

Project Report

NAME : JERRIPOTHU DAMODAR
(damuqwe123@gmail.com)

TITLE : Intelligent Customer Help Desk With
Smart Document Understanding

CATEGORY : Machine Learning

Internship at:smartinternz.com@2020

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, Node - RED
- Functional Requirements: IBM Cloud
- Technical Requirements: AI, ML, WATSON AI, PYTHON
- Software Requirements: Watson Assistant, Watson discovery.
- Project Deliverables: Smart internz Internship
- Project Team: J.Damodar
- Project Duration: 19days

1.2. Purpose:

The typical Customer Care ChatBot can answer simple questions, such as store locations and hours, directions, and may be 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 owner's manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections

of the owner's manual to help solve our customer's 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 owner's 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:

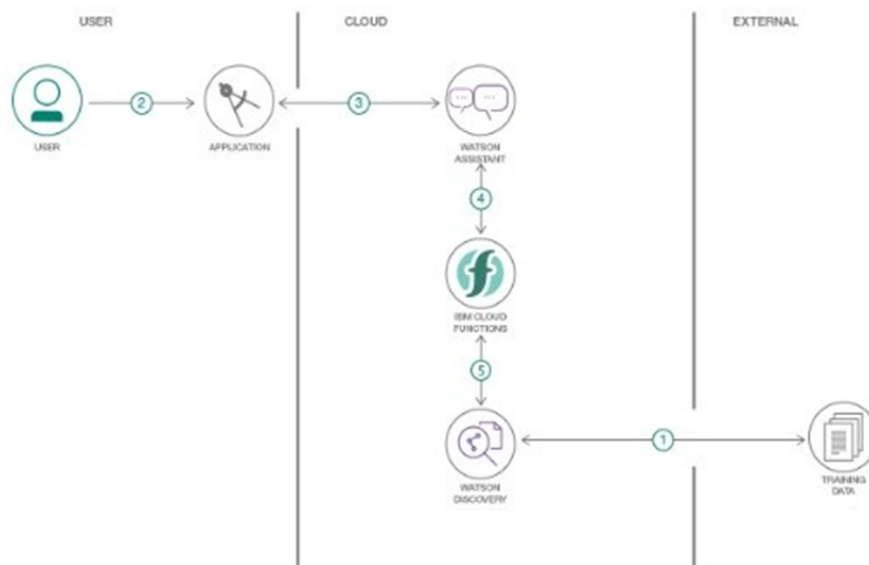
Generally Chatbots means getting input from users and getting only response questions and for some questions the output from bot will be like "try again", "I don't understand", "will you repeat again", and soon...and directs customer to customer agent but a good Customer ChatBot should minimize involvement 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 a virtual agent in chatbot so it can understand the queries that are posted by customers. The virtual agent should be trained from some insight records based on 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 back end 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 back end 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 pre-defined 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
- IBM Cloud Function

Creation of Node-RED in IBM cloud:

- Step-1: Log in to IBM and go to the catalog
- Step-2: Search for node-red and select “Node-RED Starter” Service
- Step-3: Enter the Unique name and click on create a button
- Note: Your Node-red service is starting
- Step-5: We have to configure Node red for the first time. Click on next to continue

Welcome to your new Node-RED instance on IBM Cloud

We know you're eager to start wiring up your flows, but first there are a couple of tasks you should do:

- Secure your Node-RED editor
- Browse available IBM Cloud nodes

Previous

Next

- Step-6: Secure your node red editor by giving a username and password and click on Next

Secure your Node-RED editor

☒ Secure your editor so only authorised users can access it

Username

Password

☐ Allow anyone to view the editor, but not make any changes

Must be at least 8 characters

☐ *Not recommended:* Allow anyone to access the editor and make changes

Previous

Next

- Step-7: Click Next to continue

Browse available IBM Cloud nodes

There are lots of nodes available from the community that can be used to add more capabilities to your application. The list below is just a small selection.

You can find many more nodes on the [Flow Library](#).

You can use the Palette Manager built into editor to search for and install nodes. Alternatively, you can also edit your application's package.json file and adding them to the dependencies section.

node-red-dashboard
Quickly create dashboards driven by Node-RED

node-red-contrib-ibm-wiotp-device-ops
Perform device and gateway operations using the Watson IoT Platform

node-red-contrib-iot-virtual-device
Simulate device behavior and use it to run many device instances

node-red-contrib-objectstore
Store, delete and restore objects in the ObjectStore service

node-red-contrib-bluemix-hdfs
Create and update files using HDFS for Analytics

node-red-contrib-ibmpush
Send push notifications to mobile devices using the

Previous

Next

- Step-8: Click Finish

Finish the install

You have made the following selections:

- *Not recommended:* Allow anyone to access the editor and make changes

You can change these settings at any time by setting the following environment variables via the IBM Cloud console:

- NODE_RED_USERNAME - the username
- NODE_RED_PASSWORD - the password
- NODE_RED_GUEST_ACCESS - if set to 'true', allows anyone read-only access to the editor



Previous

Finish

- Step-9: Click on Go to Node-Red flow editor to launch the flow editor

Node-RED on IBM Cloud

Node-RED

Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

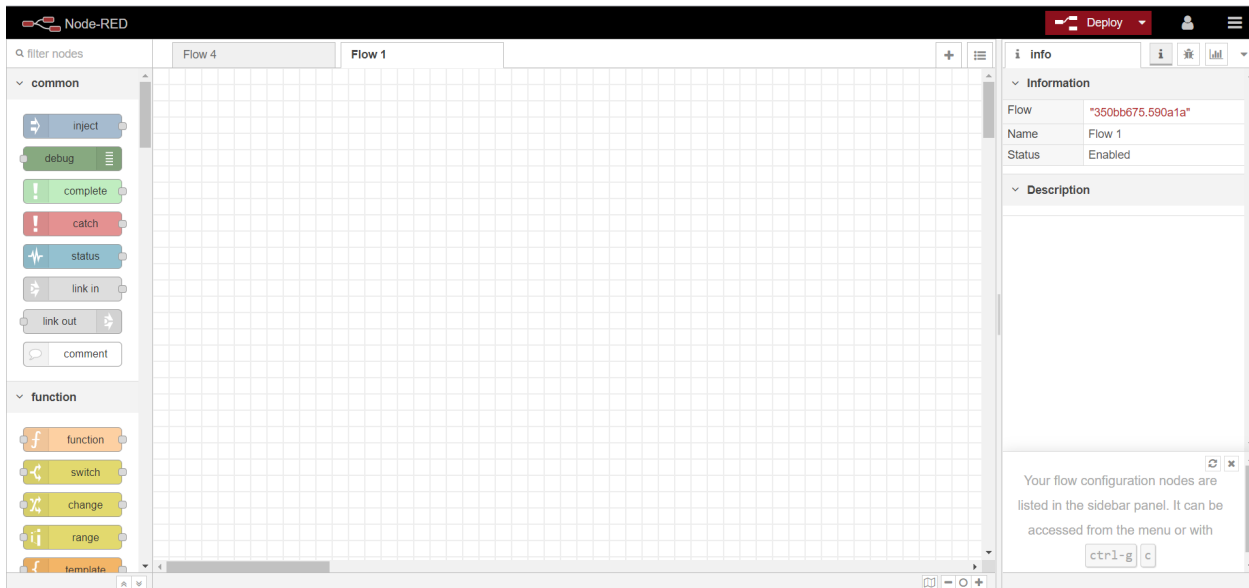
This instance is running as an IBM Cloud application, giving it access to the wide range of services available on the platform.

