

Intelligent Customer Help Desk with Smart Document Understanding

Name: RUDRA PRATAP SINGH

Category: Machine Learning

Mail id: rr4737@srmist.edu.in

College: SRM Institute Of Science and Technology

Internship At Smartinternz.com@2020

INDEX

INTRODUCTION

1-Overview

2- Purpose

LITERATURE SURVEY

1-Existing Problem

2- Proposed System

THEORITICAL ANALYSIS

1-Block Diagram

2-Hardware and software designing

EXPERIMENTAL INVESTIGATIONS

FLOWCHART

RESULT

ADVANTAGES & DISADVANTAGES

APPLICATIONS

CONCLUSION

FUTURE SCOPE

BIBILOGRAPHY APPENDIX

a. Sourcecode

b. Reference

1. INTRODUCTION

1. Overview:

Here we are going to prepare a chat bot by the use of ibm cloud services and the Watson services like assistant and discovery.

Services we are using in making chat bot are Discovery , Assistant, Cloud function and Node Red. By the end of the project, we'll learn how to make chat bot and best practices of integration of Watson services, and how they can build interactive information retrieval systems with Discovery and Assistant with the combination of functions and integrated to node red.

Project Requirements: Python, IBM Cloud, IBMWatson

Functional Requirements: IBMcloud

Technical Requirements: AI,ML,WATSONAI,PYTHON

Software Requirements: Watson assistant, Watsondiscovery,node-red.

Project Deliverables: Smartinternz Internship

Project Team: Garimella Sri Krishna Priya

Project Duration:19days

2. PURPOSE :

The typical customer care chat bot 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.

i. Scope of Work

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

2. LITERATURE SURVEY

1. Existing problem:

A good customer Chat bot 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 chat bot so that it will take care of real involvement of customer agent and customer can clarify his doubts with fast chat bots. In general we can say that the chat bot is the platform where the user can clarify his various doubts about the product which he/she want to purchase and they have the full rights to know about the product. Here we have to clarify the user doubts without the help of the agent and if the chat bot can't answer the queries they can contact to the agent or any other person.

2. Proposed solution:

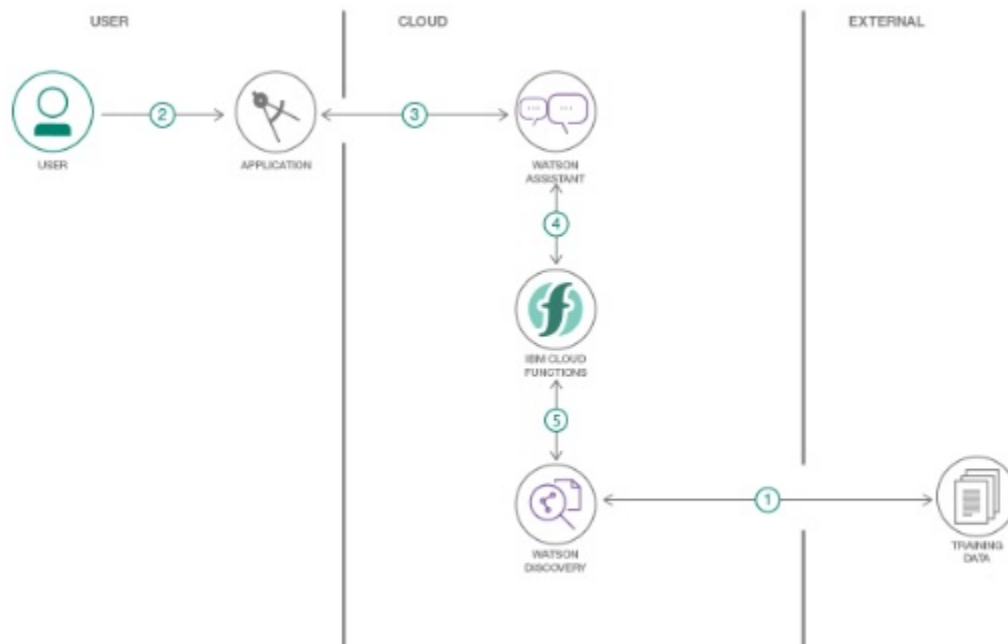
For the above problem to get solved we have to put an virtual agent in chat bot so it can understand

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 have used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED.

We use function to integrate the discovery and then the assistant and then the integration of all these are done in node-red. The discovery contains the manuals where we train them and then with the help of functions we integrate it to assistant.

3. THEORETICAL ANALYSIS

1-Block/FlowDiagram:



1. The document is annotated using Watson Discovery SDU.
2. The user interacts with the backend server via the app UI. The front end app UI is a chat bot 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.

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:

a. Watson Discovery :

The screenshot displays the IBM Watson Discovery-9r console within a web browser. The browser's address bar shows the URL: `cloud.ibm.com/services/discovery/crn%3Av1%3Abluemix%3Apublic%3Adiscovery%3Aeu-gb%3Aa%2F829e2870e7ed4869be2b57c59e05dc31%3A607...`. The console header includes the IBM Cloud logo, a search bar, and navigation links for Catalog, Docs, Support, and Manage. The user's name, RUDRA PRATA..., is visible in the top right corner.

The main content area is titled "Discovery-9r" and indicates the service is "Active". It features a "Details" link and an "Actions..." dropdown menu. A left-hand sidebar under the "Manage" section lists options: "Getting started", "Service credentials", "Plan", and "Connections".

The central panel is divided into two main sections. The top section, "Start by launching the tool", contains three buttons: "Launch Watson Discovery" (in blue), "Getting started tutorial" (with an external link icon), and "API reference". To the right of this section is a "Plan Lite" card with an "Upgrade" button. A vertical "FEEDBACK" button is located on the far right edge of the console.

The bottom section, "Credentials", includes links for "Download" and "Show credentials". It displays the "API key:" field with a masked value and a copy icon. Below it, the "URL:" field shows the endpoint: `https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/607e8ff2-83b5-4b5f-8b27-64a2293e...`, also with a copy icon.

The Windows taskbar at the bottom of the screen shows the search bar, task view button, and several application icons. The system tray on the right indicates the language is "ENG IN", the time is "7:23 PM", and the date is "5/20/2020".

b. WatsonAssistant :

The screenshot displays the IBM Watson Assistant console interface. At the top, the browser address bar shows the URL: `cloud.ibm.com/services/conversation/crn%3Av1%3Abluemix%3Apublic%3Aconversation%3Aeu-gb%3Aa%2F829e2870e7ed4869be2b57c59e05d...`. The console header includes the IBM Cloud logo, a search bar, and navigation links for Catalog, Docs, Support, and Manage. The user's name, RUDRA PRATA..., is visible in the top right corner.

The main content area is titled "Watson Assistant-y3" and indicates it is "Active". A left sidebar under the "Manage" section lists "Service credentials", "Plan", and "Connections". The main panel is divided into two sections:

- Start by launching the tool:** This section contains three buttons: "Launch Watson Assistant" (blue), "Getting started tutorial" (with an external link icon), and "API reference".
- Plan Plus Trial:** This section features an "Upgrade" button.

Below these sections is the "Credentials" area, which includes links for "Download" and "Show credentials". It displays the "API key:" field with a masked value and a copy icon, and the "URL:" field with the value `https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/7ca722df-e852-4924-a014-746eea5a...` and a copy icon. A note at the bottom states: "View all credentials in the Service credentials tab."

The Windows taskbar at the bottom shows the search bar, task view button, and several application icons. The system tray on the right indicates the language is "ENG IN" and the time is "7:27 PM 5/20/2020".

c. Node-Red:

The screenshot displays the IBM Cloud Developer console interface. The browser tabs at the top include 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom...', and 'IBM App Development'. The address bar shows the URL 'cloud.ibm.com/developer/appservice/apps/e569f21e-73c2-449c-839e-50451688a52c'. The main header of the console shows 'IBM Cloud' and a search bar. The page title is 'Node RED .ML Project' with an 'Add tags' link and an 'Actions...' dropdown menu.

The 'Details' section on the left lists the following information:

- App URL: <https://node-red-ml-project.mybluemix.net>
- Source: <https://eu-gb.git.cloud.ibm.com/rr4737/NodeREDMLProject>
- Resource group: Default
- Deployment target: Node RED .ML Project
- Created: 5/7/2020

The 'Services (1)' section shows a table with one service:

Name	Resources	Actions
Cloudant	Documentation	

The 'Deployment Automation' section on the right shows:

- Name: NodeREDMLProject
- Location: London
- Tool integrations: (Icons for various tools)
- Delivery Pipelines: Name: NodeREDMLProject, Status: No stages detected

The 'Credentials' section shows a list of credentials with a 'Show' button. The list contains one credential:

```
{  "cloudant": [    {      "name": "node-red--ml-project-cloudant-15888711528",      "credentials": {}    }  ]}
```

The bottom of the screenshot shows the Windows taskbar with a search bar and various application icons. The system tray on the right indicates the language is 'ENG IN' and the time is '7:33 PM 5/20/2020'.

