

Project Report

Name : Mayuri Bhosale(mayuriiibhosale@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: Mayuri Bhosale

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.LITERATURE SURVEY

2.1 Existing problem:

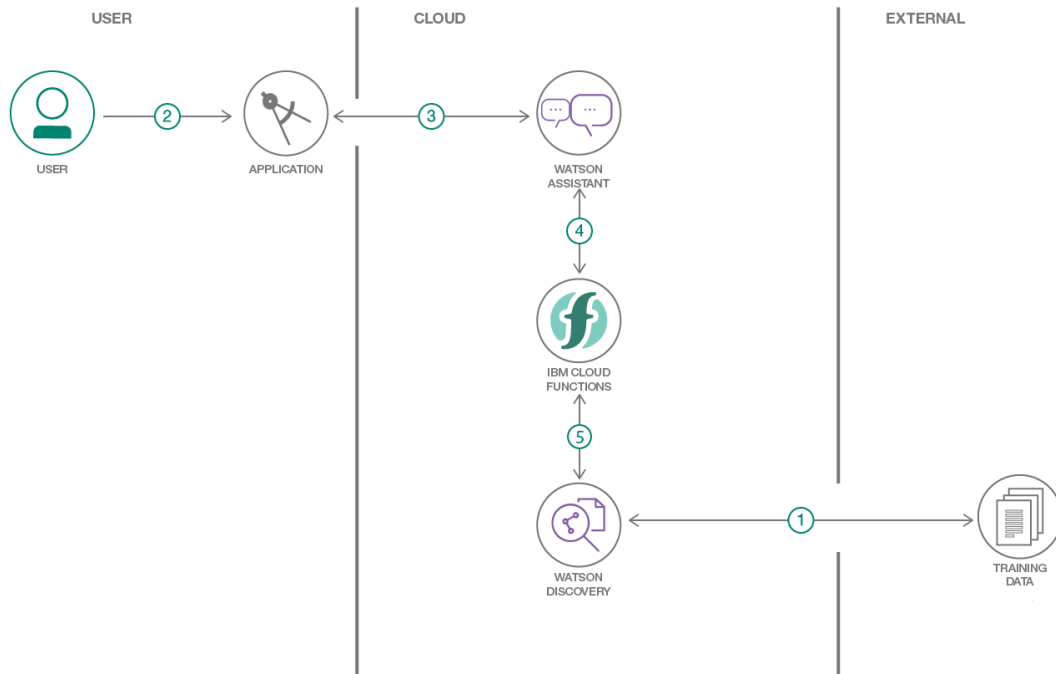
Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like “try again”, “I don’t understand”, “will you repeat again”, and so on... and directs customer to customer agent but a good customer Chatbot should minimize involvement 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.THEROTICAL 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