More information about Node-RED, including documentation, can be found at nodered.org.

Go to your Node-RED flow editor

[Learn how to customise Node-RED](#)

- Node red editor has various nodes with the respective functionality



Creation of Watson discovery instance in IBM Cloud:

Import the document as shown below, launch the Watson Discovery to land 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 *hpUserGuide.pdf*. The hp is a popular laptop company that has a wi-fi interface and multiple configuration options.

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

IBM Watson Discovery

Watson storage collection

Overview | Errors and warnings (55) | Search settings

55 documents

0 documents failed

Created on: 5/21/2020 1:10:36 am EDT
Last updated: 5/21/2020 1:10:36 am EDT

[Upload documents](#)

Identified 5 fields from your data

- footer
- subtitle
- table_of_contents
- text
- title

[Need to identify more fields? Add fields](#)

Added 6 enrichments to your data

Entity Extraction

Google (10) | HP (4) | 1 second (3) | ADB (1) | Beat Audio (1)

Sentiment Analysis

20% positive | 67% neutral | 12% negative

Concept Tagging

Google (6) | Camera (5) | Accounts receivable (3) | Following (3) | Internet (3)

Category Classification

technology and com... operating systems

Now you're ready to query!

Most common entity types and their top entities

[Run](#)

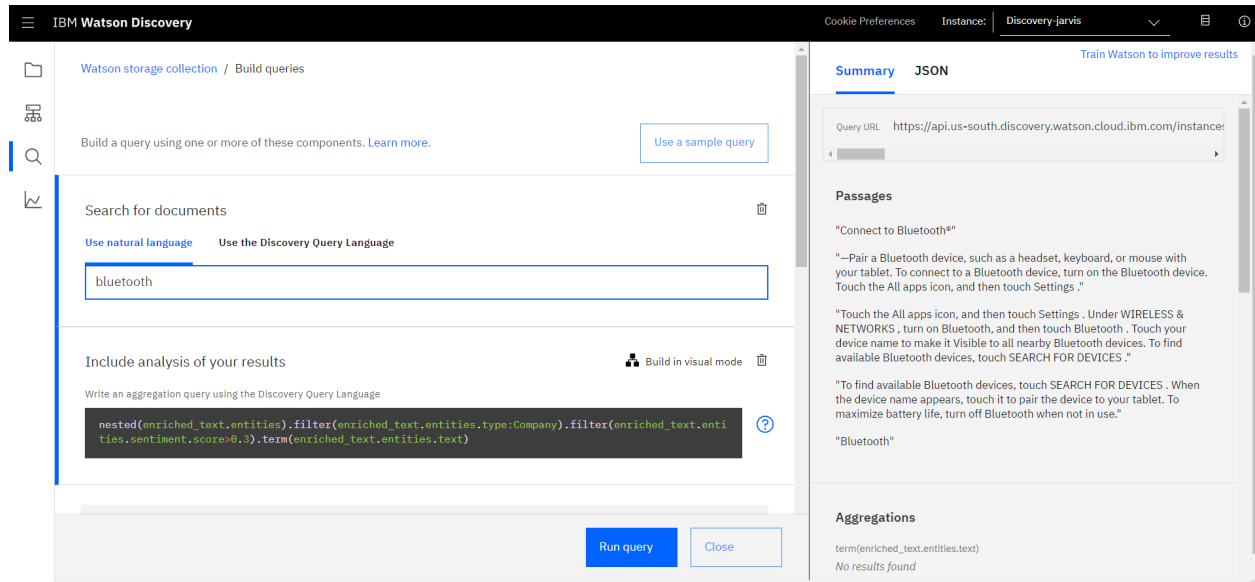
Entities of type **Company** which have negative sentiment

[Run](#)

Top people related to **/technology and computing/operating systems**

[Run](#)

- Click the Build your own query [1] button.

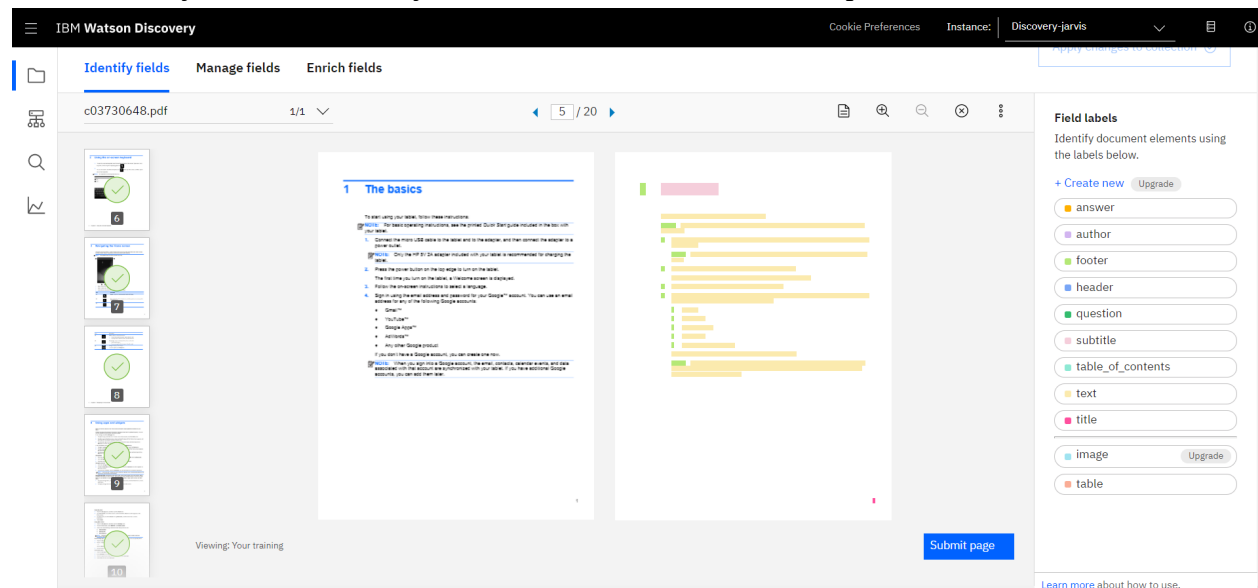


Enter queries related to the operation of the laptop and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question. Annotate with SDU

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 field 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.