2. Configure WatsonDiscovery

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, let's 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 screenshot shows the IBM Watson Discovery web interface. The browser tabs include 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom...', 'IBM Watson Service Page', and 'IBM Watson Discovery - Collectio...'. The URL bar shows a long alphanumeric string. The page header is 'IBM Watson Discovery' with a 'Cookie Preferences' link and an 'Instance: Discovery-9r' dropdown. The main content area is titled 'Intelligent Customer Help Desk with Smart Document' and has a 'Configure data' button. Below this, there are tabs for 'Overview', 'Errors and warnings (133)', and 'Search settings'. The 'Overview' tab is active, showing '133 documents'. To the right, it says '0 documents failed' with a 'View details' link, and 'Created on 5/13/2020 8:08:27 am EDT' and 'Last updated 5/13/2020 8:08:27 am EDT'. There is an 'Upload documents' button. The main section is divided into three columns. The left column, 'Identified 5 fields from your data', lists 'footer', 'subtitle', 'table_of_contents', 'text', and 'title'. Below this is a link 'Need to identify more fields? Add fields'. The middle column, 'Added 4 enrichments to your data', shows 'Entity Extraction' with results '0.3°C (4) | 0.5°F (4) | 10 °F (4) | 900 seconds (4) | 20 min (3)' and 'Sentiment Analysis' with a bar chart showing 53% positive, 33% neutral, and 14% negative. The right column, 'Now you're ready to query!', has two sections: 'Top people related to /technology and computing/operating systems' with a 'Run' button, and 'Most common entity types and their top entities' with a 'Run' button. At the bottom, there is a search bar with the text 'Type here to search' and a Windows taskbar with various icons and a system clock showing 7:39 PM on 5/20/2020.

Click the Build your own query [1] button.

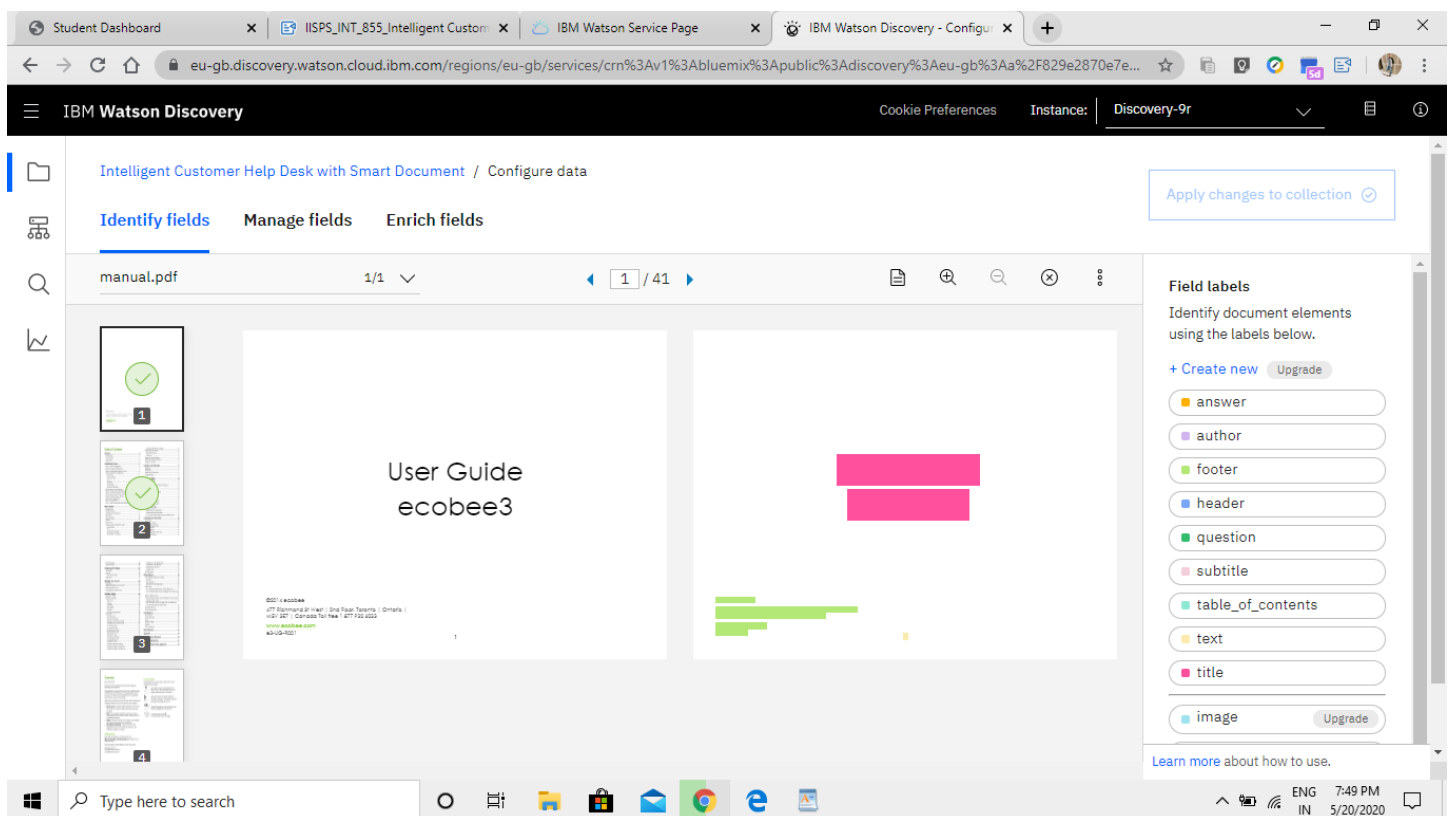
Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

The screenshot shows the IBM Watson Discovery web interface. The browser tabs include 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom...', 'IBM Watson Service Page', and 'IBM Watson Discovery - Query B...'. The address bar shows the URL: 'eu-gb.discovery.watson.cloud.ibm.com/regions/eu-gb/services/crn%3Av1%3Abluemix%3Apublic%3Adiscovery%3Aeu-gb%3Aa%2F829e2870e7e...'. The interface has a dark header with 'IBM Watson Discovery', 'Cookie Preferences', 'Instance: Discovery-9r', and a 'Train Watson to improve results' link. The main content area is titled 'Intelligent Customer Help Desk with Smart Document / Build queries'. It contains a search bar with the text 'text:Heat, text:!HVAC' and a 'Use a sample query' button. Below the search bar are two buttons: 'Run query' and 'Close'. On the right side, there is a 'Summary' tab and a 'JSON' tab. The 'Summary' tab shows a 'Query URL' and a 'Passages' section with three text excerpts related to heat pump operation.

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.

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

is the current page being annotated.

is where you select text and assign it a label.

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.

The screenshot displays the IBM Watson Discovery 'Manage fields' configuration page. The browser tabs at the top include 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom...', 'IBM Watson Service Page', and 'IBM Watson Discovery - Configur...'. The URL bar shows a long alphanumeric string. The page header indicates 'Intelligent Customer Help Desk with Smart Document / Configure data' and includes an 'Apply changes to collection' button. The main content area is divided into two sections. The left section, 'Identify fields to index', lists various fields with toggle switches: 'answer' (Off), 'author' (Off), 'footer' (Off), 'header' (Off), 'image' (Off), 'question' (Off), 'subtitle' (On), 'table' (On), 'table_of_contents' (On), 'text' (On), and 'title' (On). The right section, 'Improve query results by splitting your documents', contains a text box explaining document splitting and a dropdown menu labeled 'Split document on each occurrence of' with 'subtitle' selected.

Here is where you tell Discovery which fields to ignore. Using the on/off buttons, turn off all labels except subtitles and text.

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 configuration page for a collection named 'Discovery-9r'. The interface is divided into several sections:

- Overview:** Shows 133 documents, 0 documents failed, and a button to 'Upload documents'.
- Identified 5 fields from your data:** A list of fields with checkboxes: footer, subtitle, table_of_contents, text, and title. A link 'Add fields' is provided.
- Added 4 enrichments to your data:**
 - Entity Extraction:** Shows results for temperature (0.3°C, 0.5°F, 10°F) and time (900 seconds, 20 min).
 - Sentiment Analysis:** Shows 53% positive, 33% neutral, and 14% negative sentiment.
 - Concept Tagging:** Shows tags like Heat, Internet, HVAC, Heat pump, and Thermodynamics.
 - Category Classification:** Shows a category 'technology and com... operating systems'.
- Now you're ready to query!** Three buttons are available: 'Run' for 'Entities of type Quantity which have negative sentiment', 'Run' for 'Top entities with their average, min, max sentiment score', and 'Run' for 'Most common entity types and their top entities'.

The bottom of the screen shows a Windows taskbar with a search bar and various application icons.