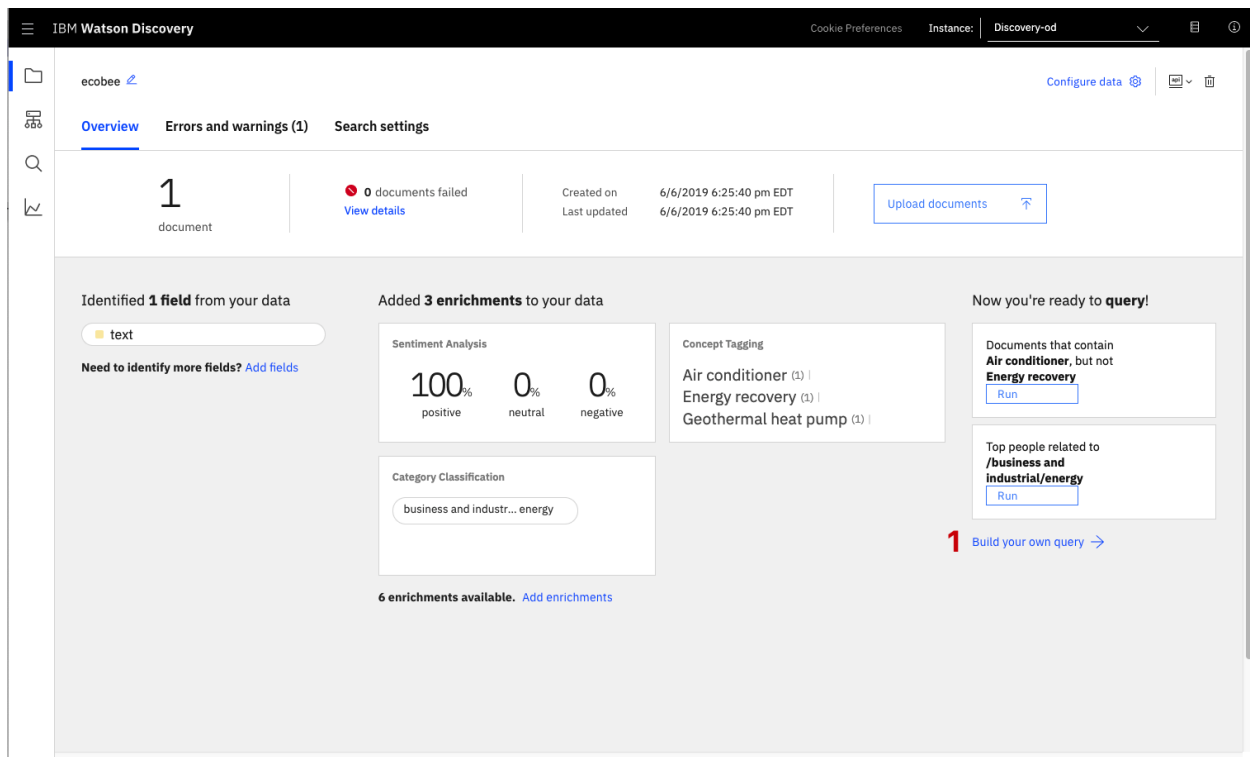
2. Configure Watson Discovery

Import the document

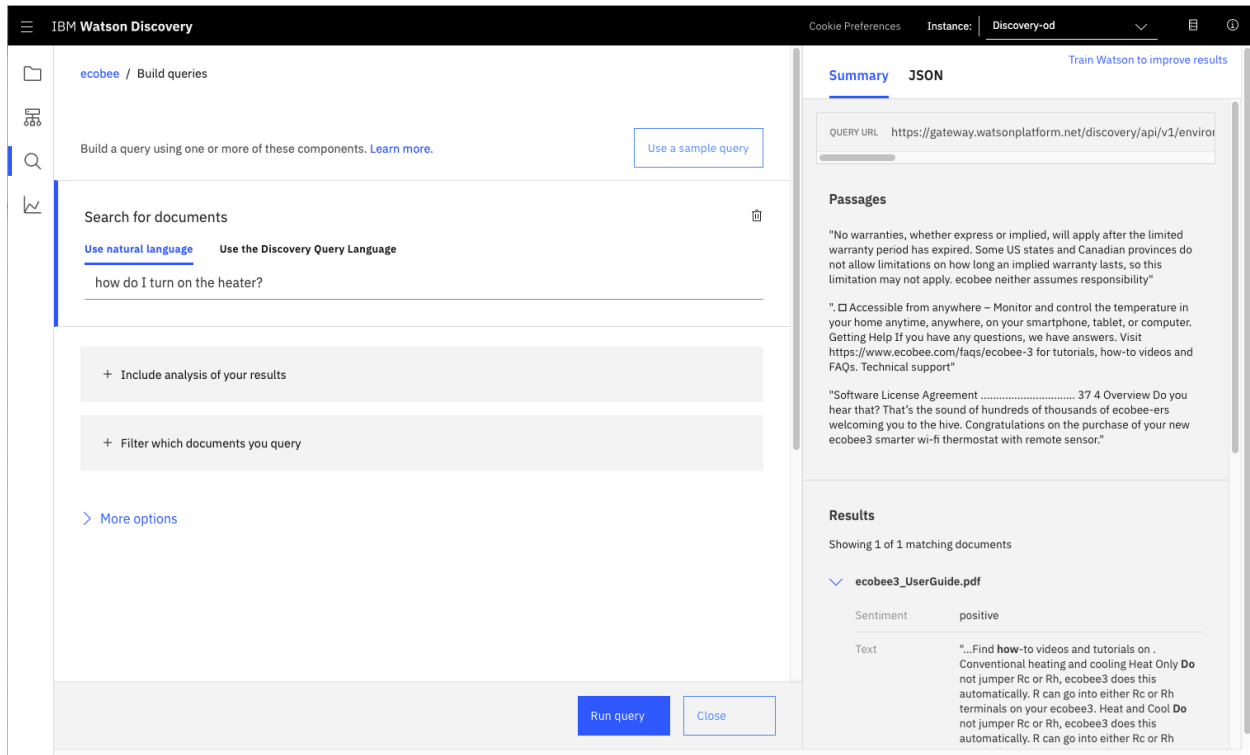
Launch the Watson Discovery tool and create a new data collection by selecting the Upload your own data option. Give the data collection a unique name. When prompted, select and upload the ecobee3_UserGuide.pdf file located in the data directory of your local repo.

