-  * @param {string} params.username
7-  * @param {string} params.password
8-  * @param {string} params.environment_id
9-  * @param {string} params.collection_id
10-  * @param {string} params.configuration_id
11-  * @param {string} params.input
12-  * @return {object}
13-  */
14-
15-
16-
17- const assert = require('assert');
18- const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19-
20- /**
21-  *
22-  * main() will be run when you invoke this action
23-  *
24-  * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
25-  *
26-  * @return The output of this action, which must be a JSON object.
27-  */
28-
29- function main(params) {
30-   return new Promise(function (resolve, reject) {
31-
32-     let discovery;
33-
34-     if (params.lam_apikey){
35-       discovery = new DiscoveryV1({
36-         'iam_apikey': params.lam_apikey,
37-         'url': params.url,
38-         'username': params.username,
39-         'password': params.password,
40-         'environment_id': params.environment_id,
41-         'collection_id': params.collection_id,
42-         'configuration_id': params.configuration_id
43-       });
44-     }
45-
46-     discovery.request(params.input, function (error, data, response) {
47-       if (error) {
48-         reject(error);
49-       } else {
50-         resolve(data);
51-       }
52-     });
53-   });
54- }
55-
56- module.exports = main;
57-

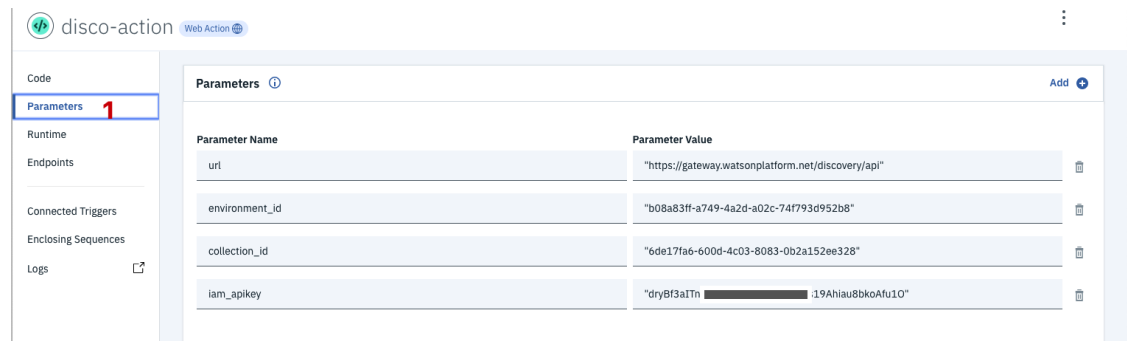
```

In the code editor window [2], cut and paste in the code from the `disco-action.js` file found in the actions directory of your local repo. The

code is pretty straight-forward - it simply connects to the Discovery service, makes a query against the collection, then returns the response.

If you press the `Invoke` button [3], it will fail due to credentials not being defined yet. We'll do this next.

Select the `Parameters` tab [1]:



Add the following keys:

- url
- environment_id
- collection_id
- iam_apikey

For values, please use the values associated with the Discovery service you created in the previous step.

Note: Make sure to enclose your values in double quotes.

Now that the credentials are set, return to the `Code` panel and press the `Invoke` button again. Now you should see actual results returned from the Discovery service:

IBM Cloud | Search resources and offerings... | Catalog | Docs | Support | Manage | Richard Hagarty's Account

Functions / Actions / disco-action | Namespace: IBM Cloud Storage_DSX-journey-2 | Location: Dallas

disco-action Web Action

Code Node.js 10

```
1 - /**
2  *
3  * @param {object} params
4  * @param {string} params.iam_apikey
5  * @param {string} params.url
6  * @param {string} params.username
7  * @param {string} params.password
8  * @param {string} params.environment_id
9  * @param {string} params.collection_id
10 * @param {string} params.configuration_id
11 * @param {string} params.input
12 *
13 * @return {object}
14 *
15 */
16
17 const assert = require('assert');
18 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
19
20 /**
21 *
22 * main() will be run when you invoke this action
23 *
24 * @param Cloud Functions actions accept a single parameter, which must be a JSON object.
25 *
26 * @return The output of this action, which must be a JSON object.