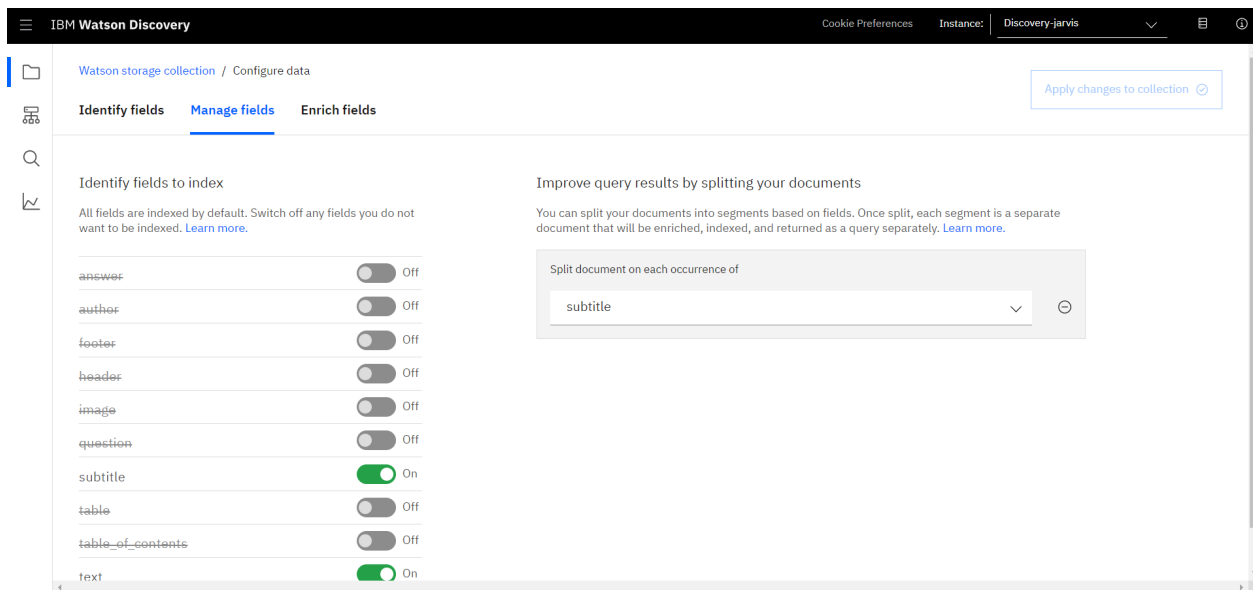
As you go through the annotations one page at a time, Discovery is learning and should start automatically updating the upcoming pages. Once you get to a page that is already correctly annotated, you can stop, or simply click Submit [5] to acknowledge it is correct. The more pages you annotate, the better 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.

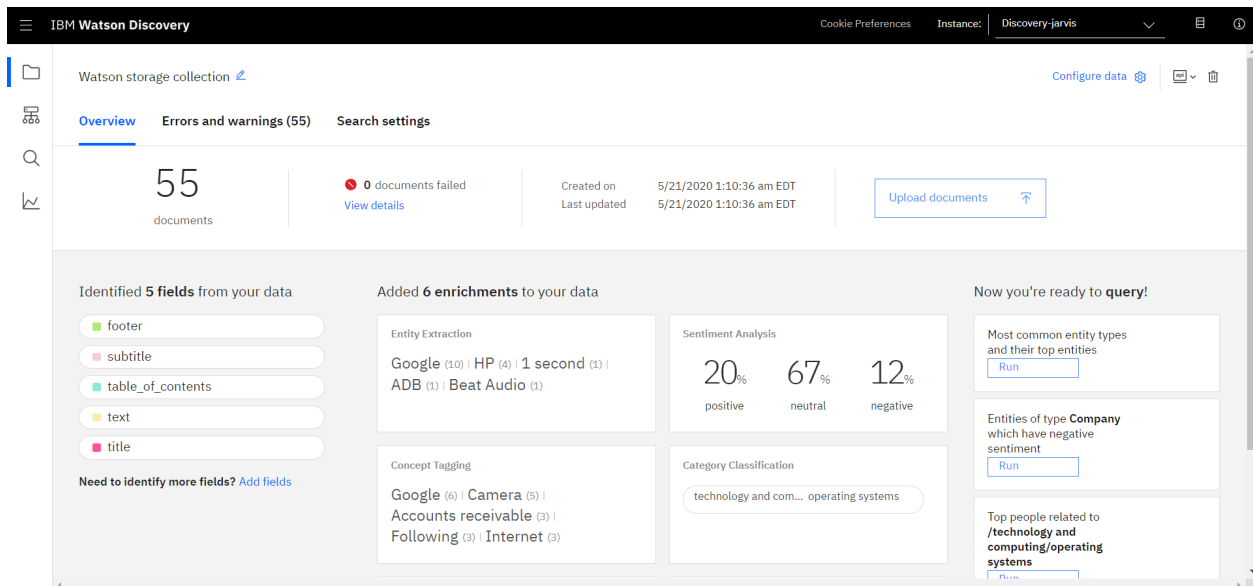
Next, click on the Manage fields [1] tab.



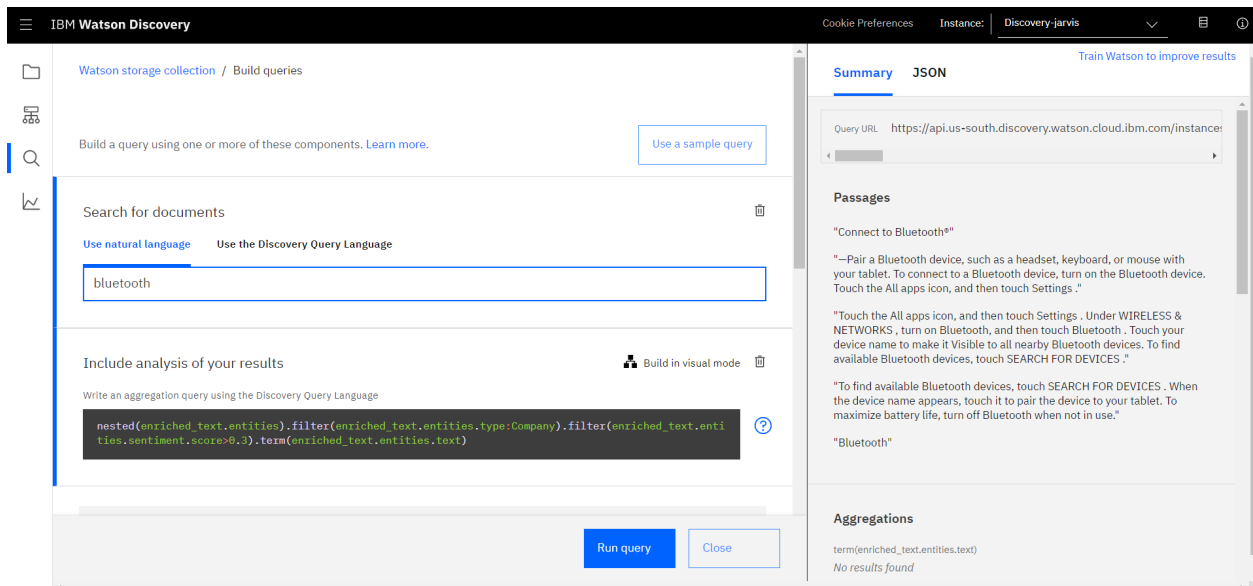
- [2] Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.
- [3] is telling Discovery to split the document apart, based on subtitle.
- Click [4] to submit your changes.

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:

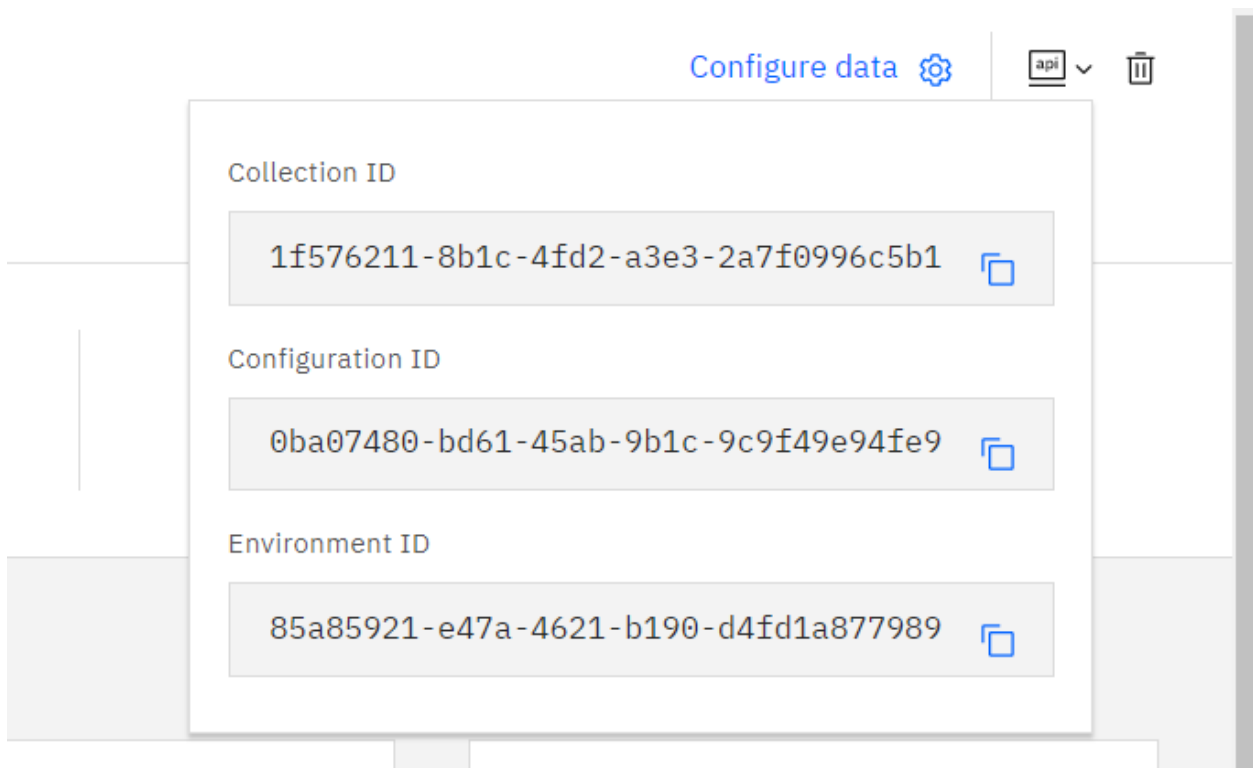


Return to the query panel (click Build your own query) and see how much better the results are.

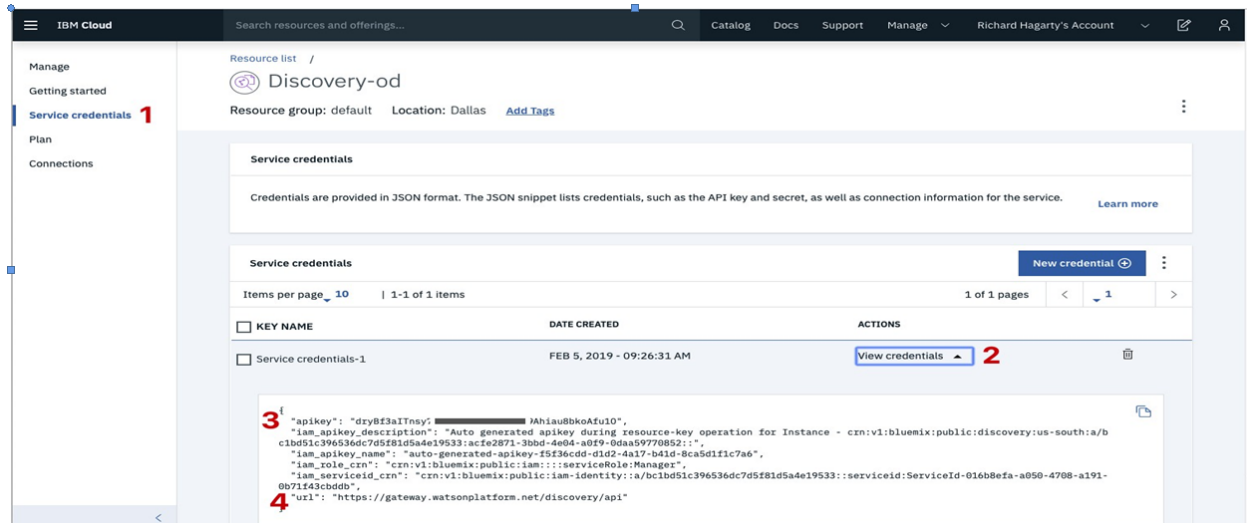


In upcoming steps, you will need to provide the credentials to access your Discovery collection. The values can be found in the following locations.

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

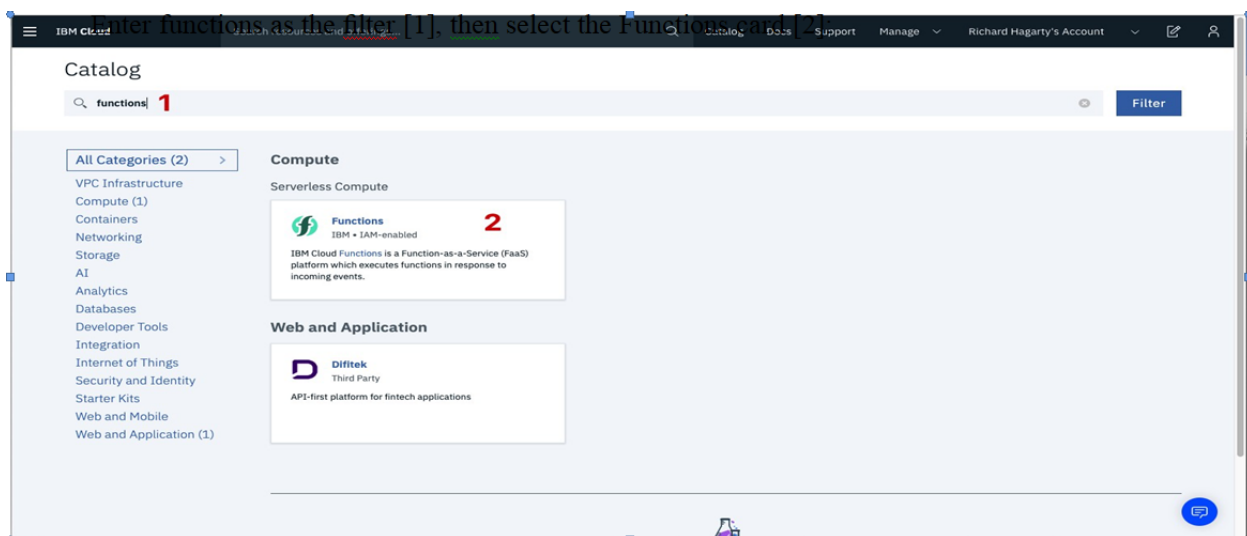


For credentials, return to the main panel of your Discovery Service, and click the Service credentials [1] tab:



Creating IBM Cloud function:

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.