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

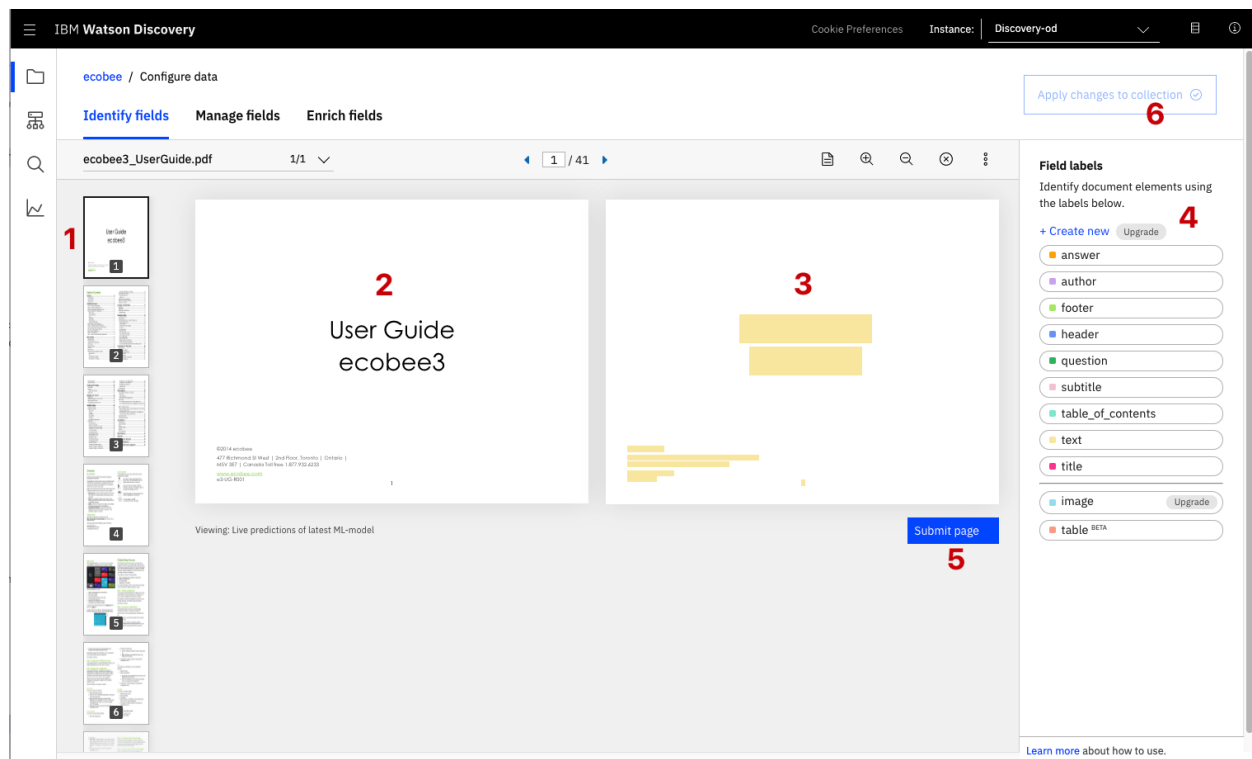


Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

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

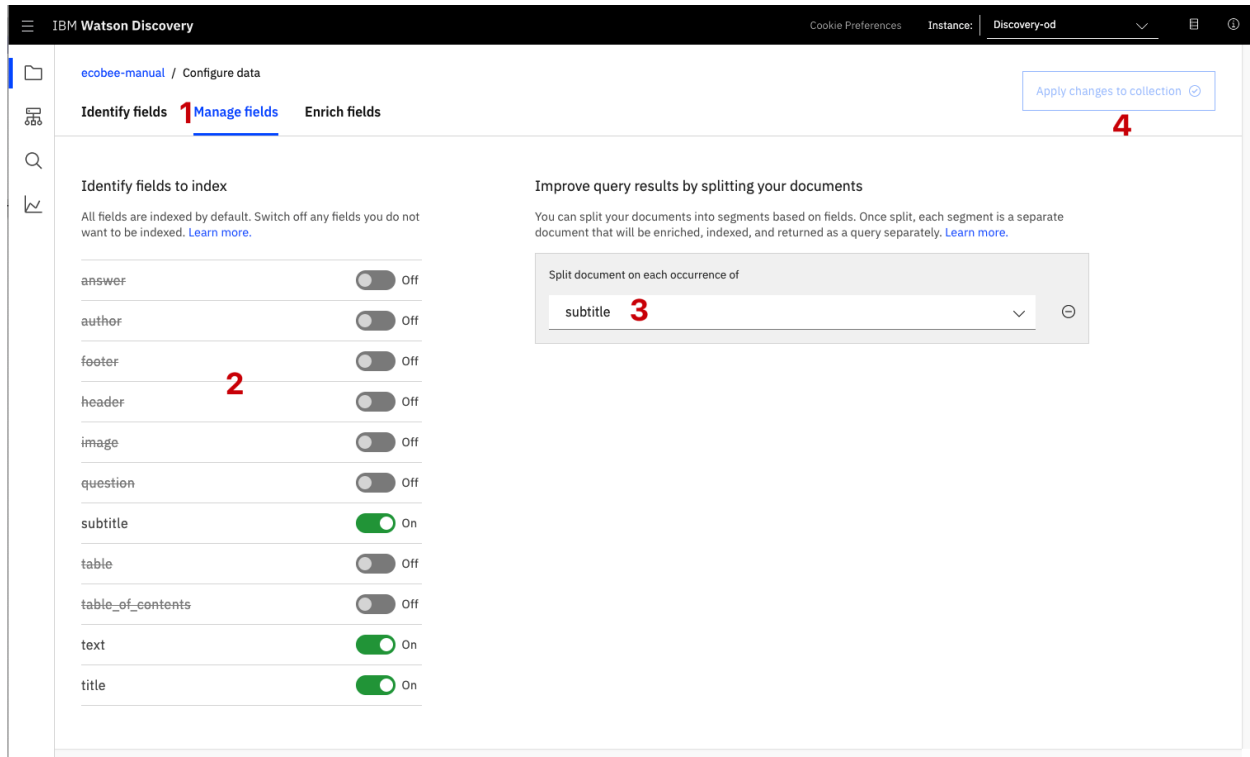
All page numbers as footers

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 shows the IBM Watson Discovery Overview page for the 'ecobee-manual' dataset. The page displays 130 documents, 0 failed documents, and search settings. It highlights identified fields (footer, subtitle, table_of_contents, text, title) and added enrichments (Entity Extraction, Sentiment Analysis, Concept Tagging, Category Classification). The Sentiment Analysis section shows 37% positive, 26% neutral, and 36% negative sentiment. The page also includes a 'Build your own query' link.

IBM Watson Discovery

Cookie Preferences Instance: Discovery-od

ecobee-manual

Configure data

Overview Errors and warnings (130) Search settings

130 documents

0 documents failed View details

Created on 3/28/2019 4:27:53 pm EDT
Last updated 3/28/2019 4:27:53 pm EDT

Upload documents

Identified 5 fields from your data

- footer
- subtitle
- table_of_contents
- text
- title

Need to identify more fields? Add fields

Added 4 enrichments to your data

Entity Extraction

0.3°C (4) | 0.5°F (4) | 10°F (4) | 900 seconds (4) | 20 min (3)

Sentiment Analysis

37% positive 26% neutral 36% negative

Concept Tagging

Heat (17) | Internet (14) | HVAC (13) | Netscape (13) | Temperature (13)

5 enrichments available. Add enrichments

Now you're ready to query!

Entities of type Quantity which have negative sentiment

Run

Documents that contain Heat, but not Internet

Run

Top entities with their average, min, max sentiment score

Run

Build your own query

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

The screenshot shows the IBM Watson Discovery Query panel for the 'ecobee-manual' dataset. The search query is 'how do I turn on the heater?'. The results show 10 of 38 matching documents, including 'ecobee3_UserGuide.pdf'. The panel includes a 'Run query' button and a 'Close' button.

IBM Watson Discovery

Cookie Preferences Instance: Discovery-od

ecobee-manual / Build queries

Build a query using one or more of these components. Learn more. Use a sample query

Search for documents

Use natural language Use the Discovery Query Language

how do I turn on the heater?

+ Include analysis of your results

+ Filter which documents you query

> More options

Run query Close

Summary JSON Train Watson to improve results

QUERY URL https://gateway.watsonplatform.net/discovery/api/v1/environ...

Passages

"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."

"The amount of indoor air required to maintain sufficient indoor air quality depends on how big your house is, how many people live there, and the capacity of your ventilation device. You should consult with a local contractor who can guide you on how often you should be running your ventilation device."

"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."

"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."