27 *
28 */
29 function main(params) {
30   return new Promise(function (resolve, reject) {
31
32     let discovery;
33
34     if (params.iam_apikey){
35       discovery = new DiscoveryV1({
36         'iam_apikey': params.iam_apikey,
37         'url': params.url,
38         'version': '2019-03-25'
39       });
40     }
41   });
42 }
```

Change Input | Invoke

Activations

disco-action 1050 ms 6/6/2019, 10:45:14

Activation ID: e1bfc0ff21544c95bfc0ff21549c85a1

Results:

```
{
  "matching_results": 14,
  "passages": [],
  "results": {
    "enriched_text": {
      "categories": {
        "label": "/technology and computing/operating systems",
        "score": 0.842265
      },
      "label": "/technology and computing/hardware/computer",
      "score": 0.835879
    },
    "label": "/technology and computing/hardware/computer/peripherals/computer monitors",
    "score": 0.832234
  },
  "concepts": [
    {
      "dbpedia_resource": "http://dbpedia.org/resource/IPhone",
      "relevance": 0.917306,
      "text": "IPhone"
    },
    {
      "dbpedia_resource": "http://dbpedia.org/resource/Personal_digital_assistant",
      "relevance": 0.887088,
      "text": "Personal digital assistant"
    }
  ]
}
```

Next, go to the Endpoints panel [1]:

IBM Cloud | Search resources and offerings... | Catalog | Docs | Support | Manage | 1984008 - Richard ...

Functions / Actions / disco-action | Namespace: IBM Cloud Storage_DSX-journey-2 | Location: Dallas

disco-action Web Action

Code

Parameters

Runtime

Endpoints 1

Connected Triggers

Enclosing Sequences

Logs

Web Action

☒ Enable as Web Action | Allow your Cloud Functions actions to handle HTTP events. Learn more about Web Actions.

☐ Raw HTTP handling | When enabled your Action receives requests in plain text instead of a JSON body

HTTP METHOD	AUTH	URL
ANY	Public	3 https://us-south.functions.cloud.ibm.com/api/v1/web/IBM%20Cloud%20Storage_DSX-journey-2/default/disco-action

REST API

HTTP METHOD	AUTH	URL
POST	API-KEY	https://us-south.functions.cloud.ibm.com/api/v1/namespaces/IBM%20Cloud%20Storage_DSX-journey-2/actions/disco-action

CURL

4 curl -u API-KEY -X POST https://us-south.functions.cloud.ibm.com/api/v1/namespaces/IBM%20Cloud%20Storage_DSX-journey-2/actions/disco-action?blocking=true

Click the checkbox for Enable as web Action [2]. This will generate a public endpoint URL [3].

Take note of the URL value [3], as this will be needed by Watson Assistant in a

future step.

To verify you have entered the correct Discovery parameters, execute the provided `curl` command [4]. If it fails, re-check your parameter values.

NOTE: An IBM Cloud Functions service will not show up in your dashboard resource list. To return to your defined `Action`, you will need to access Cloud Functions by selecting `Create Resource` from the main dashboard panel (as shown at the beginning of this step).

4. Configure Watson Assistant

As shown below, launch the `watson Assistant` tool and create a new dialog skill. Select the `Use sample skill` option as your starting point.

This dialog skill contains all of the nodes needed to have a typical call center conversation with a user.

Add new intent

The default customer care dialog does not have a way to deal with any questions involving outside resources, so we will need to add this.

Create a new `intent` that can detect when the user is asking about operating the Ecobee thermostat.

From the `Customer Care Sample Skill` panel, select the `Intents` tab.

Click the `Create intent` button.

Name the intent `#Product_Information`, and at a minimum, enter the following example questions to be associated with it.

Name the node "Ask about product" [1] and assign it our new intent [2].