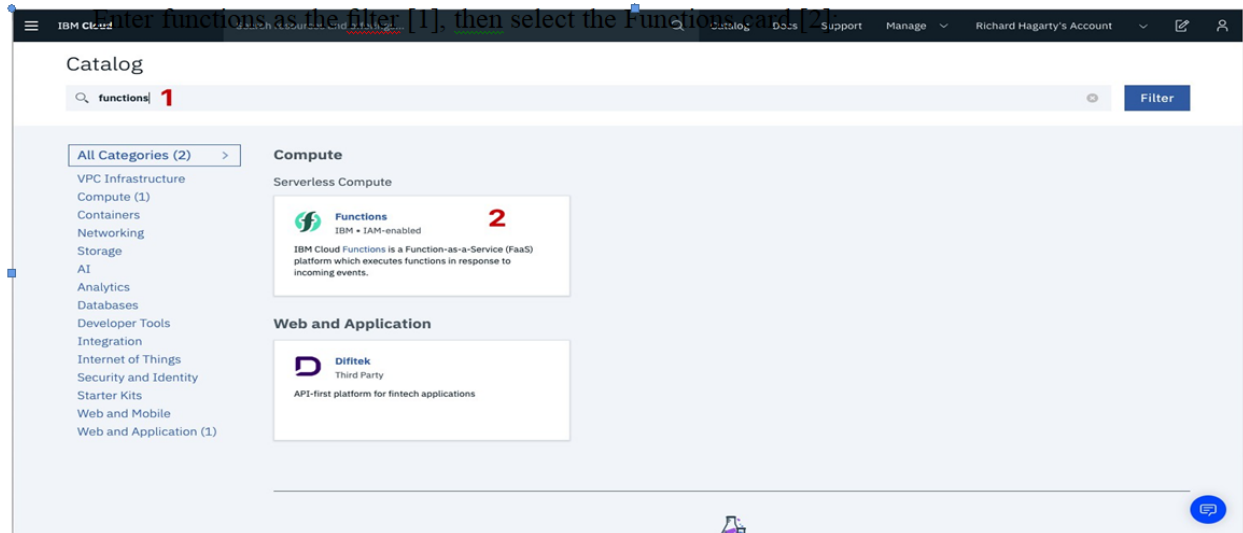


Enter functions as the filter [1], then select the Functions card [2]:

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.js10 [3] runtime. Click the Create button [4] to create the action.



Once your actions is created, click on the Code tab[1]

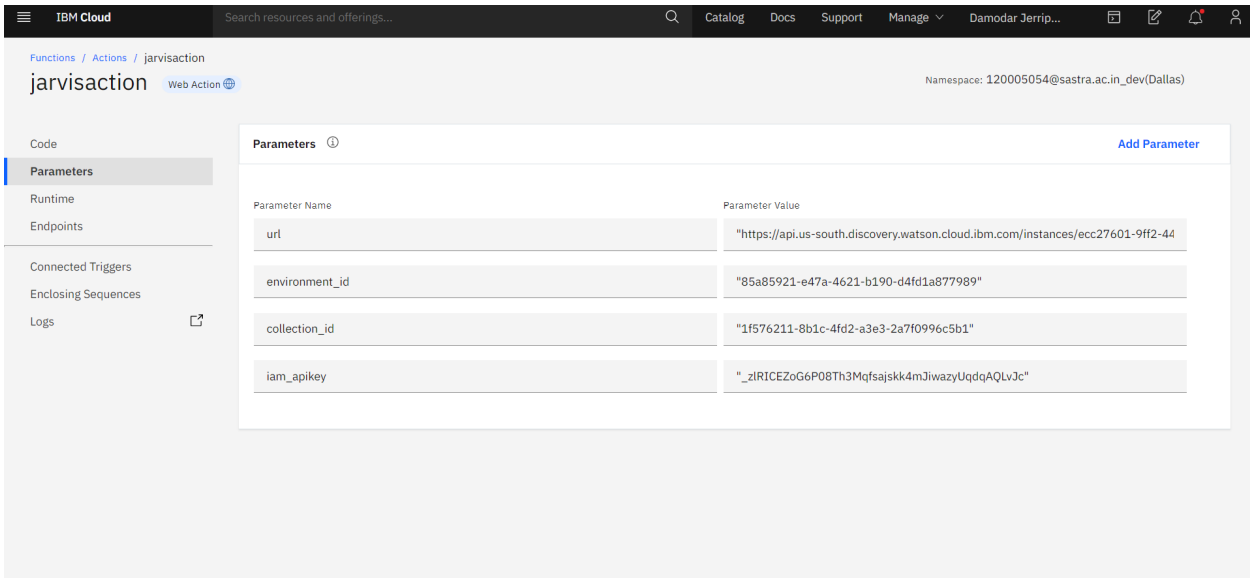


In the code editor window [2], cut and paste in the code from the disc.js file found in the actions directory of your local report. The code is pretty straight forward it simply connects to the Discovery service, makes a query against the collection, then returns the response.

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]:



The screenshot shows the IBM Cloud Functions console. The breadcrumb navigation is 'Functions / Actions / jarvisaction'. The function 'jarvisaction' is selected, and the 'Parameters' tab is active. The namespace is '120005054@sastra.ac.in_dev(Dallas)'. The left sidebar shows options: Code, Parameters (selected), Runtime, Endpoints, Connected Triggers, Enclosing Sequences, and Logs. The main area displays a table of parameters:

Parameter Name	Parameter Value
url	"https://api.us-south.discovery.watson.cloud.ibm.com/instances/ecc27601-9ff2-44"
environment_id	"85a85921-e47a-4621-b190-d4fd1a877989"
collection_id	"1f576211-8b1c-4fd2-a3e3-2a7f0996c5b1"
iam_apikey	"_zIRICEZoG6P08Th3Mqfsajskk4mJiwazyUqdqAQLvJc"

An 'Add Parameter' link is visible in the top right of the parameters section.

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:

The screenshot shows the IBM Cloud Functions console for a function named 'jarvisaction'. The 'Code' tab is selected, showing a Node.js script that uses the Watson Discovery API. The script defines a 'main' function that takes parameters and returns a Promise. It uses the 'watson-developer-cloud/discovery/v1' package. The 'Invoke' button is visible. To the right, the 'Activations' panel shows a successful invocation with a response from the Watson Assistant API.

```

1
2
3 const assert = require('assert');
4 const DiscoveryV1 = require('watson-developer-cloud/discovery/v1');
5
6 function main(params) {
7   return new Promise(function (resolve, reject) {
8
9     let discovery;
10
11     if (params.iam_apikey) {
12       discovery = new DiscoveryV1({
13         'iam_apikey': params.iam_apikey,
14         'url': params.url,
15         'version': '2019-03-25'
16       });
17     } else {
18       discovery = new DiscoveryV1({
19         'username': params.username,
20         'password': params.password,
21         'url': params.url,
22         'version': '2019-03-25'
23       });
24     }
25
26     discovery.query({
27       'environment_id': params.environment_id,
28       'collection_id': params.collection_id,
29       'natural_language_query': params.input,
30       'passages': true,
31       'count': 5,
32       'response_format': 'JSON'
33     }, function (err, results) {
34       if (err) {
35         reject(err);
36       } else {
37         resolve(results);
38       }
39     });
40   });
41 }

```

The 'Activations' panel shows a successful invocation with a response from the Watson Assistant API:

```

{
  "matching_results": 55,
  "passages": [],
  "results": [
    {
      "enriched_text": {
        "categories": [
          {
            "label": "/technology and computing/tech news",
            "score": 0.986853
          },
          {
            "label": "/technology and computing/internet technology/social network",
            "score": 0.745854
          },
          {
            "label": "/technology and computing/operating systems",
            "score": 0.711869
          }
        ],
        "concepts": [
          {
            "dbpedia_resource": "http://dbpedia.org/resource/Camera",
            "relevance": 0.92796,
            "text": "Camera"
          }
        ]
      }
    }
  ]
}

```

Next, go to the End points panel [1]:

The screenshot shows the IBM Cloud Functions console for a function named 'jarvisaction'. The 'Endpoints' tab is selected, showing the 'Web Action' configuration. The 'Enable as Web Action' checkbox is checked, and the 'Raw HTTP handling' checkbox is unchecked. The 'HTTP Method' is set to 'ANY', and the 'Auth' is set to 'Public'. The 'URL' is https://us-south.functions.cloud.ibm.com/api/v1/web/120005054%40sastra.ac.in_dev/default/jarvisaction.