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

The screenshot displays the IBM Watson Discovery 'Build queries' interface. On the left, the 'Search for documents' section shows a query input field containing 'text:Heat, text:IHVAC'. Below this, there are options to 'Include analysis of your results' and 'Filter which documents you query'. A 'Run query' button is visible at the bottom of the query builder. On the right, a 'Summary' panel shows the 'JSON' response of the query. The response is a JSON object with the following structure:

```
{
  "matching_results": 133,
  "session_token": "1_09InTEr5Y1ayoTv3_xrc2EdmX",
  "passages": [],
  "results": [
    {
      "id": "af48ac7e2cdd79c388a470d1c3f9d339_1",
      "result_metadata": {...},
      "subtitle": [...],
      "text": "",
      "table_of_contents": [...],
      "extracted_metadata": {...},
      "segment_metadata": {...}
    },
    {
      "id": "af48ac7e2cdd79c388a470d1c3f9d339_2",
      "result_metadata": {...},
      "subtitle": [...],
      "text": "Do you hear that? That's the sound of hundreds of thousands of ecobee-ers welcoming you to the hive. Congratulations on the purchase of your new ecobee3 smarter wi-fi thermostat with remote sensor. This guide will provide an overview of the features and capabilities of the new product and will help you get up and running. Make sure you also take a look at the Quick Start Guide and the Installation Guide that come in the box with your ecobee3. Remote sensors - Deliver the right temperature in the rooms that matter most as well as detect when these rooms are occupied. Smart - Your ecobee3 understands your home's unique energy profile and the weather outside, making sure you're comfortable at all times. Intuitive - With its 3.5\" full color touch display, your ecobee3 has intuitive controls just like your smartphone. Accessible from anywhere - Monitor and control the temperature in your home anytime, anywhere, on your smartphone, tablet, or computer."
    }
  ]
}
```

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:

Student Dashboard | IISPS_INT_855_Intelligent Custom... | IBM Watson Service Page | IBM Watson Discovery - Query B... | +

eu-gb.discovery.watson.cloud.ibm.com/regions/eu-gb/services/crn%3Av1%3Abluemix%3Apublic%3Adiscovery%3Aeu-gb%3Aa%2F829e287...

IBM Watson Discovery

Intelligent Customer Help Desk with Smart Document

Overview | Errors and warnings (133) | Search settings

133 documents

0 documents failed
[View details](#)

Created on 5/13/2020 8:08:27 am EDT
Last updated 5/13/2020 8:08:27 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 4 enrichments to your data

Entity Extraction

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

Concept Tagging

Heat (1.7) | Internet (1.2) | HVAC (1.1) |
Heat pump (1.1) | Thermodynamics (1.0)

5 enrichments available. [Add enrichments](#)

Sentiment Analysis

53% positive 33% neutral 14% negative

Category Classification

technology and com... operating systems

Now you're ready to query!

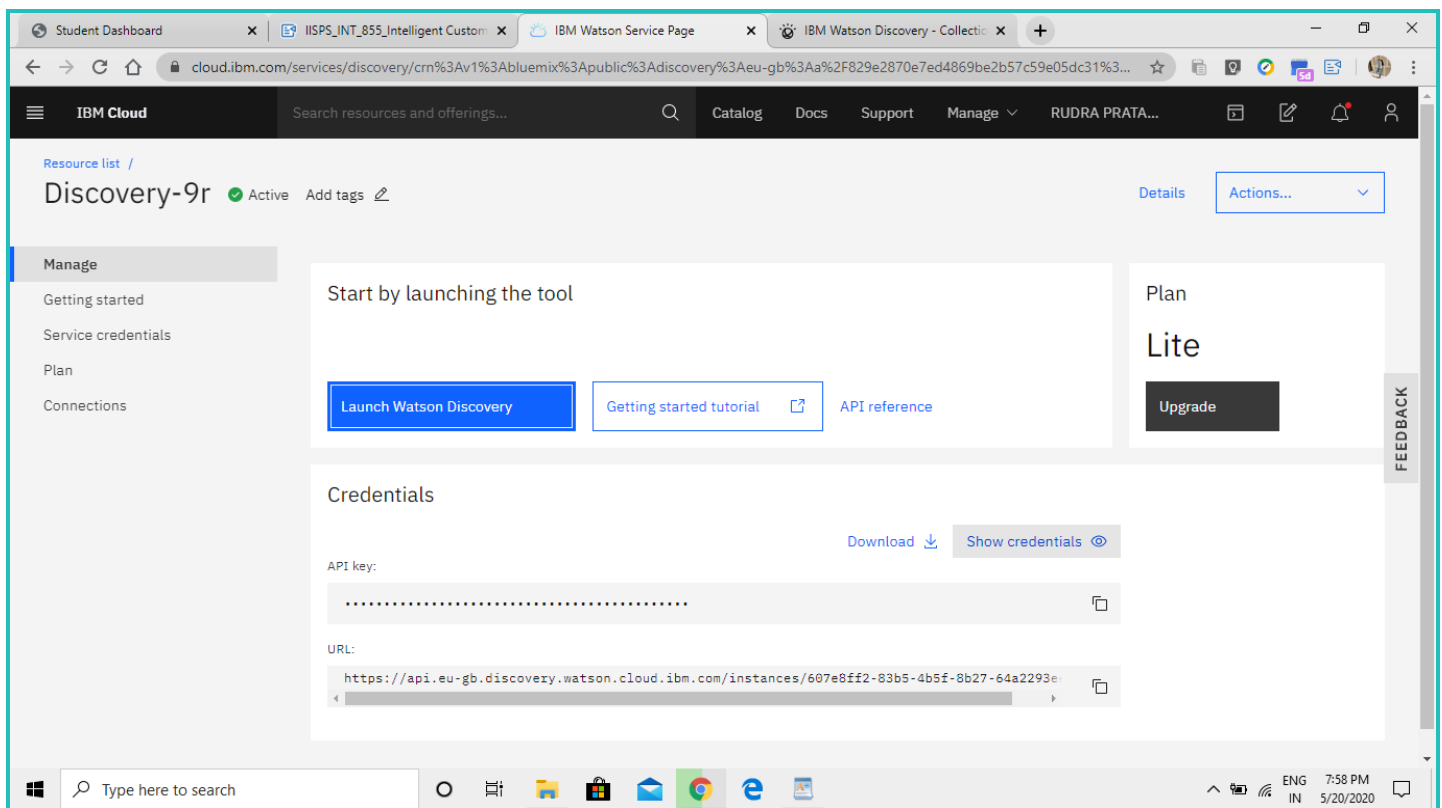
Entities of type **Quantity** which have negative sentiment
[Run](#)

Top entities with their average, min, max sentiment score
[Run](#)

Entities of type **Quantity** which have positive sentiment
[Run](#)

[Build your own query](#)

For credentials, return to the main panel of your Discovery service, and click the Service credentials tab.