The screenshot shows the IBM Watson Assistant interface. At the top, there's a dark blue header with 'IBM Watson Assistant'. Below it, a breadcrumb trail shows 'Skills /'. The main title is 'Customer Care Sample Skill copy' with a subtitle 'Sample simple customer service skill to get you started.' Below the title is a navigation bar with tabs: 'Intents', 'Entities', 'Dialog' (highlighted with a red '1'), 'Analytics', 'Options', 'Versions', and 'Content Catalog'. The 'Dialog' tab displays a list of nodes. The nodes are: 'Directions and location' (with a right arrow icon), 'Make an appointment', 'Transfer to agent' (with a right arrow icon), 'Small Talk' (with a folder icon), and 'anything_else'. Each node shows its name, a unique identifier, and statistics. A red '2' points to the three-dot menu icon next to the 'Small Talk' node. A context menu is open over the 'anything_else' node, with a red '3' pointing to the 'Add node below' option. The context menu options are: 'Add node to folder', 'Add node above', 'Add node below', 'Add folder', 'Move', 'Duplicate', 'Jump to', and 'Delete'.

Skills /

Customer Care Sample Skill copy
Sample simple customer service skill to get you started.

Intents Entities **1** Dialog Analytics Options Versions Content Catalog

Directions and location
#Customer_Care_Store_Location
3 Responses / 0 Context Set / Skip user input / Returns

Make an appointment
#Customer_Care_Appointments
3 Responses / 7 Context Set / 5 Slots / Does not return

Transfer to agent
#General_Connect_to_Agent
1 Responses / 0 Context Set / Does not return

Small Talk
3 Dialog nodes / No digressions

anything_else
1 Responses / 0 Context Set / Returns

Add node to folder
Add node above
Add node below **3**
Add folder
Move
Duplicate
Jump to
Delete

This means that if Watson Assistant recognizes a user input such as "how do I set the time?", it will direct the conversation to this node.

Enable webhook from Assistant

Set up access to our WebHook for the IBM Cloud Functions action you created in Step #4.

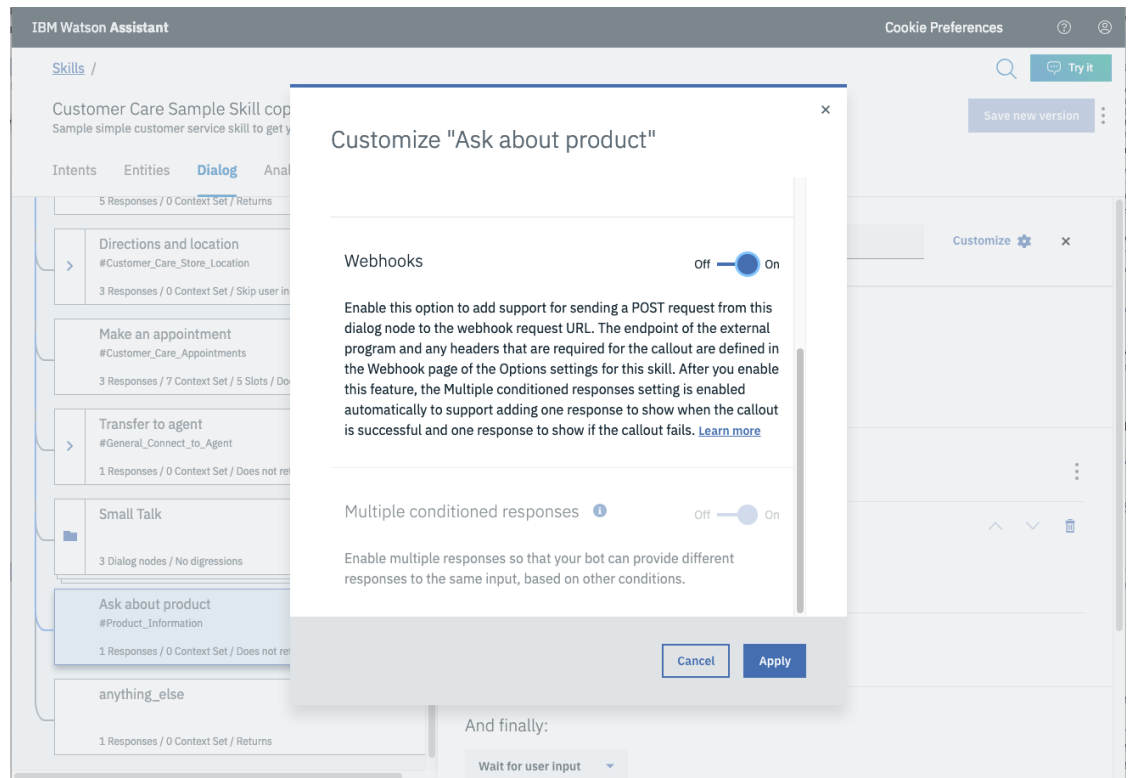
Select the **options** tab [1]:

The screenshot shows the IBM Watson Assistant interface. At the top, the header reads 'IBM Watson Assistant' and 'Cookie Preferences'. Below the header, the breadcrumb is 'Skills /'. The main title is 'Customer Care Sample Skill for Disco' with a subtitle 'Sample simple customer service skill to get you started.' On the right, there are buttons for 'Try it' and 'Save new version'. A navigation bar includes 'Intents', 'Entities', 'Dialog', 'Analytics', 'Options' (highlighted with a red '1'), 'Versions', and 'Content Catalog'. On the left, a sidebar shows 'Webhooks' (selected), 'Autocorrection', and 'System Entities'. The main content area is titled 'Webhooks' and contains a description: 'A webhook is a mechanism that allows your dialog skill to call an external API when specific dialog nodes are triggered. Specify the request URL for the external API you want to be able to invoke. You will then be able to access this URL from within the dialog editor. [Learn