Results

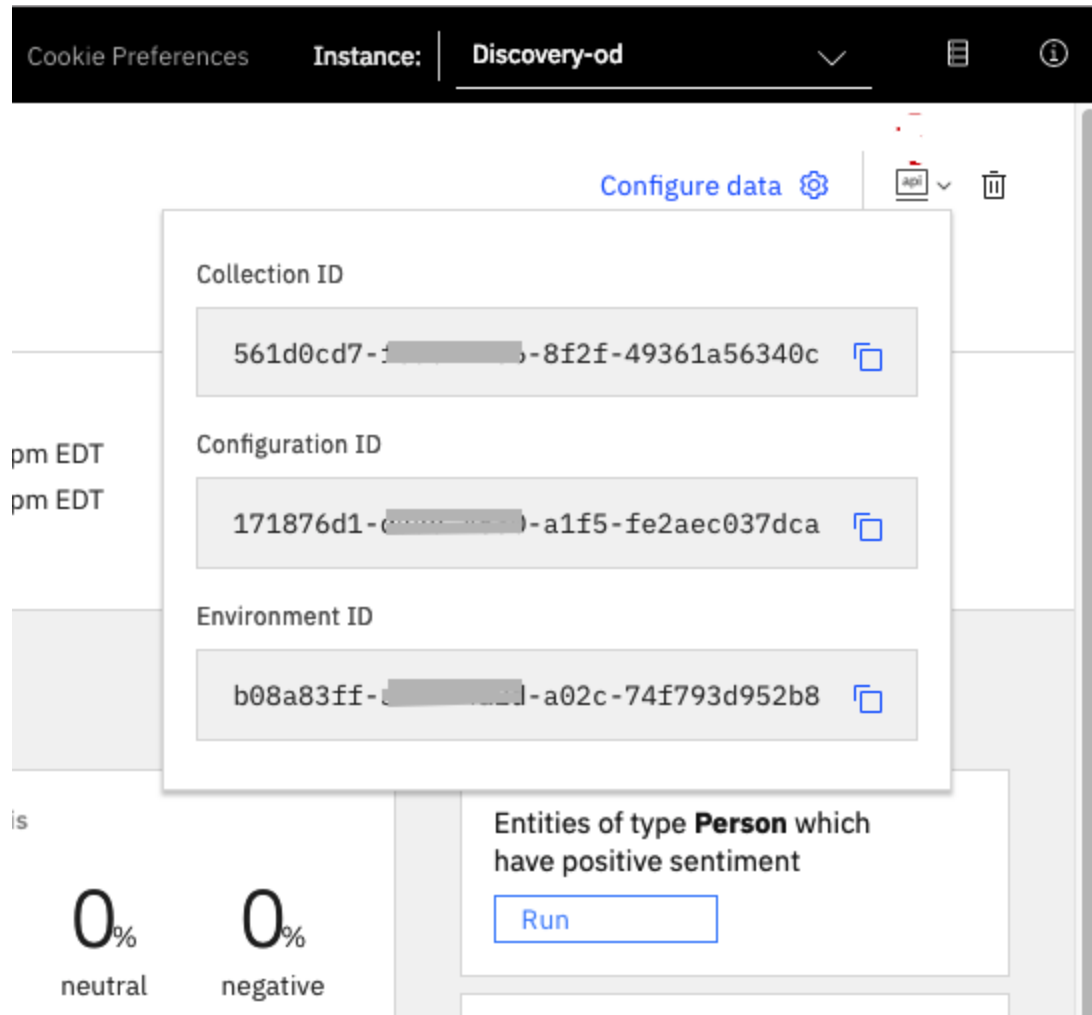
Showing 10 of 38 matching documents

ecobee3_UserGuide.pdf

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:

Resource list / **Discovery-od**

Resource group: default Location: Dallas [Add Tags](#)

Service credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service. [Learn more](#)

Service credentials [New credential](#)

Items per page **10** | 1-1 of 1 items 1 of 1 pages < 1 >

KEY NAME	DATE CREATED	ACTIONS
Service credentials-1	FEB 5, 2019 - 09:26:31 AM	View credentials 2

```

{
  "apikey": "dry8f3aITnsy; [REDACTED] AHiau8bkoAfu10",
  "iam_apikey_description": "Auto generated apikey during resource-key operation for Instance - crn:v1:bluemix:public:discovery:us-south:a/bc1bd51c396536dc7d5f81d5a4e19533:acf2871-3b0d-4e04-a0f9-8da59770852::",
  "iam_apikey_name": "auto-generated-apikey-f5f36cdd-d1d2-4a17-b41d-8ca5d1f1c7a6",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/bc1bd51c396536dc7d5f81d5a4e19533::serviceid:ServiceId-016b8efa-a050-4708-a191-0b71f43cbddb",
  "url": "https://gateway.watsonplatform.net/discovery/api"
}

```

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]:

Catalog

functions **1** [Filter](#)

All Categories (2)

- VPC Infrastructure
- Compute (1)
- Containers
- Networking
- Storage
- AI
- Analytics
- Databases
- Developer Tools
- Integration
- Internet of Things
- Security and Identity
- Starter Kits
- Web and Mobile
- Web and Application (1)

Compute

Serverless Compute

Functions **2**

IBM • IAM-enabled

IBM Cloud Functions is a Function-as-a-Service (FaaS) platform which executes functions in response to incoming events.

Web and Application

Difftek

Third Party

API-first platform for fintech applications

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

The screenshot shows the 'Create Action' panel in the IBM Cloud Functions console. The left sidebar contains navigation links: Functions, Getting Started, Actions, Triggers, APIs, Monitor, Logs, and Namespace Settings. The main area is titled 'Create Action' and contains the following fields:

- Action Name:** A text input field containing 'disco-action-2' (marked with a red 1).
- Enclosing Package:** A dropdown menu showing '(Default Package)' (marked with a red 2). A 'Create Package' button is to the right.
- Runtime:** A dropdown menu showing 'Node.js 10' (marked with a red 3).

Below the runtime dropdown, there is a link: 'Looking for Java, .NET or Docker? [Docker](#) Actions can be created with the [CLI](#)'. At the bottom of the panel are three buttons: 'Cancel', 'Previous', and 'Create' (marked with a red 4).

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

The screenshot shows the 'disco-action' code editor in the IBM Cloud Functions console. The left sidebar contains navigation links: Code (marked with a red 1), Parameters, Runtime, Endpoints, Connected Triggers, Enclosing Sequences, and Logs. The main area is titled 'disco-action' and shows the 'Code' tab selected. The code editor displays the following JavaScript code:

```
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,
```

The 'Invoke' button is located in the top right corner of the code editor (marked with a red 3).