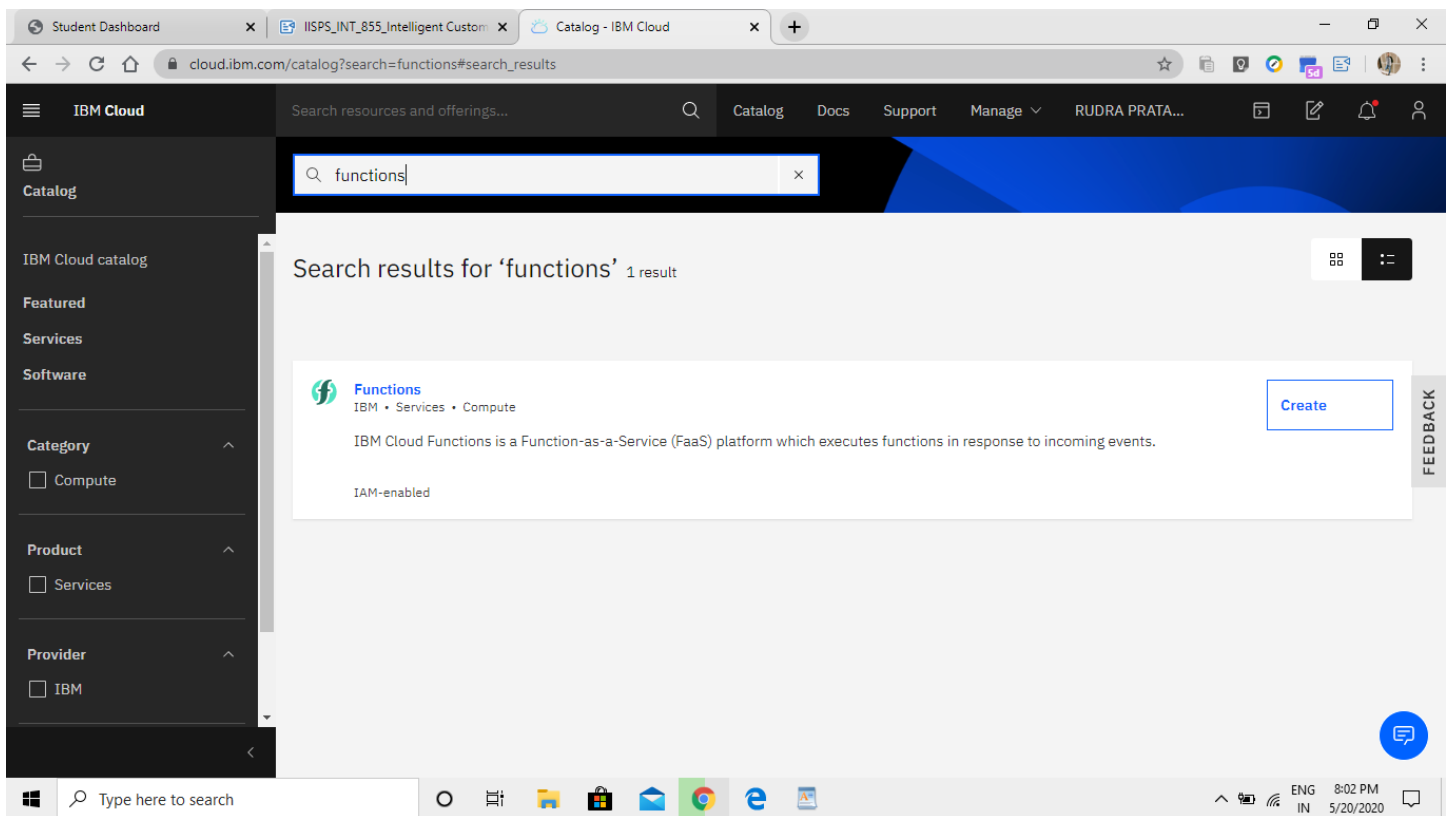


Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

3. Create IBM Cloud Functions action

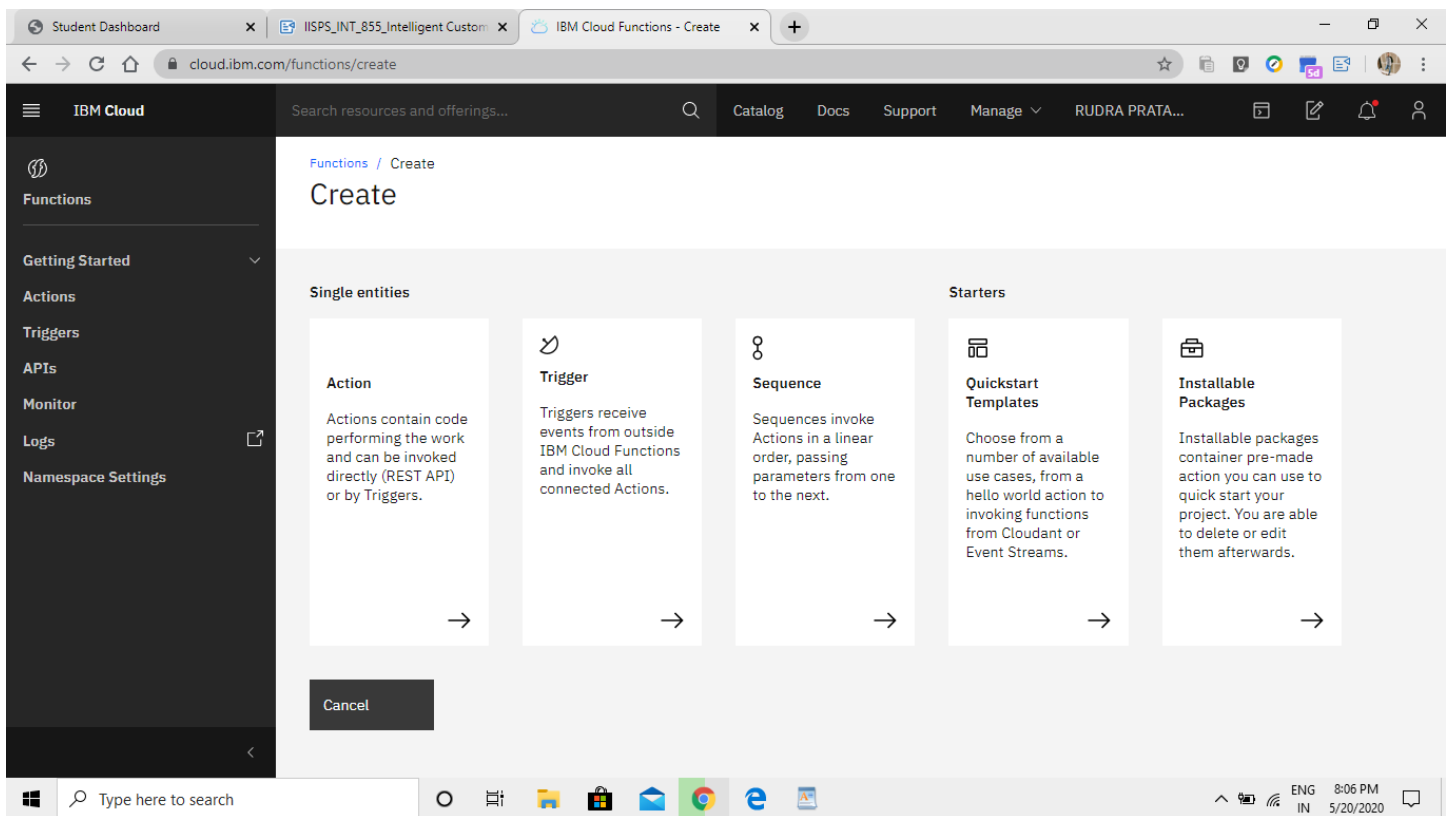
Now let's create the web action that will make queries against our Discovery collection.

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.



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

Student Dashboard | IISPS_INT_855_Intelligent Custom... | IBM Cloud Functions - Actions

cloud.ibm.com/functions/actions

IBM Cloud

Search resources and offerings...

Actions

rr4737@srmist.edu.in_dev
Dallas (CF-Based)

Search Actions

Create

Default Package

Name	Runtime	Web Action	Memory	Timeout
Intelligent Customer Help Desk with Smart Document Understanding	Node.js 10	Enabled	256 MB	60 s

Items per page: 10 | 1-1 of 1 items | 1 | 1 of 1 pages

Student Dashboard | IISPS_INT_855_Intelligent Custom... | IBM Cloud Functions - Actions

cloud.ibm.com/functions/details/action/rr4737%2540srmist.edu.in_dev//Intelligent%2520Customer%2520Help%2520Desk%2520with%252...

Intelligent Customer Help Desk with Smart Document Understanding

Web Action

Namespace: rr4737@srmist.edu.in_dev(Dallas)

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

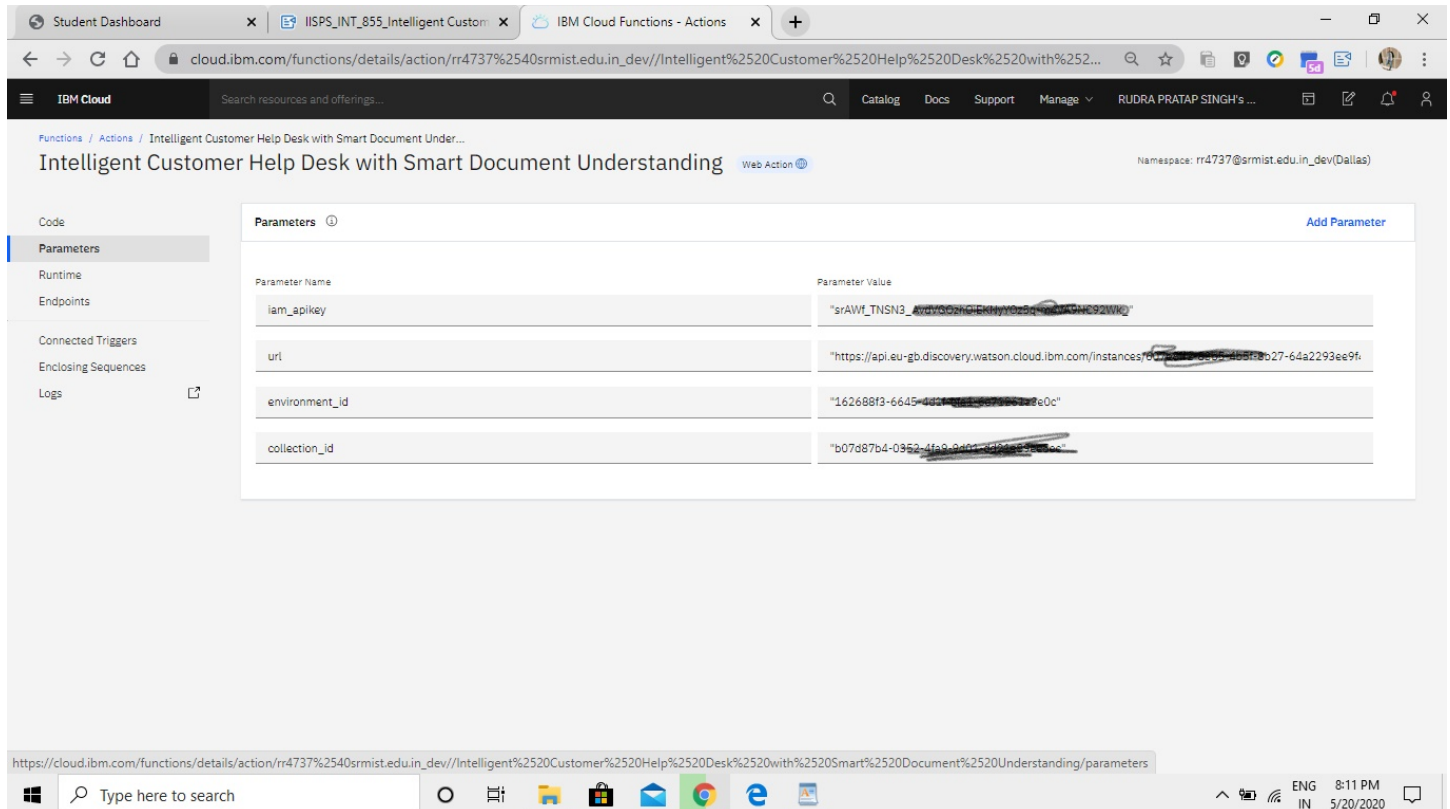
Logs

```
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=     else {
42=       discovery = new DiscoveryV1({
```

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 displays the IBM Cloud Functions console for the action 'Intelligent Customer Help Desk with Smart Document Understanding'. The left sidebar shows the 'Code' panel selected. The main area is divided into two sections: 'Code' and 'Activations'.