more](#)'. Below this is a 'URL' section with a red '2' and the input field containing 'https://us-south.functions.cloud.ibm.com/api/v1/web/IBM%20Cloud%20Stor'. Underneath is a 'Headers' section with the instruction 'Add HTTP headers for authorization or any other parameters required for invoking the specified request URL.' and a table with columns 'HEADER NAME' and 'HEADER VALUE'. At the bottom of the headers section are links 'Add header' and 'Add authorization'. A 'Next step' section at the bottom states: 'To trigger this webhook from an individual dialog node, enable the webhook from the Customize page in node details. [Go to dialog](#)'.

Enter the public URL endpoint for your action [2].

Important: Add **.json** to the end of the URL to specify the result should be in JSON format.

Return to the **Dialog** tab, and click on the **Ask about product** node. From the details panel for the node, click on **Customize**, and enable **Webhooks** for this node:



Click **Apply**.

The dialog node should have a **Return variable** [1] set automatically to `$webhook_result_1`. This is the variable name you can use to access the result from the Discovery service query.

IBM Watson Assistant

Customer Care Sample Skill for Disco
Sample simple customer service skill to get you started.

Intents Entities **Dialog** Analytics Options Versions Content Catalog

#Customer_Care_Store_Hours
5 Responses / 0 Context Set / Returns

Directions and location
#Customer_Care_Store_Location
3 Responses / 0 Context Set / Skip user input / Returns

Make an appointment
#Customer_Care_Appointments
3 Responses / 7 Context Set / 5 Slots / Does not return

Transfer to agent
#General_Connect_to_Agent
1 Responses / 0 Context Set / Does not return

Small Talk
3 Dialog nodes / No digressions

Ask about product
#Product_Information
2 Responses / 0 Context Set / Does not return

anything_else
1 Responses / 0 Context Set / Returns

Ask about product

If assistant recognizes:

#Product_Information

Then callout to my webhook:

Parameters

KEY	VALUE
2 input	"<?input.text?>"

Add parameter

Return variable

1 \$webhook_result_1

You will also need to pass in the users question via the parameter `input` [2]. The key needs to be set to the value:





```
"<?input.text?>"
```

If you fail to do this, Discovery will return results based on a blank query.

Optionally, you can add these responses to aid in debugging:

Return variable`$webhook_result_1`

Then respond with

	IF ASSISTANT RECOGNIZES	RESPOND WITH		
1	<code>\$webhook_result_1</code>	<code>\$webhook_result_1</code>		
2	<code>anything_else</code>	Try again later		

[Add response](#) 

Test in Assistant Tooling

From the **Dialog** panel, click the **Try it** button located at the top right side of the panel.

Enter some user input:

Try it out

Clear

Manage Context 3



Hello, I'm a demo customer care virtual assistant to show you the basics. I can help with directions to my store, hours of operation and booking an in-store appointment



hello

#General_Greetings



Hello. Good evening



how do I turn on the heater?