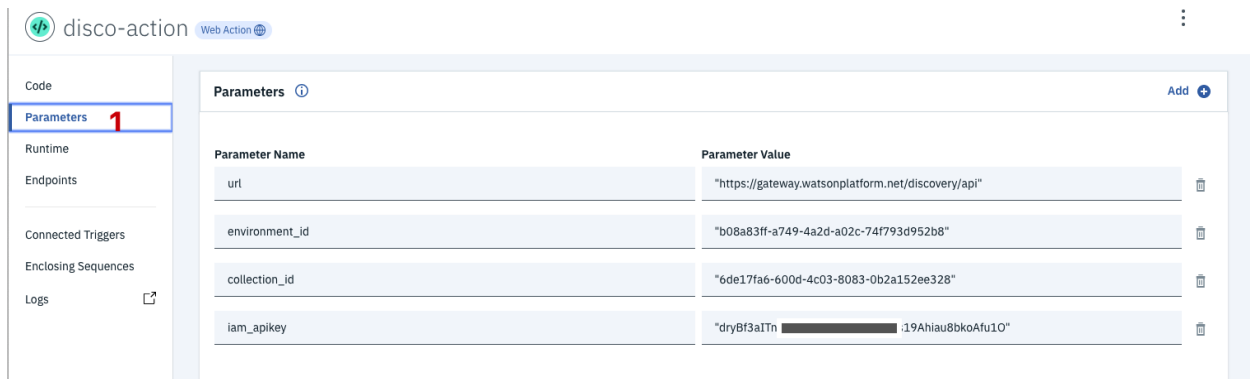
In the code editor window [2], cut and paste in the code from the disco-action.js file

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.

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:

The screenshot shows the IBM Cloud Functions console for the 'disco-action' in the 'IBM Cloud Storage_DSX-journey-2' namespace. The 'Code' tab is active, displaying a Node.js 10 script. The script uses the 'assert' library and the 'DiscoveryV1' class from the 'watson-developer-cloud/discovery/v1' package. It defines a 'main' function that takes 'params' and returns a Promise. The Promise resolves to a 'discovery' object, which is then used to search for 'iPhone' using the 'discover' method. The results are returned as a JSON object.

```
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-     discovery.discover(params.input, function (err, results) {
43-       if (err) {
44-         reject(err);
45-       } else {
46-         resolve(results);
47-       }
48-     });
49-   });
50- }
51-
52- exports.main = main;
```

The 'Activations' tab shows a successful activation of the 'disco-action' with a duration of 1050 ms, occurring on 6/6/2019 at 10:45:14. The activation ID is 'elbfc0ff21544c85bfc0ff21549c85a1'. The results are a JSON object containing 'matching_results' (14), 'passages' (an array of objects), and 'results' (an array of objects). The 'results' array contains two objects, each with 'enriched_text' and 'categories' (an array of objects). The 'categories' array contains two objects, each with 'label' and 'score'.

```
{
  "matching_results": 14,
  "passages": [
    {
      "text": "The iPhone is a line of smartphones designed and marketed by Apple Inc. It is a touchscreen-enabled device that functions as a web browser, email client, and other applications, and is able to download and run third-party mobile apps from the App Store.",
      "score": 0.842265
    },
    {
      "text": "The iPhone is a line of smartphones designed and marketed by Apple Inc. It is a touchscreen-enabled device that functions as a web browser, email client, and other applications, and is able to download and run third-party mobile apps from the App Store.",
      "score": 0.835879
    }
  ],
  "results": [
    {
      "label": "/technology and computing/hardware/computer peripherals/computer monitors",
      "score": 0.832254
    },
    {
      "label": "/technology and computing/hardware/computer peripherals/computer monitors",
      "score": 0.832254
    }
  ],
  "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]:

The screenshot shows the 'Endpoints' panel for the 'disco-action' in the 'IBM Cloud Storage_DSX-journey-2' namespace. The 'Web Action' section is expanded, showing the 'Enable as Web Action' checkbox checked. The 'Raw HTTP handling' checkbox is unchecked. The 'HTTP METHOD' is set to 'ANY', the 'AUTH' is 'Public', and the 'URL' is 'https://us-south.functions.cloud.ibm.com/api/v1/web/IBM%20Cloud%20Storage_DSX-journey-2/default/disco-action'. The 'REST API' section is also expanded, showing the 'HTTP METHOD' set to 'POST', the 'AUTH' is 'API-KEY', and the 'URL' is 'https://us-south.functions.cloud.ibm.com/api/v1/namespaces/IBM%20Cloud%20Storage_DSX-journey-2/actions/disco-action'. The 'CURL' section shows a curl command for a POST request to the REST API endpoint.

```
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.

4. Configure Watson Assistant

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.

The screenshot shows the Watson Assistant configuration page for the intent #Product_Information. The page has a header with a back arrow, the intent name, and a 'Try it' button. Below the header, there are sections for 'Intent name', 'Description (optional)', and 'Add user example'. The 'Intent name' section shows '#Product_Information' with a copy icon. The 'Description (optional)' section shows 'User wants help using the thermostat'. The 'Add user example' section has a text input field with the placeholder 'Type a user example here' and two buttons: 'Add example' and 'Show recommendations'. Below these sections is a table of user examples.

<input type="checkbox"/> User examples (3) ▼	Added	0 conflicts	Show only conflicts ⓘ
<input type="checkbox"/> How do I access the settings ✎	2 hours ago		
<input type="checkbox"/> How do I set the time ✎	2 hours ago		
<input type="checkbox"/> How do I turn on the heater ✎	2 hours ago		

Create new dialog node

Now we need to add a node to handle our intent. Click on the Dialog [1] tab, then click on the drop down menu for the Small Talk node [2], and select the Add node below [3] option.

Skills /

Customer Care Sample Skill copy

Sample simple customer service skill to get you started.

Intents

Entities

1 Dialog

Analytics

Options

Versions

Content Catalog

The screenshot displays the LLM Studio interface with a workflow consisting of four nodes:

- 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.

A context menu is open over the 'Small Talk' node, showing the following options:

- Add node to folder (labeled with a red 2)
- Add node above
- Add node below (labeled with a red 3)
- Add folder
- Move
- Duplicate
- Jump to
- Delete

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

IBM Watson Assistant

Skills /

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

Save new version

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

1 Ask about product Customize x

If assistant recognizes:

2 #Product_Information

Then respond with

Text

Enter response text

Response variations are set to **sequential**. Set to [random](#)
[Learn more](#)

Add response type

And finally:

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]:

IBM Watson Assistant

Cookie Preferences

Skills /

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

Try it

Save new version

IntentsEntitiesDialogAnalyticsOptionsVersionsContent Catalog

Webhooks

Autocorrection

System Entities

Webhooks

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](#)

URL

2 `https://us-south.functions.cloud.ibm.com/api/v1/web/IBM%20Cloud%20Stor`

Headers

Add HTTP headers for authorization or any other parameters required for invoking the specified request URL.