HTTP Method	Auth	URL
ANY	Public	https://us-south.functions.cloud.ibm.com/api/v1/web/120005054%40sastra.ac.in_dev/default/jarvisaction

HTTP Method	Auth	URL
POST	API-KEY	https://us-south.functions.cloud.ibm.com/api/v1/namespaces/120005054%40sastra.ac.in_dev/actions/jarvisaction

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 provide 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).

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 HP laptop manual.

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

Click the Create intent button.

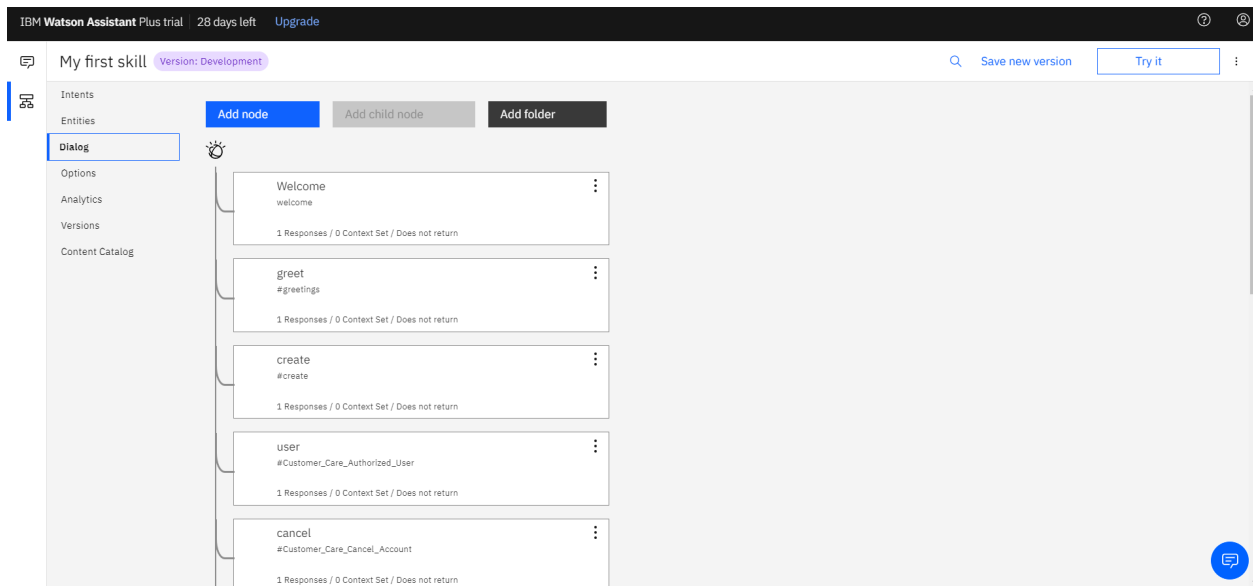
Name the intent #Product_Information, and at a minimum, enter the following example questions to be associated with it.

The screenshot shows the IBM Watson Assistant Plus trial interface. The breadcrumb navigation is "#product_information". The "Intent name" field is set to "#product_information". The "Description (optional)" field is empty. The "User example" field contains the text "Type a user example here, e.g. I want to pay my credit card bill". Below the input fields are buttons for "Add example" and "Show recommendations". A table lists user examples, with the first entry being "ACCOUNTS" added 2 days ago. The interface includes a header with "IBM Watson Assistant Plus trial", "28 days left", and an "Upgrade" button. A footer shows "Showing 1-35 of 35 examples" and pagination controls.

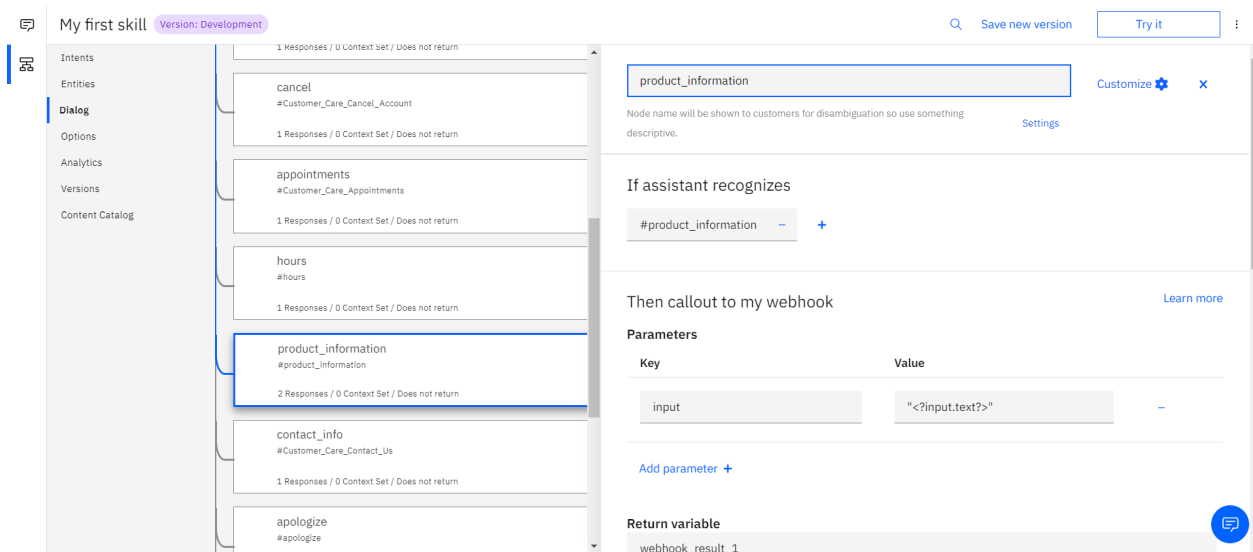
	Added ↑↓	Conflicts (0) ↑↓
<input type="checkbox"/> User examples (35) ↑		
<input type="checkbox"/> ACCOUNTS	2 days ago	

Create dialog node

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



Name the node "Product Information" [1] and assign it our new intent[2].



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

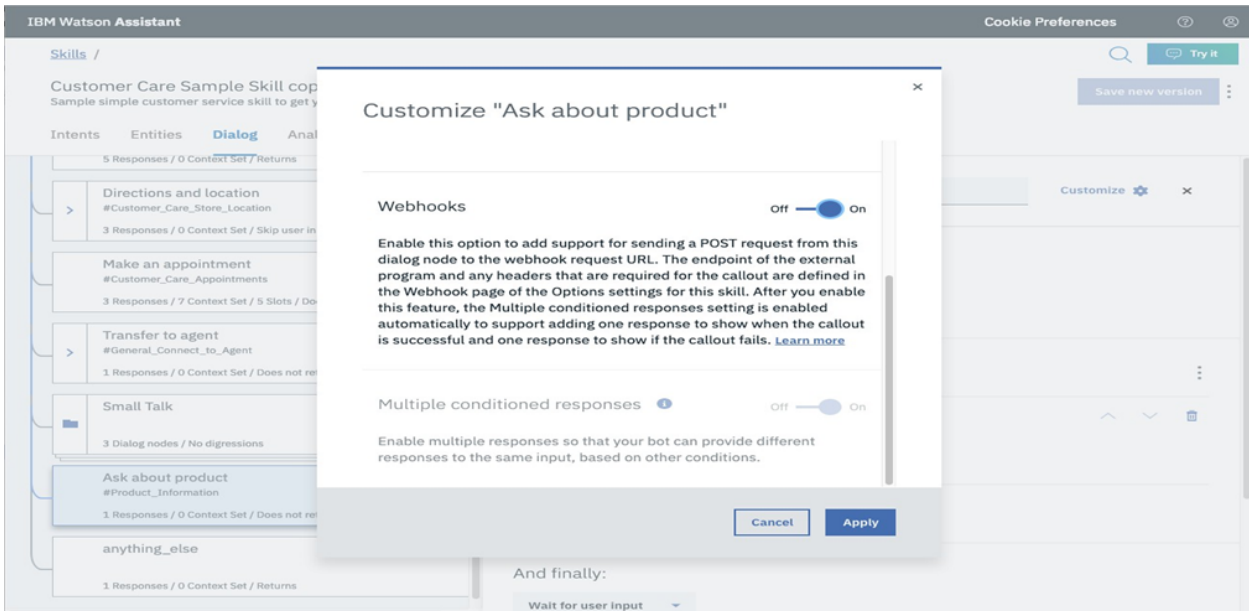
Setup access to our WebHook for the IBM Cloud Functions action you created in Step#4.