Code Panel: The code is written in JavaScript and uses the 'watson-developer-cloud/discovery/v1' API. It defines a 'main' function that takes 'params' as input and returns a Promise. The code includes comments and a 'const' declaration for the 'DiscoveryV1' class.

```
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
30 function main(params) {
31   return new Promise(function (resolve, reject) {
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     else {
42       discovery = new DiscoveryV1({
43         'username': params.username,
44         'password': params.password,
45         'url': params.url,
46         'version': '2019-03-25'
47       });
48     }
49
50     // ... (rest of the code) ...
51   });
52 }
```

Activations Panel: The 'Activations' panel shows a single activation with the ID '42f94f9619704817b94f961970701719'. The activation was triggered on 5/20/2020 at 20:15:43. The 'Results' section shows the output of the function, which is a JSON object containing 'matching_results', 'passages', and 'results'.

```
{
  "matching_results": 133,
  "passages": [],
  "results": [
    {
      "extracted_metadata": {
        "author": {
          "name": "Nashib"
        },
        "file_type": "pdf",
        "file_name": "manual.pdf",
        "sha1": "d1a84cc00008e58e1ebca080c43faebc9fa7bc12"
      },
      "id": "af48ac7e2cd79c388a4701c3f9d339_1",
      "result_metadata": {
        "confidence": 0,
        "score": 1
      },
      "segment_metadata": {
        "id": "8f960c68-524e-4637-b840-0b124feb71e",
        "parent_id": "af48ac7e2cd79c388a4701c3f9d339",
        "total_segments": 133
      },
      "subtitle": {
        "table_of_contents": [
          "Overview",
          "Getting Help",
          "Touch Screen",
          "Web Portal",
          "Guided Setup Process",
          "5 Step 1. Wiring Configuration"
        ]
      }
    }
  ]
}
```

Next, go to the Endpoints panel [1]:

The screenshot shows the IBM Cloud Functions console for an action named 'Intelligent Customer Help Desk with Smart Document Understanding'. The 'Web Action' tab is selected, and the 'Enable as Web Action' checkbox is checked. The URL is `https://us-south.functions.cloud.ibm.com/api/v1/web/rr4737%40srmist.edu.in_dev/default/Intelligent%20Customer%20Help%20Desk%20with%20Smart%20Document%20Understanding`. The REST API section shows a POST method with an API key. The CURL section shows a curl command: `curl -X POST https://us-south.functions.cloud.ibm.com/api/v1/web/rr4737%40srmist.edu.in_dev/default/Intelligent%20Customer%20Help%20Desk%20with%20Smart%20Document%20Understanding?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_enquiry, and at a minimum, enter the following example questions to be associated with it.

The screenshot shows a web browser window with multiple tabs. The active tab is 'IBM Watson Assistant'. The address bar shows the URL: eu-gb.assistant.watson.cloud.ibm.com/eu-gb/cm/v1:bluemixpublic:conversation:eu-gb:a~2F829e2870e7ed4869be2b57c59e05dc31:7ca722df-e8... The page header indicates 'IBM Watson Assistant Plus trial | 26 days left | Upgrade'. The main content area is titled '#product_enquiry' and shows the 'Intent name' field with the value '#product_enquiry'. Below this is the 'Description (optional)' field with the placeholder text 'Add a description to this intent'. The 'User example' section has the placeholder text 'Type a user example here, e.g. I want to pay my credit card bill'. At the bottom of the form, there are two buttons: 'Add example' and 'Show recommendations'. Below the form, a message states 'No examples yet. Train your virtual assistant with this intent by adding unique examples of what your users would say.' The Windows taskbar is visible at the bottom of the screen.

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.

Student Dashboard x IISPS_INT_855_Intelligent Custom... x IBM Watson Service Page x IBM Watson Assistant x +

eu-gb.assistant.watson.cloud.ibm.com/eu-gb/crn:v1:bluemixpublic:conversation:eu-gb:a~2F829e2870e7ed4869be2b57c59e05dc31:7ca722df-e8...

IBM Watson Assistant Plus trial | 26 days left Upgrade

My first skill Version: Development Save new version Try it

Intents Entities Dialog Options Analytics Versions Content Catalog

Add node Add child node Add folder

Welcome
welcome
1 Responses / 0 Context Set / Does not return

#General
#General_Ending
1 Responses / 0 Context Set / Does not return

#General_Greetings
1 Responses / 0 Context Set / Does not return

Tell_me_a_joke
#General_Jokes || @General_jokes
1 Responses / 0 Context Set / Does not return

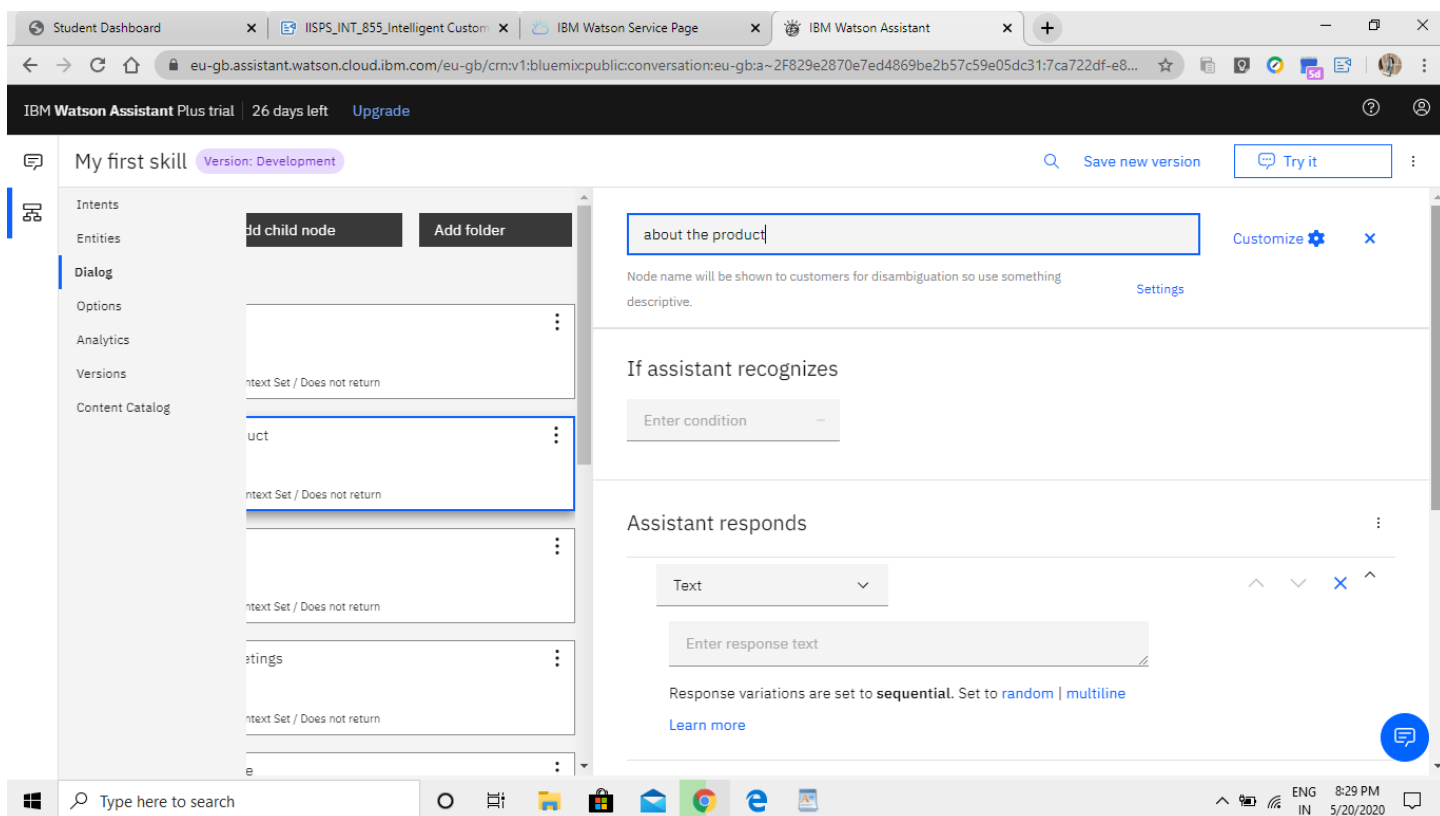
What is your name?

Add child node
Add node above
Add node below
Add folder
Move
Duplicate
Jump to
Delete

Type here to search

ENG IN 8:27 PM 5/20/2020

Name the node "Ask about product" [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.