HEADER NAME	HEADER VALUE
-------------	--------------

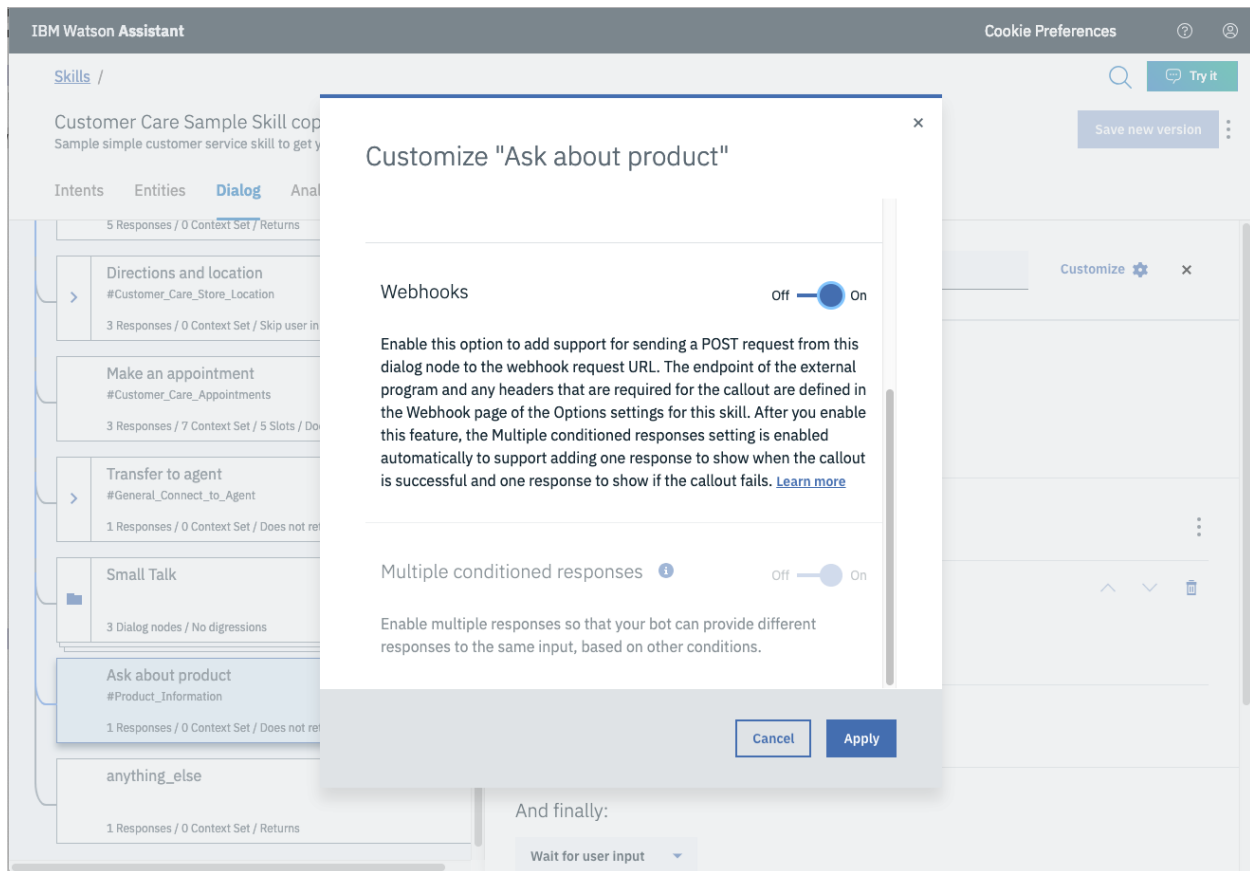
[Add header](#) [Add authorization](#)

Next step

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].

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

Skills /

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

Then respond with

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	\$webhook_result_1	\$webhook_result_1		
2	anything_else	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



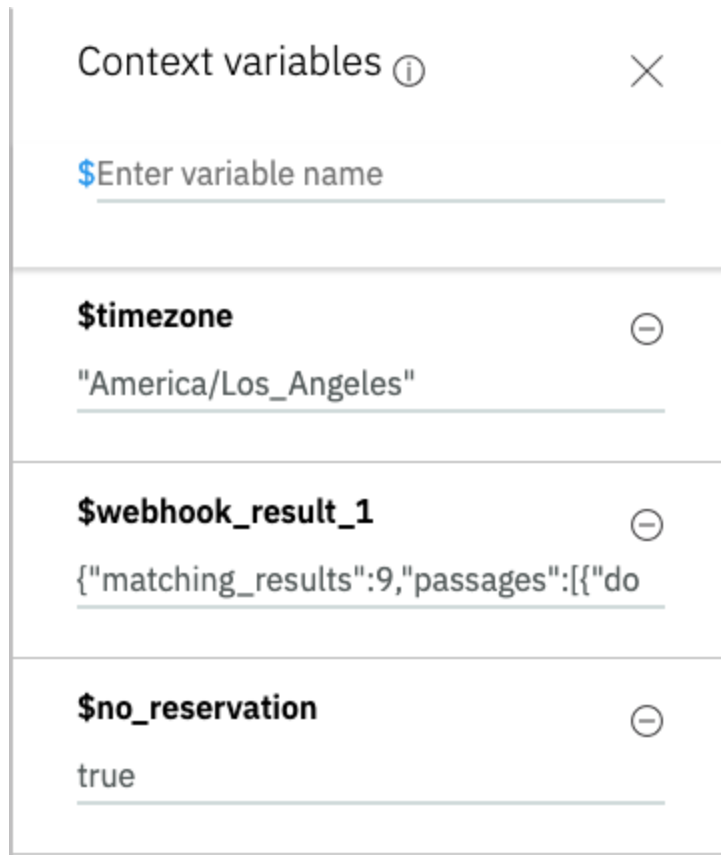
[{"document_id":"3a5efee70d8c-c9d70e2b94d22c15e2d1_2","end_offset":2791,"field":"text","passage_score":6.752501692678998,"passage_text":"Specify what the heat pump runs when the O/B Reversing Valve is engaged: On Cool runs cooling when O/B engages (most cases), or On Heat runs heating when O/B engages. 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"	
\$webhook_result_1	⊖
{ "matching_results": 9, "passages": [{ "do	
\$no_reservation	⊖
true	