Select the Options tab[1]:

Enter the public URL end point 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.

The screenshot shows the IBM Watson Assistant interface for a skill named "My first skill" (Version: Development). On the left, a sidebar lists various components: Intents, Entities, Dialog, Options, Analytics, Versions, and Content Catalog. The "Dialog" section is selected, and a list of intents is displayed. The "product_information" intent is highlighted, showing its details: #product_information, 2 Responses / 0 Context Set / Does not return.

The main configuration area for the "product_information" intent is shown on the right. It includes a "Node name" field set to "product_information", a "Node description" field, and a "Settings" link. Below this, the "If assistant recognizes" section shows the intent name "#product_information" with a plus sign to add more. The "Then callout to my webhook" section has a "Learn more" link. The "Parameters" section is a table with two columns: "Key" and "Value". It contains one row with "input" as the key and "<?input.text?>" as the value. There is an "Add parameter +" link below the table. The "Return variable" section shows "webhook_result_1".

This screenshot shows the same IBM Watson Assistant interface, but with the "Assistant responds" section configured. The "Return variable" section still shows "webhook_result_1". The "Assistant responds" section has a table with two columns: "If assistant recognizes" and "Respond with". It contains two rows:

	If assistant recognizes	Respond with	
1	\$webhook_result_1	"<?\$webhook_result_1.passaj	Settings -
2	anything_else	Try again	Settings -

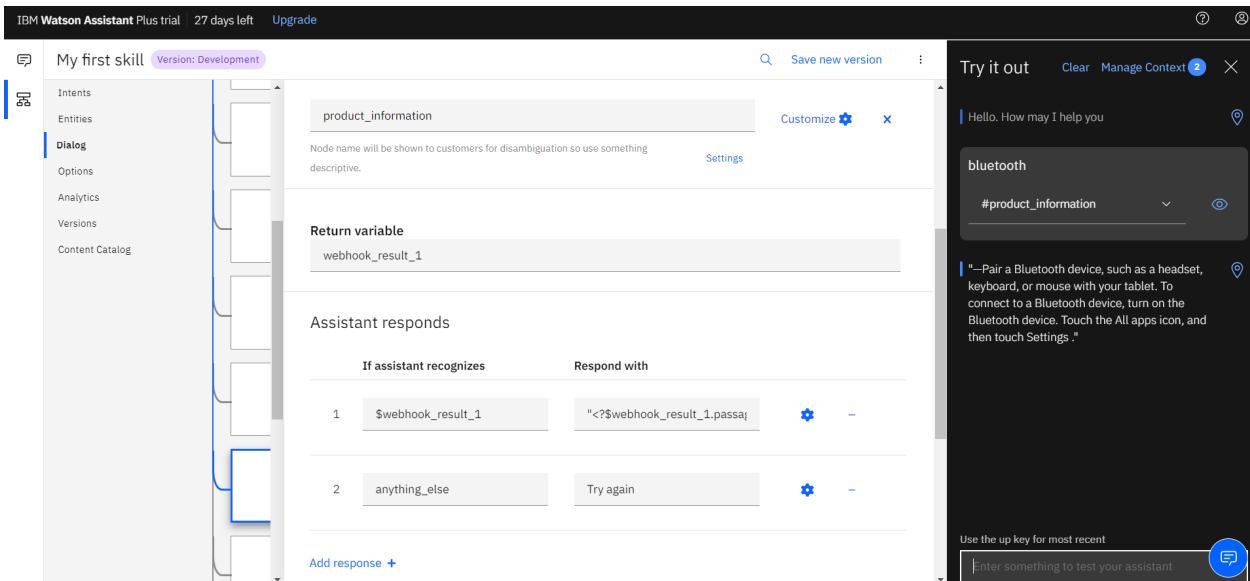
There is an "Add response +" link at the bottom of the table. The top of the interface shows the "IBM Watson Assistant Plus trial | 27 days left | Upgrade" banner.

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.

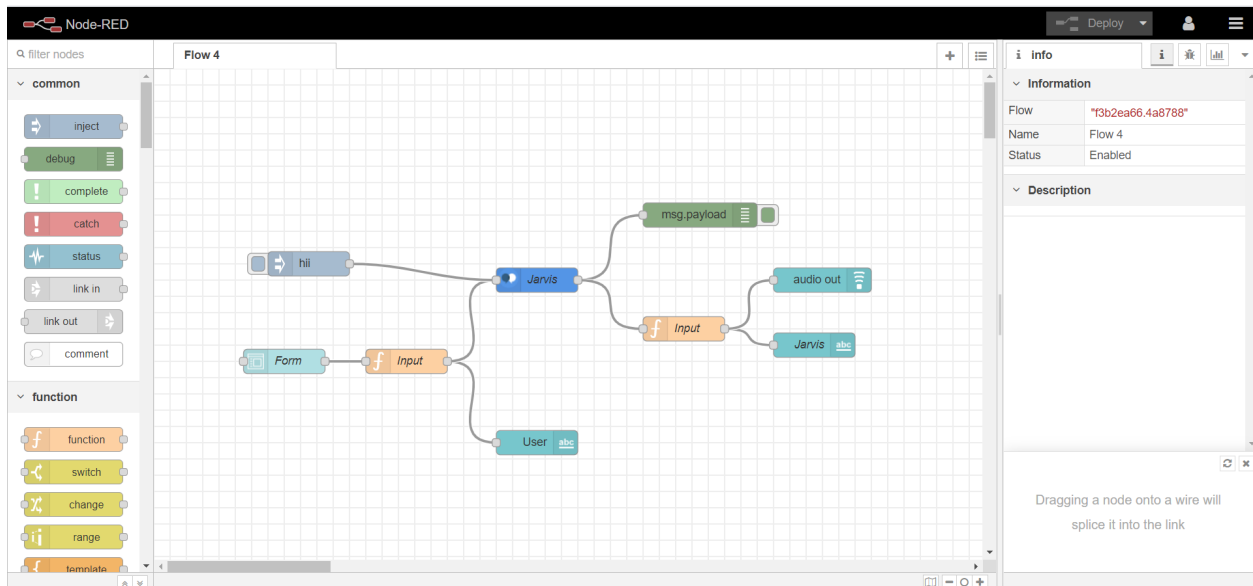
Optionally, you can add these responses to aid in debugging:

Add "<?webhook_result_1.passages[0].passage_text?>" in respond within Assistant responds block as shown below.