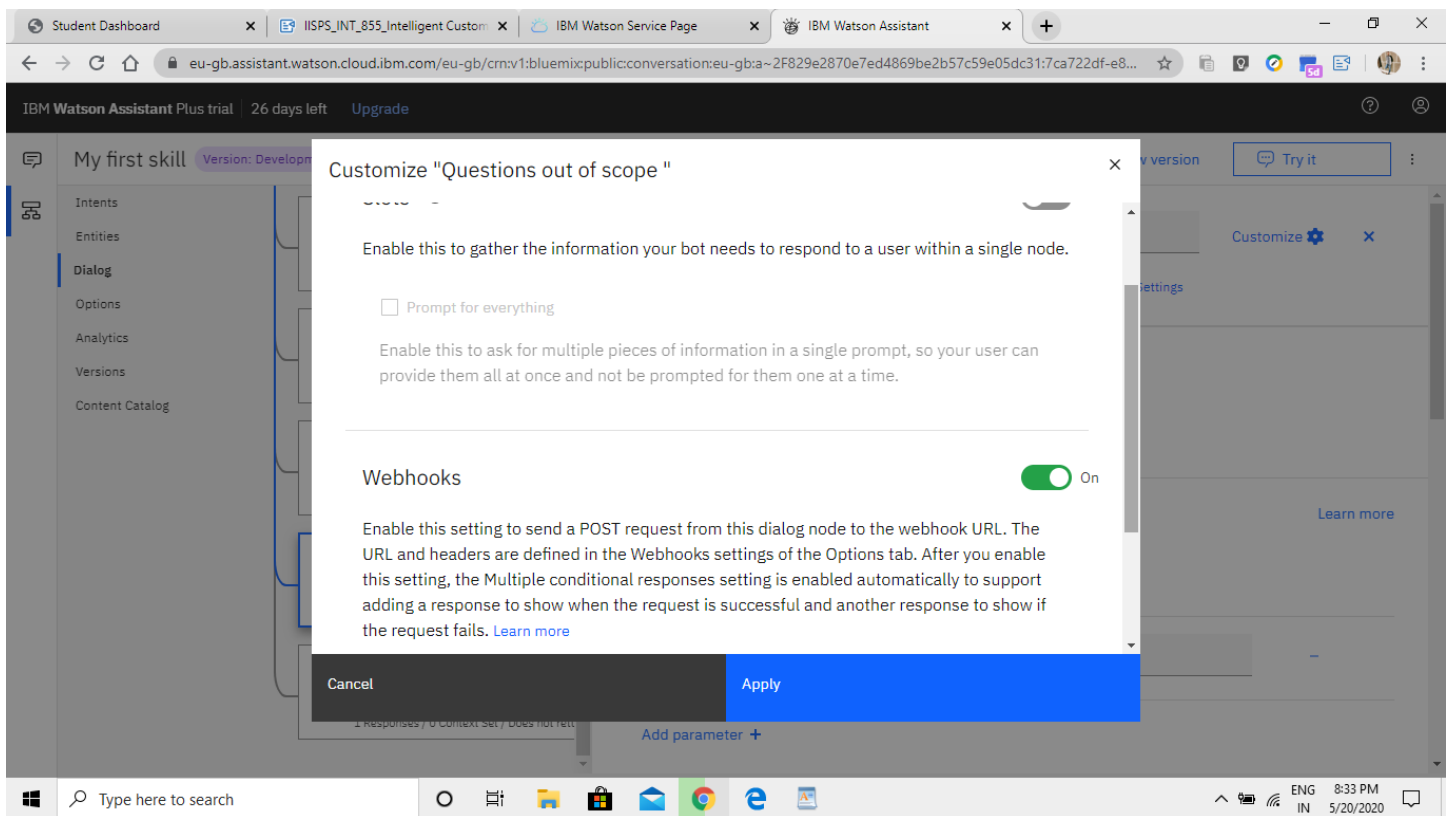
Enable webhook from Assistant

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 web interface. The browser's address bar displays the URL: `eu-gb.assistant.watson.cloud.ibm.com/eu-gb/crn:v1:bluemixpublic:conversation:eu-gb:a~2F829e2870e7ed4869be2b57c59e05dc31:7ca722df-e8...`. The page title is "My first skill" with a "Version: Development" tag. A sidebar on the left lists navigation options: Intents, Entities, Dialog, Options, Webhooks (selected), Disambiguation, Autocorrection, Irrelevance Detection, System Entities, Analytics, Versions, and Content Catalog. The main content area is titled "Webhooks" and includes a description: "A webhook is a mechanism that allows you to call out to an external program based on events in your dialog." Below this, the "Webhook setup" section prompts the user to "Specify the request URL for an external API you want to be able to invoke from dialog nodes." The "URL" field contains the text: `https://us-south.functions.cloud.ibm.com/api/v1/web/rr4737%40srmist.edi`. The "Headers" section states: "Add HTTP headers for authorization or any other parameters required for invoking the webhook." It features a table with columns "Header name" and "Header value", and buttons for "Add header" and "Add authorization". At the bottom of the main area, a "Next step" section is partially visible. The bottom of the screen shows the Windows taskbar with a search bar and various application icons, and the system tray indicating the time as 8:32 PM on 5/20/2020.

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.

The screenshot displays the IBM Watson Assistant web interface. The browser's address bar shows the URL: `eu-gb.assistant.watson.cloud.ibm.com/eu-gb/crn:v1:bluemixpublic:conversation:eu-gb:a~2F829e2870e7ed4869be2b57c59e05dc31:7ca722df-e8...`. The page title is "Intelligent Customer Help Desk with Smart Document Understanding" with a "Version: Development" tag. On the left, a sidebar menu includes "Intents", "Entities", "Dialog", "Options", "Analytics", "Versions", and "Content Catalog". The "Dialog" panel is active, showing a flowchart with several nodes. The selected node is "questions out of scope" with the intent "#Questions_out_of_scope" and "2 Responses / 0 Context Set / Does not return". Below it is the "anything_else" node. The right-hand panel shows the configuration for the selected node. It includes a text input field containing "about the product", a "Customize" button, and a "Settings" link. Below this is a "Parameters" table with one entry: "input" with the value "<?input.text?>". There is also a "Return variable" section with "webhook_result_1". At the bottom of the right panel is the "Assistant responds" section. The Windows taskbar at the bottom shows the search bar and various application icons, with the system clock indicating 8:35 PM on 5/20/2020.

You 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:

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:

The screenshot displays the IBM Watson Assistant console. The main window shows a dialog flow for 'Intelligent Customer Help Desk with Smart Document Understanding'. The flow includes nodes for 'Welcome message', 'Bot name', 'What are your hours?', and 'Where are you located?'. The 'Welcome message' node has a response of 'welcome'. The 'Bot name' node has a response of '#Bot_name || @Bot_name'. The 'What are your hours?' node has a response of '#Customer_Care_Store_Hours'. The 'Where are you located?' node has a response of '#Customer_Care_Store_Location'. The console also shows a 'Try it out' panel on the right with a chat interface. The chat interface shows a user input 'How to turn on heater?' and a response from the assistant: '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.' The chat interface also includes a 'Manage Context' button and a search bar.

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:

The screenshot displays the IBM Watson Assistant web interface. The browser tabs include 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom...', 'IBM Watson Service Page', and 'IBM Watson Assistant'. The URL bar shows a conversation ID. The interface features a sidebar with navigation options: Intents, Entities, Dialog (selected), Options, Analytics, Versions, and Content Catalog. The main area shows a dialog flow for 'Intelligent Customer Help Desk with Smart Document Understanding' (Version: Development). The flow includes several intents: '#Thanks', 'Please transfer me to an agent' (with entity '#General_Connect_to_Agent'), 'What can I do' (with entity '#Help'), 'Questions out of scope' (with entity '#Questions_out_of_scope'), and 'anything_else'. The 'Questions out of scope' intent is currently selected, showing its configuration. A text input field contains 'Questions out of scope |'. Below it, a note states: 'Node name will be shown to customers for disambiguation so use something descriptive.' and a 'Settings' link is present. A 'webhook_result_1' node is shown. The 'Assistant responds' section contains a table with two rows:

	If assistant recognizes	Respond with		
1	\$webhook_result_1	"<?\$webhook_result_1.passa	⚙️	—
2	anything_else	Try again later	⚙️	—

An 'Add response +' button is at the bottom of the table. The bottom of the screen shows a Windows taskbar with a search bar and various application icons. The system tray on the right indicates 'ENG IN', '8:44 PM', and '5/20/2020'.

4.Create flow and configure node:

Integration of Watson assistant in Node-RED:

1. Double-click on the Watson assistant node
2. Give a name to your node and enter the username, password and work space id of your Watson assistant service.