#Product_Information



[{"document_id":"3a5efee70d8c-c9d70e2b94d22c15e2d1_2","end_offset":2791,"field":"text","passage_score":6.752501692678998,"passage_text":"Specify what the heat pump runs when the O/B Reversing Valve is engaged: On Cool runs cooling when O/B engages (most cases), or On Heat runs heating when O/B engages. 4. Touch Next. You will be returned to the Equipment configu-



Note that the input "how do I turn on the heater?" has triggered our Ask about product dialog node, which is indicated by the #Product_Information response.

And because we specified that \$webhook_result_1.passages be the response, that value is displayed also.

You can also verify that the call was successfully completed by clicking on the Manage Context button at the top right. The response from the Discovery query will be stored in the \$webhook_result_1 variable:

Context variables ⓘ

×

\$Enter variable name

\$timezone

⊖

"America/Los_Angeles"

\$no_reservation

⊖

true

\$my_creds

⊖

{"user":"7a4d1a77-2429-43b1-b6ed-a

\$webhook_result_1

⊖

{"activationId":"774b9c41df9546158b

5.Create flow and configure node:

Integration of watson assistant in Node-RED

Double-click on the Watson assistant node

Give a name to your node and enter the username, password and workspace id of your Watson assistant service

Edit assistant node

Delete

Cancel

Done

Properties

Name

Name

Username

Username

Password

Password

API Key

.....

Service Endpoint

https://api.eu-gb.assistant.watson.cloud.ibm.com

Workspace ID

bd7b6cad-cc41-4eec-87e0-7529a4148943

Timeout Period

Leave empty to disable

☐ Save context

☐ Enabled

- After entering all the information click on Done
- Drag inject node on to the flow from the Input section
- Drag Debug on to the flow from the output section
- Double-click on the inject node
- Select the payload as a string
- Enter a sample input to be sent to the assistant service and click on done
- Connect the nodes as shown below and click on Deploy

- Open Debug window as shown below
- Click on the button to send input text to the assistant node
- Observe the output from the assistant service node
- The Bot output is located inside "output.text"
- Drag the function node to parse the JSON data and get the bot response
- Double click on the function node and enter the JSON parsing code as shown below and click on done
- Connect the nodes as shown below and click on Deploy
- Re-inject the flow and observe the parsed output
- For creating a web application
- UI we need "dashboard" nodes which should be installed manually.
- Go to navigation pane and click on manage palette
- Click on install
- Search for "node-red-dashboard" and click on install and again click on install on the prompt
- The following message indicates dashboard nodes are installed, close the manage palette
- Search for "Form" node and drag on to the flow
- Double click on the "form" node to configure
- Click on the edit button to add the "Group" name and "Tab" name
- Click on the edit button to add tab name to web application