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

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

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

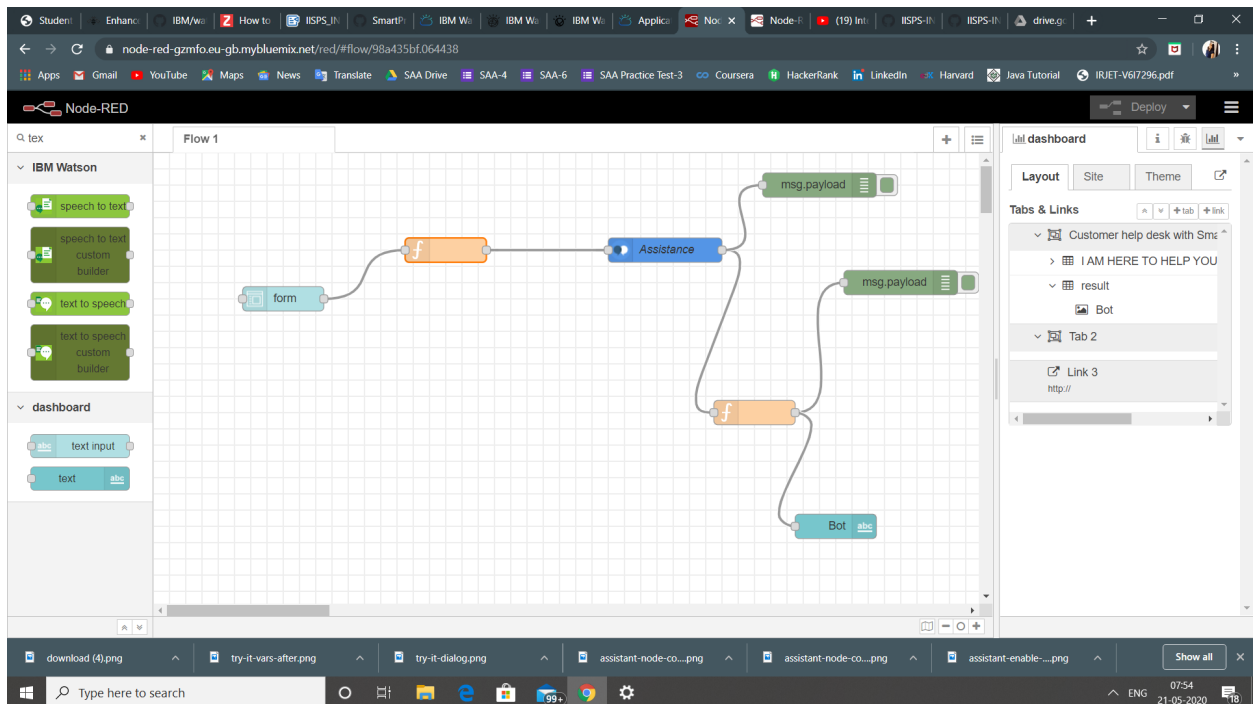
Connect the output of "input parsing" function node to "You" text node and output of

“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



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 chatbot 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": "7253a121.16642",
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": ""
  },
  {
    "id": "b1b11140.4e4ef",
    "type": "inject",

```

```
"z": "7253a121.16642",
"name": "",
"topic": "",
"payload": "Hello Node-RED!",
"payloadType": "str",
"repeat": "",
"crontab": "",
"once": false,
"onceDelay": "",
"x": 141,
"y": 61,
"wires": [
  [
    "2371449b.4bf2cc"
  ]
],
},
{
  "id": "f2f2649a.0d0d98",
  "type": "debug",
  "z": "7253a121.16642",
  "name": "",
  "active": true,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "payload",
  "targetType": "msg",
  "x": 670,
  "y": 140,
  "wires": []
},
{
  "id": "e150cc25.d7aa",
  "type": "function",
  "z": "7253a121.16642",
  "name": "input parsing",
```

```
"func": "msg.payload=msg.payload.text;\nreturn msg;",
"outputs": 1,
"noerr": 0,
"x": 270,
"y": 240,
"wires": [
  [
    "2371449b.4bf2cc",
    "49234419.64d55c"
  ]
],
},
{
  "id": "d3215cca.6b835",
  "type": "ui_form",
  "z": "7253a121.16642",
  "name": "",
  "label": "",
  "group": "a790cfa9.6f041",
  "order": 1,
  "width": "0",
  "height": "0",
  "options": [
    {
      "label": "Enter your query",
      "value": "text",
      "type": "text",
      "required": true,
      "rows": null
    }
  ],
  "formValue": {
    "text": ""
  },
  "payload": "",
  "submit": "submit",
  "cancel": "cancel",
```

```
"topic": "",
"x": 107.5,
"y": 302,
"wires": [
  [
    "e150cc25.d7aa"
  ]
]
},
{
  "id": "49234419.64d55c",
  "type": "ui_text",
  "z": "7253a121.16642",
  "group": "a790cfa9.6f041",
  "order": 2,
  "width": "0",
  "height": "0",
  "name": "",
  "label": "You",
  "format": "{{msg.payload}}",
  "layout": "col-center",
  "x": 453.5,
  "y": 306,
  "wires": []
},
{
  "id": "d2a61bef.00f978",
  "type": "ui_text",
  "z": "7253a121.16642",
  "group": "d2f10b81.4883c8",
  "order": 1,
  "width": "6",
  "height": "4",
  "name": "",
  "label": "Bot",
  "format": "{{msg.payload}}",
  "layout": "col-center",
```