Edit assistant V2 node

Delete Cancel Done

Properties

Name My first assistant

Username Username

Password Password

API Key

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

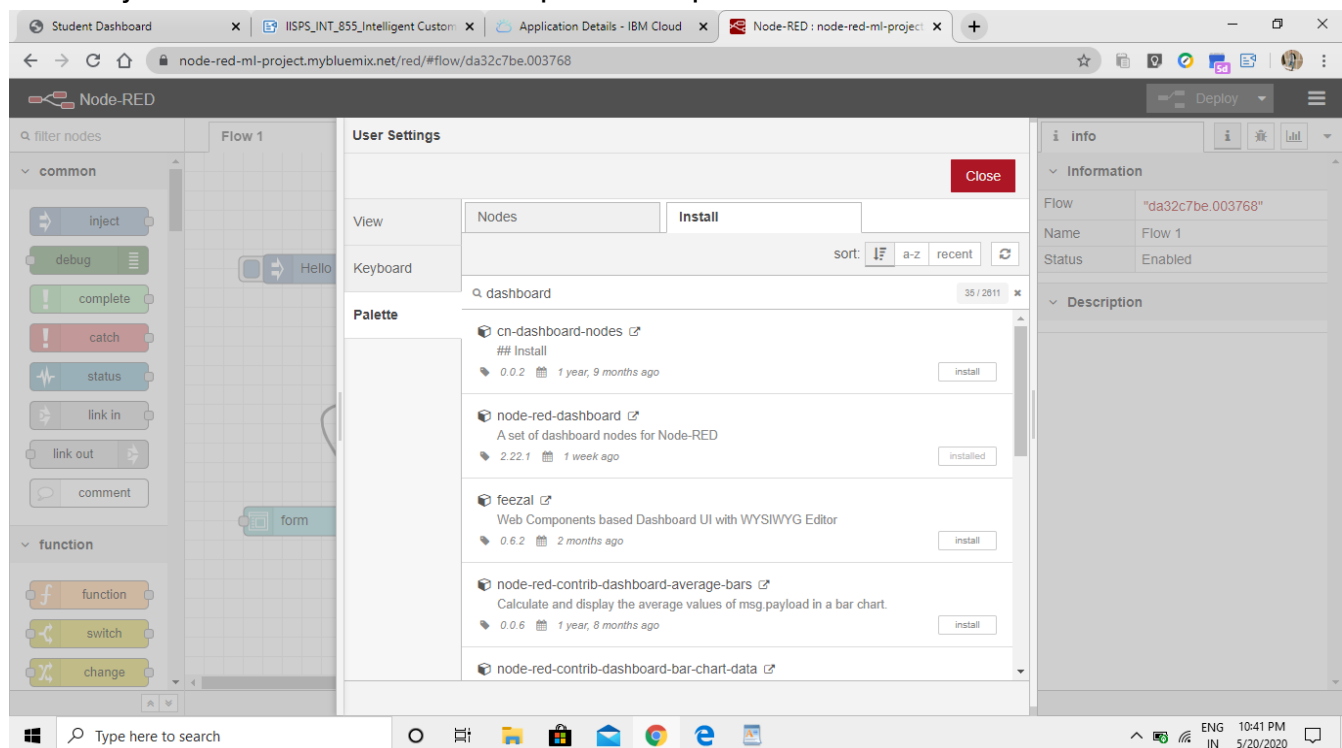
Assistant ID d3fa58b9-7946-4784-b120-5e025c58c9ba

Timeout Period Leave empty to disable

☐ Switch on Debug

☐ Enabled

1. After entering all the information click on Done
2. Drag inject node on to the flow from the Input section
3. Drag Debug on to the flow from the output section
4. Double-click on the inject node
5. Select the payload as a string
6. Enter a sample input to be sent to the assistant service and click on done



1. For creating a web application

UI we need "dashboard" nodes which should be installed manually.

2. 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.

The screenshot shows the Node-RED web interface in a browser. The top navigation bar includes tabs for 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom', 'Application Details - IBM Cloud', and 'Node-RED : node-red-ml-project'. The main workspace displays a flow named 'Flow 1' with three nodes: 'Hello', 'Input Parsing', and 'form'. The 'Input Parsing' node is selected, and its configuration panel is open. The panel shows the node's name as 'Input Parsing' and the function code as:

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

The right sidebar contains the 'info' panel, which displays the node's information, including its ID, name, type, and a description. The description states: 'A JavaScript function block to run against the messages being received by the node. The messages are passed in as a JavaScript object called msg. By convention it will have a msg.payload property containing the body of the message. The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.' The bottom status bar shows the system time as 10:43 PM on 5/20/2020.

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. FLOW CHART

At first go to manage palette and install dashboard. Now, Create the flow with the help following node:

Inject

Assistant

Debug

Function

Ui_Form

Ui_Text

The screenshot displays the Node-RED web interface in a browser. The top navigation bar includes tabs for 'Student Dashboard', 'IISPS_INT_855_Intelligent Custom', 'Application Details - IBM Cloud', and 'Node-RED : node-red-ml-project'. The address bar shows the URL 'node-red-ml-project.mybluemix.net/red/#flow/da32c7be.003768'. The main workspace, titled 'Flow 1', contains a chatbot flow with the following components: an 'inject' node with the text 'Hello', an 'assistant' node, a function node labeled 'Parsing', a 'msg.payload' node, a 'BOT' node with the text 'abc', an 'Input Parsing' function node, a 'form' node, and a 'You' node with the text 'abc'. The left sidebar shows a 'filter nodes' search bar and two categories of nodes: 'common' (inject, debug, complete, catch, status, link in, link out, comment) and 'function' (function, switch, change). The right sidebar displays the 'info' panel for the selected 'Input Parsing' node, showing its ID, name, type, and a detailed description of its functionality as a JavaScript function block.

Node-RED

Flow 1

inject

debug

complete

catch

status

link in

link out

comment

function

function

switch

change

info

Information

Node	"98432ace.295d08"
Name	Input Parsing
Type	function

Description

Node Help

A JavaScript function block to run against the messages being received by the node.

The messages are passed in as a JavaScript object called `msg`.

By convention it will have a `msg.payload` property containing the body of the message.

The function is expected to return a message object (or multiple message objects), but can choose to return nothing in order to halt a flow.

Details

See the [online documentation](#) for more

6.RESULTS

Finally our Node-RED dash board integrates all the components and displayed in the Dashboard UI by typing <https://node-red-qituc.eu-gb.mybluemix.net/ui/> in browser

Student Dashboard x IISPS_INT_855_Intelligent Cust... x Application Details - IBM Cloud x Node-RED : node-red-ml-proje x Node-RED Dashboard x

node-red-ml-project.mybluemix.net/ui/#/!/?socketid=tyv03agAW5wL8m3qAAAA

PAGE

CHATBOT
Enter your Input
hello
SUBMIT **CANCEL**

OUTPUT
You
BOT
hello
Hello. Good evening

Type here to search

ENG IN 10:47 PM 5/20/2020

Student Dashboard x IISPS_INT_855_Intelligent Cust... x Application Details - IBM Cloud x Node-RED : node-red-ml-proje x Node-RED Dashboard x

node-red-ml-project.mybluemix.net/ui/#/!/?socketid=tyv03agAW5wL8m3qAAAA

PAGE

CHATBOT
Enter your Input
How to turn on heater?
SUBMIT **CANCEL**

OUTPUT
You
BOT
How to turn on heater?
"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."

Type here to search

ENG IN 10:49 PM 5/20/2020

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.
can ask queries from home.

Disadvantages:

1. Some times chat bot can mislead customers
2. Giving same answer for different sentiments.
3. 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.
chat bot is the easy way of communication
It can be used for various applications
it is used to know about the product
it can also be used in knowing basic information

9.CONCLUSION

By doing the above procedure and all we successfully created Intelligent help desk smart chat bot using Watson assistant, Watson discovery, Node-RED and cloud-functions.
The conclusion for this is that it is the easy way of communication where we can resolve the

customer or the consumer queries without the help of the person we can do it virtually. So customer can access 24 hours for the queries what he have and get resolved his problem simply by sitting in his place and asking the bot some questions. If the bot does not understand it just give you the number of the customer care where you can contact them for future queries.

10.FUTURE SCOPE

We can include Watson studio text to speech and speech to text services to access the chat bot hands free. This is one of the future scope of this project.

11. BIBILOGRAPHY

we used some IBM guidelines etc to resolve our problems.

APPENDIX

Source Code

CloudFunction(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 thisaction

*

@paramCloudFunctionsactionsacceptasingleparameter,whichmustbeaJSONobject.

*

@return The output of this action, which must be a JSONObject.