- Give sample tab name and click on add do the same thing for the group
- Give the label as "Enter your input", Name as "text" and click on Done
- Drag a function node, double-click on it and enter the input parsing code as shown below

Edit function node

Delete

Cancel

Done

⚙️ Properties

⚙️

📄

🖨️

🏷️ Name

📄 ▼

🔑 Function

↗️

1 msg.payload = msg.payload.input;

2 return msg;

🔗 Outputs

☐ Enabled

- Click on done
- Connect the form output to the input of the function node and output of the function to input of assistant node
- Search for “text” node from the “dashboard” section
- Drag two “text” nodes on to the flow
- Double click on the first text node, change the label as “You” and click

on Done

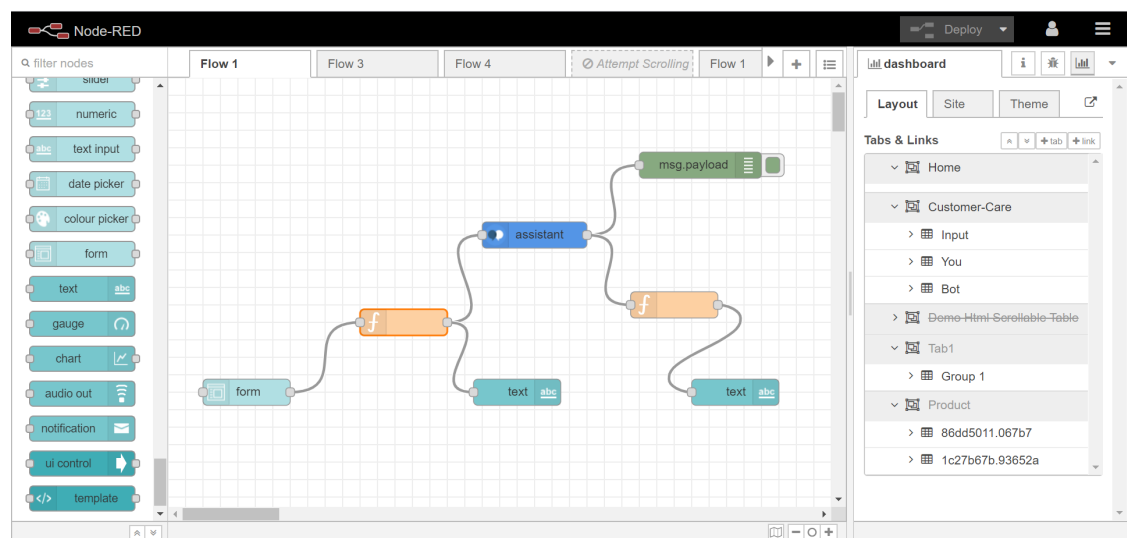
- Double click on the second text node, change the label as “Bot” and click on Done
- Connect the output of “input parsing” function node to “ You” text node and output of “Parsing” function node to the input of “Bot” text node
- Click on Deploy

5.FLOWCHART

At first go to manage pallette and install dashboard.

Now,Create the flow with the help of following node:

- ✓ Inject
- ✓ Assistant
- ✓ Debug
- ✓ Function
- ✓ Ui_Form
- ✓ Ui_Text



6.RESULTS

Finally our Node-RED dash board integrates all the components and displayed in the Dashboard UI by typing URL -

<https://node-red--smart.eu-gb.mybluemix.net/ui/#!/1?socketid=WIBFLUijr5ajh>

[1HZAAAN](#) in browser.

Customer-Care

Input

Enter the input

hey

SUBMITCANCEL

You

hey

Bot

Hello. Good evening !

Customer-Care

Input

Enter the input

How to turn on heater?

SUBMITCANCEL

You

How to turn on heater?

Bot

This menu lets you test the wiring and connections of the devices connected to the thermostat by turning them on or off. The equipment will turn off when you exit the menu. Warning: Compressor protection and minimum run-time features are not enforced while in this mode. The following pages provide wiring diagrams for common HVAC equipment configurations. Need help with your ecobee3 wiring? Find how-to videos and tutorials on . 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 & DISADVANTAGES

Advantages:

- Companies can deploy chatbots to rectify simple and general human queries .
- Reduces man power
- Cost efficient
- No need to divert calls to customer agent and customer agent can look on other works.

Disadvantages:

- Some times chatbot can mislead customers
- Giving same answer for different sentiments.
- Some times cannot connect to customer sentiments and intentions

8.APPLICATIONS

- It can deploy in popular social media applications like facebook,slack,telegram.
- Chatbot can deploy any website to clarify basic doubts of viewers.

9.CONCLUSION

By doing the above procedure and all we successfully created Intelligent helpdesk smart chartbot using Watson assistant, Watson discovery, Node-RED and cloud-functions.

10.FUTURE SCOPE

We can include watson studio text to speech and speech to text services to access the chatbot handsfree. This is one of the future scope of this project.