```

    "x": 688.5,
    "y": 237,
    "wires": []
  },
  {
    "id": "29aac8ee.b98c68",
    "type": "function",
    "z": "7253a121.16642",
    "name": "parsing",
    "func": "msg.payload = msg.payload.output.generic[0].text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 470,
    "y": 160,
    "wires": [
      [
        "f2f2649a.0d0d98",
        "d2a61bef.00f978"
      ]
    ]
  },
  {
    "id": "2371449b.4bf2cc",
    "type": "watson-assistant-v2",
    "z": "7253a121.16642",
    "name": "My first assistant",
    "service-endpoint":
    "https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/ae44476e-77ab-4
    578-9d11-23464ea66634",
    "assistant_id": "d3fa58b9-7946-4784-b120-5e025c58c9ba",
    "debug": false,
    "restart": false,
    "return_context": true,
    "alternate_intents": false,
    "multisession": true,
    "timeout": "",
    "optout-learning": false,

```

```
"x": 391.5,
"y": 100,
"wires": [
  [
    "29aac8ee.b98c68",
    "caf5ac15.c978d"
  ]
]
},
{
  "id": "caf5ac15.c978d",
  "type": "debug",
  "z": "7253a121.16642",
  "name": "",
  "active": true,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "payload",
  "targetType": "msg",
  "x": 557.5,
  "y": 61,
  "wires": []
},
{
  "id": "a790cfa9.6f041",
  "type": "ui_group",
  "z": "",
  "name": "",
  "tab": "27e055be.28bb7a",
  "order": 1,
  "disp": true,
  "width": "6",
  "collapse": false
},
{
  "id": "d2f10b81.4883c8",
```

```
    "type": "ui_group",
    "z": "",
    "name": "",
    "tab": "27e055be.28bb7a",
    "order": 3,
    "disp": true,
    "width": "6",
    "collapse": false
  },
  {
    "id": "27e055be.28bb7a",
    "type": "ui_tab",
    "z": "",
    "name": "Product",
    "icon": "dashboard",
    "order": 2,
    "disabled": false,
    "hidden": false
  }
]

[
  {
    "id": "7253a121.16642",
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": ""
  },
  {
    "id": "b1b11140.4e4ef",
    "type": "inject",
    "z": "7253a121.16642",
    "name": "",
    "topic": "",
    "payload": "Hello Node-RED!",
    "payloadType": "str",
```

```
"repeat": "",
"crontab": "",
"once": false,
"onceDelay": "",
"x": 141,
"y": 61,
"wires": [
  [
    "2371449b.4bf2cc"
  ]
],
},
{
  "id": "f2f2649a.0d0d98",
  "type": "debug",
  "z": "7253a121.16642",
  "name": "",
  "active": true,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "payload",
  "targetType": "msg",
  "x": 670,
  "y": 140,
  "wires": []
},
{
  "id": "e150cc25.d7aa",
  "type": "function",
  "z": "7253a121.16642",
  "name": "input parsing",
  "func": "msg.payload=msg.payload.text;\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 270,
  "y": 240,
```

```
"wires": [
  [
    "2371449b.4bf2cc",
    "49234419.64d55c"
  ]
],
{
  "id": "d3215cca.6b835",
  "type": "ui_form",
  "z": "7253a121.16642",
  "name": "",
  "label": "",
  "group": "a790cfa9.6f041",
  "order": 1,
  "width": "0",
  "height": "0",
  "options": [
    {
      "label": "Enter your query",
      "value": "text",
      "type": "text",
      "required": true,
      "rows": null
    }
  ],
  "formValue": {
    "text": ""
  },
  "payload": "",
  "submit": "submit",
  "cancel": "cancel",
  "topic": "",
  "x": 107.5,
  "y": 302,
  "wires": [
    [
```

```
        "e150cc25.d7aa"
    ]
]
},
{
    "id": "49234419.64d55c",
    "type": "ui_text",
    "z": "7253a121.16642",
    "group": "a790cfa9.6f041",
    "order": 2,
    "width": "0",
    "height": "0",
    "name": "",
    "label": "You",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "x": 453.5,
    "y": 306,
    "wires": []
},
{
    "id": "d2a61bef.00f978",
    "type": "ui_text",
    "z": "7253a121.16642",
    "group": "d2f10b81.4883c8",
    "order": 1,
    "width": "6",
    "height": "4",
    "name": "",
    "label": "Bot",
    "format": "{{msg.payload}}",
    "layout": "col-center",
    "x": 688.5,
    "y": 237,
    "wires": []
},
{
```

```

    "id": "29aac8ee.b98c68",
    "type": "function",
    "z": "7253a121.16642",
    "name": "parsing",
    "func": "msg.payload = msg.payload.output.generic[0].text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 470,
    "y": 160,
    "wires": [
      [
        "f2f2649a.0d0d98",
        "d2a61bef.00f978"
      ]
    ]
  },
  {
    "id": "2371449b.4bf2cc",
    "type": "watson-assistant-v2",
    "z": "7253a121.16642",
    "name": "My first assistant",
    "service-endpoint":
    "https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/ae44476e-77ab-4
    578-9d11-23464ea66634",
    "assistant_id": "d3fa58b9-7946-4784-b120-5e025c58c9ba",
    "debug": false,
    "restart": false,
    "return_context": true,
    "alternate_intents": false,
    "multisession": true,
    "timeout": "",
    "optout-learning": false,
    "x": 391.5,
    "y": 100,
    "wires": [
      [
        "29aac8ee.b98c68",

```

```
        "caf5ac15.c978d"
    ]
}
{
    "id": "caf5ac15.c978d",
    "type": "debug",
    "z": "7253a121.16642",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 557.5,
    "y": 61,
    "wires": []
},
{
    "id": "a790cfa9.6f041",
    "type": "ui_group",
    "z": "",
    "name": "",
    "tab": "27e055be.28bb7a",
    "order": 1,
    "disp": true,
    "width": "6",
    "collapse": false
},
{
    "id": "d2f10b81.4883c8",
    "type": "ui_group",
    "z": "",
    "name": "",
    "tab": "27e055be.28bb7a",
    "order": 3,
```



```
    "disp": true,  
    "width": "6",  
    "collapse": false  
  },  
  {  
    "id": "27e055be.28bb7a",  
    "type": "ui_tab",  
    "z": "",  
    "name": "Product",  
    "icon": "dashboard",  
    "order": 2,  
    "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-nodered/>
4. <http://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red>
5. <https://github.com/IBM/watson-discovery-sdu-with-assistant>
6. <https://www.youtube.com/watch?v=Jpr3wVH3FVA>