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 work space id of your Watson assistant service
- After entering all the information click on Done
- Drag inject node onto the flow from the Input section
- Drag Debug onto the flow from the output section
- Double-click on the inject node
- Select the payload as astring
- 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

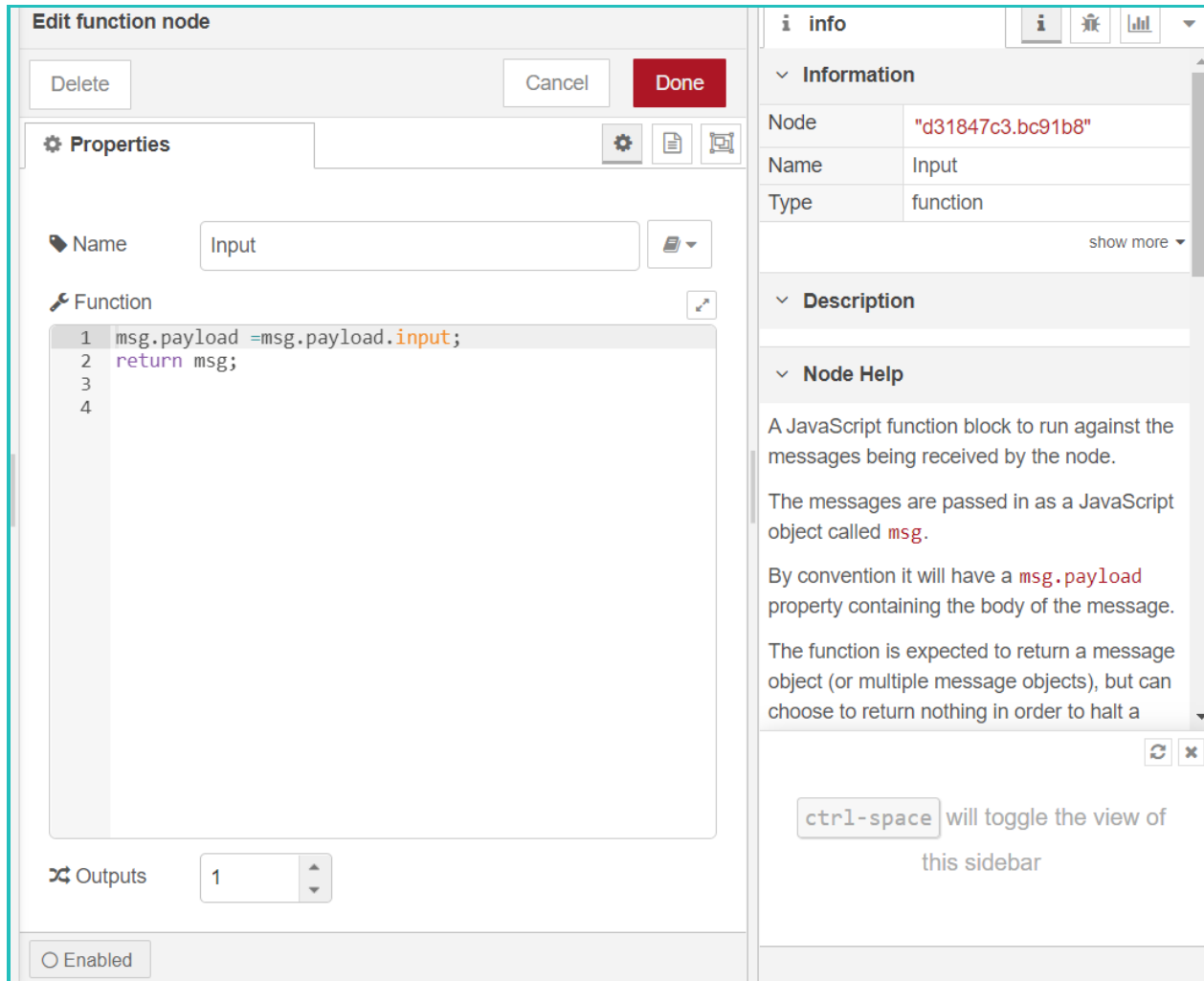
We are done integrating Watson assistant service to Node-red. In the next tab, we will create a web application using Node-red for the chatbot. For creating a web application UI we need “dashboard” nodes which should be installed manually.

- Goto navigation panel 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

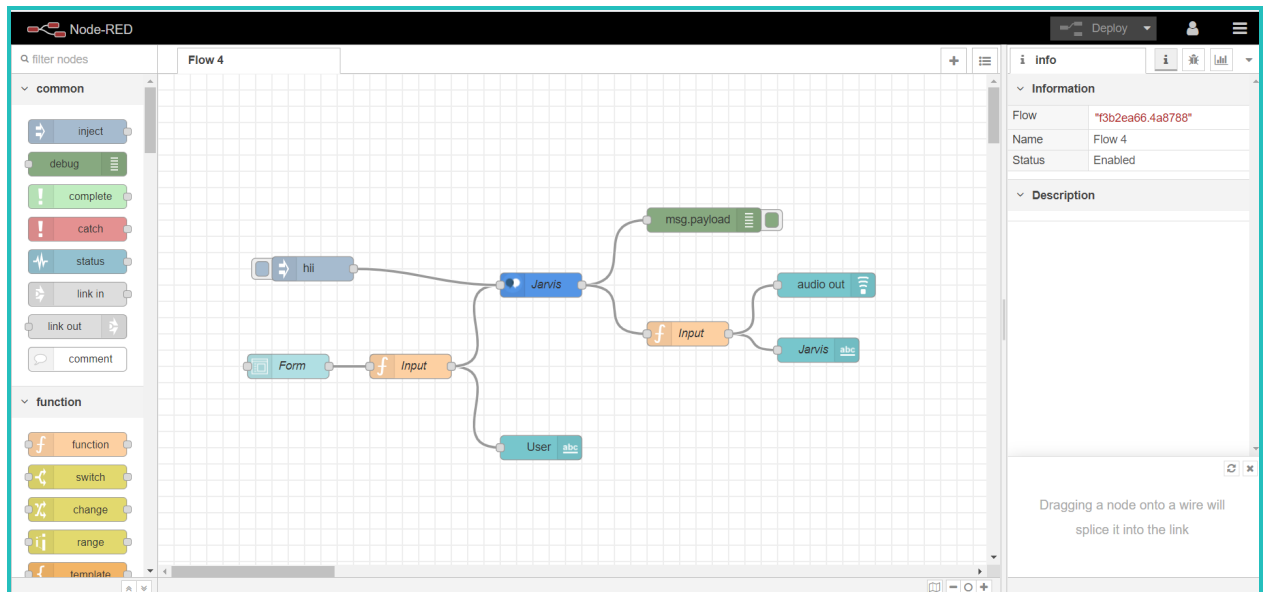
- Connect the output of “input parsing” function node to “You” textnode and output of “Parsing” function node to the input of “Bot” text node.
- Click on Deploy

5.FLOWCHART

1.Create flow and configure node:

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 dashboard integrates all the components and displayed in the Dashboard UI by typing **Url** :<https://node-red-tjxoz.mybluemix.net/ui/>

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:

- Sometimes chatbot can mislead customers
- Giving same answer for different sentiments.
- Sometimes 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 procedure and all we successfully created Intelligent help desk 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 hands free. This is one of the future scope of this project.

11. BIBILOGRAPHY APPENDIX

Cloud function *Node.js 10* code for discovery integration webhook generation:

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

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);  
});  
});  
}
```