11. BIBILOGRAPHY

APPENDIX

Source Code

1.Cloud Function(Node.js)

```
/**
 *
 * @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': '2020-05-09'
      });
    }
    else {
      discovery = new DiscoveryV1({
        'username': params.username,
        'password': params.password,
        'url': params.url,
        'version': '2020-05-11'
      });
    }

    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. Node Red (flow.json)

```

[{"id":"fa7ec4cb.ee4338","type":"tab","label":"Flow
1","disabled":false,"info":""},{id":"f2f2649a.0d0d98","t
ype":"debug","z":"fa7ec4cb.ee4338","name":"","active":tru
e,"tosidebar":true,"console":false,"tostatus":false,"comp
lete":"payload","targetType":"msg","x":610,"y":100,"wires
":[ ]},{id":"30b48a44.0807b6","type":"ui_form","z":"fa7ec
4cb.ee4338","name":"","label":"","group":"c8c57d62.c6dab"
,"order":0,"width":"6","height":"6","options":[{"label":"
Enter
the
input","value":"input","type":"text","required":true,"row
s":null}], "formValue":{"input":""},"payload":"","submit":
"submit","cancel":"cancel","topic":"","x":90,"y":360,"wir
es":[["c74f35b3.359468"]]}, {"id":"c74f35b3.359468","type"
:"function","z":"fa7ec4cb.ee4338","name":"","func":"msg.p
ayload
=
msg.payload.input;\nreturn
msg;","outputs":1,"noerr":0,"x":270,"y":280,"wires":[["95
eb3cdb.b8246","d0e9a30e.a7201"]]}, {"id":"245a5441.9688cc"
,"type":"function","z":"fa7ec4cb.ee4338","name":"","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.con
text.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;\n\n\n\n\n\n//
msg.payload.text=\"\";\n//
if(msg.payload.context.webhook_result_1){\n
\n//
msg.payload.text=msg.payload.text+\"\\n\"+msg.payload.con

```

```

text.webhook_result_1.results[0].text;\n//
msg.payload=msg.payload.text;\n//      }\n\n//      else\n//
msg.payload  =  msg.payload.output.text[0];\n//  return
msg;\n\n\n", "outputs":1, "noerr":0, "x":580, "y":260, "wires"
: [[ "6440ccbe.b7cac4" ] ] }, { "id": "95eb3cdb.b8246", "type": "wa
tson-conversation-v1", "z": "fa7ec4cb.ee4338", "name": "", "wo
rkspaceid": "bd7b6cad-cc41-4eec-87e0-7529a4148943", "multiu
ser": false, "context": false, "empty-payload": true, "service-
endpoint": "https://api.eu-gb.assistant.watson.cloud.ibm.c
om/instances/deb5b32c-0147-4100-bc81-c4453eb57fb8", "timeo
ut": "", "optout-learning": false, "x":420, "y":180, "wires": [[
"f2f2649a.0d0d98", "245a5441.9688cc" ] ] }, { "id": "6440ccbe.b7
cac4", "type": "ui_text", "z": "fa7ec4cb.ee4338", "group": "5e9
bf438.4618ec", "order":0, "width": "18", "height": "6", "name":
"", "label": "", "format": "{msg.payload}", "layout": "col-ce
nter", "x":650, "y":360, "wires": [ ] }, { "id": "d0e9a30e.a7201",
"type": "ui_text", "z": "fa7ec4cb.ee4338", "group": "f56ca633.
f95f78", "order":1, "width": "6", "height": "2", "name": "", "lab
el": "", "format": "{msg.payload}", "layout": "col-center", "
x":400, "y":360, "wires": [ ] }, { "id": "c8c57d62.c6dab", "type":
"ui_group", "z": "", "name": "Input", "tab": "b99e7174.45895", "
order":1, "disp": true, "width": "6", "collapse": false }, { "id":
"5e9bf438.4618ec", "type": "ui_group", "z": "", "name": "Bot", "
tab": "b99e7174.45895", "order":3, "disp": true, "width": "18",
"collapse": false }, { "id": "f56ca633.f95f78", "type": "ui_grou
p", "z": "", "name": "You", "tab": "b99e7174.45895", "order":2, "
disp": true, "width": "6", "collapse": false }, { "id": "b99e7174.
45895", "type": "ui_tab", "z": "", "name": "Customer-Care", "ico
n": "dashboard", "disabled": false, "hidden": false } ]

```

Reference

1. https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery
2. <https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started>

3. <https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-node-red/>
4. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
5. <https://github.com/IBM/watson-discovery-sdu-witistat>
6. <https://www.youtube.com/watch?v=Jpr3wVH3FVA>