*

*/

```
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": "d0d85ff2.0f31e",  
    "type": "tab",  
    "label": "Flow 1",  
    "disabled": false,  
    "info": ""
```

```
},
{
  "id": "81a35420.f26278",
  "type": "ui_form",
  "z": "d0d85ff2.0f31e",
  "name": "form",
  "label": "",
  "group": "eee5eb1.9de6f18",
  "order": 0,
  "width": "6",
  "height": "6",
  "options": [
    {
      "label": "enter your queries",
      "value": "text",
      "type": "text",
      "required": true,
      "rows": null
    }
  ],
  "formValue": {
    "text": ""
  },
  "payload": "",
  "submit": "submit",
  "cancel": "cancel",
  "topic": "",
  "x": 70,
  "y": 400,
  "wires": [
    [
      "2fc109df.9935e6"
    ]
  ]
},
```

```
{
  "id": "e4411f54.83b62",
  "type": "ui_text",
  "z": "d0d85ff2.0f31e",
  "group": "eee5eb1.9de6f18",
  "order": 1,
  "width": 0,
  "height": 0,
  "name": "",
  "label": "you",
  "format": "{{msg.payload}}",
  "layout": "row-spread",
  "x": 310,
  "y": 400,
  "wires": []
},
```

```
{
  "id": "8ea21b5b.9dfc78",
  "type": "inject",
  "z": "d0d85ff2.0f31e",
  "name": "",
  "topic": "",
  "payload": "hello",
  "payloadType": "str",
  "repeat": "",
  "crontab": "",
  "once": false,
  "onceDelay": 0.1,
  "x": 90,
  "y": 120,
  "wires": [
    [
      "3464ad27.b91b72"
    ]
  ]
}
```

```
},
{
  "id": "2fc109df.9935e6",
  "type": "function",
  "z": "d0d85ff2.0f31e",
  "name": "input parse",
  "func": "msg.payload=msg.payload.text;\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 190,
  "y": 300,
  "wires": [
    [
      "3464ad27.b91b72",
      "e4411f54.83b62"
    ]
  ]
},
{
  "id": "971684ee.01b3d8",
  "type": "function",
  "z": "d0d85ff2.0f31e",
  "name": "parsing",
  "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 560,
  "y": 200,
  "wires": [
    [
      "ae891913.e5e228",
      "de71deca.ca4b1"
    ]
  ]
},
```

```
{
  "id": "ae891913.e5e228",
  "type": "ui_text",
  "z": "d0d85ff2.0f31e",
  "group": "eee5eb1.9de6f18",
  "order": 0,
  "width": "6",
  "height": "9",
  "name": "",
  "label": "bot",
  "format": "{{msg.payload}}",
  "layout": "row-left",
  "x": 730,
  "y": 320,
  "wires": []
},
{
  "id": "de71deca.ca4b1",
  "type": "debug",
  "z": "d0d85ff2.0f31e",
  "name": "",
  "active": true,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "payload",
  "targetType": "msg",
  "x": 730,
  "y": 80,
  "wires": []
},
{
  "id": "3464ad27.b91b72",
  "type": "watson-conversation-v1",
  "z": "d0d85ff2.0f31e",
```

```
"name": "customer service bot",
"workspaceid": "2b5beca9-21e4-4391-b502-309530f12a63",
"multiuser": false,
"context": true,
"empty-payload": false,
"service-endpoint": "",
"timeout": "",
"optout-learning": false,
"x": 340,
"y": 220,
"wires": [
  [
    "971684ee.01b3d8"
  ]
],
},
{
  "id": "eee5eb1.9de6f18",
  "type": "ui_group",
  "z": "",
  "name": "form",
  "tab": "59bfe54e.277f4c",
  "order": 2,
  "disp": true,
  "width": "6",
  "collapse": false
},
{
  "id": "59bfe54e.277f4c",
  "type": "ui_tab",
  "z": "",
  "name": "Home",
  "icon": "dashboard",
  "disabled": false,
  "hidden": false
}
```

```
}  
]
```

```
[  
  {  
    "id": "d0d85ff2.0f31e",  
    "type": "tab",  
    "label": "Flow 1",  
    "disabled": false,  
    "info": ""  
  },  
  {  
    "id": "eee5eb1.9de6f18",  
    "type": "ui_group",  
    "z": "",  
    "name": "form",  
    "tab": "59bfe54e.277f4c",  
    "order": 2,  
    "disp": true,  
    "width": "6",  
    "collapse": false  
  },  
  {  
    "id": "59bfe54e.277f4c",  
    "type": "ui_tab",  
    "z": "",  
    "name": "Home",  
    "icon": "dashboard",  
    "disabled": false,  
    "hidden": false  
  },  
  {  
    "id": "22f35ab6.d3ff06",  
    "type": "ui_group",
```



```
"z": "",
"name": "Default",
"tab": "59bfe54e.277f4c",
"order": 1,
"disp": true,
"width": "6",
"collapse": false
},
{
  "id": "3b2ea5a6.df5fca",
  "type": "ui_base",
  "theme": {
    "name": "theme-light",
    "lightTheme": {
      "default": "#0094CE",
      "baseColor": "#0094CE",
      "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
      "edited": true,
      "reset": false
    },
    "darkTheme": {
      "default": "#097479",
      "baseColor": "#097479",
      "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif",
      "edited": false
    },
    "customTheme": {
      "name": "Untitled Theme 1",
      "default": "#4B7930",
      "baseColor": "#4B7930",
      "baseFont": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
    },
  },
}
```

```
"themeState": {  
  "base-color": {  
    "default": "#0094CE",  
    "value": "#0094CE",  
    "edited": false  
  },  
  "page-titlebar-backgroundColor": {  
    "value": "#0094CE",  
    "edited": false  
  },  
  "page-backgroundColor": {  
    "value": "#fafafa",  
    "edited": false  
  },  
  "page-sidebar-backgroundColor": {  
    "value": "#ffffff",  
    "edited": false  
  },  
  "group-textColor": {  
    "value": "#1bbfff",  
    "edited": false  
  },  
  "group-borderColor": {  
    "value": "#ffffff",  
    "edited": false  
  },  
  "group-backgroundColor": {  
    "value": "#ffffff",  
    "edited": false  
  },  
  "widget-textColor": {  
    "value": "#111111",  
    "edited": false  
  },  
  "widget-backgroundColor": {
```

```
    "value": "#0094ce",
    "edited": false
  },
  "widget-borderColor": {
    "value": "#ffffff",
    "edited": false
  },
  "base-font": {
    "value": "-apple-system,BlinkMacSystemFont,Segoe
UI,Roboto,Oxygen-Sans,Ubuntu,Cantarell,Helvetica Neue,sans-serif"
  }
},
"angularTheme": {
  "primary": "indigo",
  "accents": "blue",
  "warn": "red",
  "background": "grey"
}
},
"site": {
  "name": "Node-RED Dashboard",
  "hideToolbar": "false",
  "allowSwipe": "false",
  "lockMenu": "false",
  "allowTempTheme": "true",
  "dateFormat": "DD/MM/YYYY",
  "sizes": {
    "sx": 48,
    "sy": 48,
    "gx": 6,
    "gy": 6,
    "cx": 6,
    "cy": 6,
    "px": 0,
    "py": 0
  }
}
```

```
    }  
  }  
},  
{  
  "id": "81a35420.f26278",  
  "type": "ui_form",  
  "z": "d0d85ff2.0f31e",  
  "name": "form",  
  "label": "",  
  "group": "eee5eb1.9de6f18",  
  "order": 0,  
  "width": "6",  
  "height": "6",  
  "options": [  
    {  
      "label": "enter your queries",  
      "value": "text",  
      "type": "text",  
      "required": true,  
      "rows": null  
    }  
  ],  
  "formValue": {  
    "text": ""  
  },  
  "payload": "",  
  "submit": "submit",  
  "cancel": "cancel",  
  "topic": "",  
  "x": 70,  
  "y": 400,  
  "wires": [  
    [  
      "2fc109df.9935e6"  
    ]  
  ]  
}
```

```
]
},
{
  "id": "e4411f54.83b62",
  "type": "ui_text",
  "z": "d0d85ff2.0f31e",
  "group": "eee5eb1.9de6f18",
  "order": 1,
  "width": 0,
  "height": 0,
  "name": "",
  "label": "you",
  "format": "{{msg.payload}}",
  "layout": "row-spread",
  "x": 310,
  "y": 400,
  "wires": []
},
{
  "id": "8ea21b5b.9dfc78",
  "type": "inject",
  "z": "d0d85ff2.0f31e",
  "name": "",
  "topic": "",
  "payload": "hello",
  "payloadType": "str",
  "repeat": "",
  "crontab": "",
  "once": false,
  "onceDelay": 0.1,
  "x": 90,
  "y": 120,
  "wires": [
    [
      "3464ad27.b91b72"
```

```
    ]
  ]
},
{
  "id": "2fc109df.9935e6",
  "type": "function",
  "z": "d0d85ff2.0f31e",
  "name": "input parse",
  "func": "msg.payload=msg.payload.text;\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 190,
  "y": 300,
  "wires": [
    [
      "3464ad27.b91b72",
      "e4411f54.83b62"
    ]
  ]
},
{
  "id": "971684ee.01b3d8",
  "type": "function",
  "z": "d0d85ff2.0f31e",
  "name": "parsing",
  "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
  "outputs": 1,
  "noerr": 0,
  "x": 560,
  "y": 200,
  "wires": [
    [
      "ae891913.e5e228",
      "de71deca.ca4b1"
    ]
  ]
}
```

```
]
},
{
  "id": "ae891913.e5e228",
  "type": "ui_text",
  "z": "d0d85ff2.0f31e",
  "group": "eee5eb1.9de6f18",
  "order": 0,
  "width": "6",
  "height": "9",
  "name": "",
  "label": "bot",
  "format": "{{msg.payload}}",
  "layout": "row-left",
  "x": 730,
  "y": 320,
  "wires": []
},
{
  "id": "de71deca.ca4b1",
  "type": "debug",
  "z": "d0d85ff2.0f31e",
  "name": "",
  "active": true,
  "tosidebar": true,
  "console": false,
  "tostatus": false,
  "complete": "payload",
  "targetType": "msg",
  "x": 730,
  "y": 80,
  "wires": []
},
{
  "id": "3464ad27.b91b72",
```

```
"type": "watson-conversation-v1",  
"z": "d0d85ff2.0f31e",  
"name": "customer service bot",  
"workspaceid": "2b5beca9-21e4-4391-b502-309530f12a63",  
"multiuser": false,  
"context": true,  
"empty-payload": false,  
"service-endpoint": "",  
"timeout": "",  
"optout-learning": false,  
"x": 340,  
"y": 220,  
"wires": [  
  [  
    "971684ee.01b3d8"  
  ]  
]  
}  
]
```

References:

https://www.ibm.com/cloud/architecture/tutorials/cognitive_discovery

[h:https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started](https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started)

[h https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/](https://developer.ibm.com/recipes/tutorials/how-to-create-a-watson-chatbot-on-nodered/)

[h:https://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red](https://www.iotgyan.com/learning-resource/integration-of-watson-assistant-to-node-red)

[h:https://github.com/IBM/watson-discovery-sdu-with-assistant](https://github.com/IBM/watson-discovery-sdu-with-assistant)

h: <https://www.youtube.com/watch?v=Jpr3wVH3FVA>