

Index

1) Introduction

1.1 Overview

1.2 Purpose

2) Literature Survey

2.1 Existing Problem

2.2 Proposed Solution

3) Theoretical Analysis

3.1 Block Diagram

3.2 Hardware / Software Designing

4) Experimental Investigations

5) Flowchart

6) Result

7) Advantages & Disadvantages

8) Applications

9) Conclusion

10) Future Scope

11) Bibliography

Appendix

A. Source Code

INTRODUCTION

1.1) Overview:

In this Project we will use various Artificial Intelligence tools provided by IBM. To develop an interactive AI chatbot which reply to basic queries and retrieve data from pre-loaded data sets to reply complicated or more specific queries.

The tools are listed as follows:

- 1) **IBM Watson Discovery:** is an AI search technology that breaks open data silos and retrieves specific answers to your questions while analysing trends and relationships buried in enterprise data.
- 2) **IBM Watson Assistant:** is a conversation AI platform that helps you provide customers fast, straightforward and accurate answers to their questions, across any application, device or channel.
- 3) **IBM Cloud Function:** is a distributed compute service that executes application logic in response to requests from web or mobile apps. You can set up specific actions to occur based on HTTP-based API requests from web apps or mobile apps.
- 4) **Node-RED Application:** Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

1.2) Purpose:

The purpose of this Project is to create a self-automated Chatbot which can reply to simple queries and retrieve complex data from pre-loaded data sets to reply to complicated or more specific queries.

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.

LITERATURE SURVEY

2.1) Existing Problem:

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

The most common problem which is faced in the industry with Chatbot's are the have limited amount of replies for the numerous question which user might have, which increases the demand of involvement of human representative, that fails the cause for which a virtual Chatbot is created.

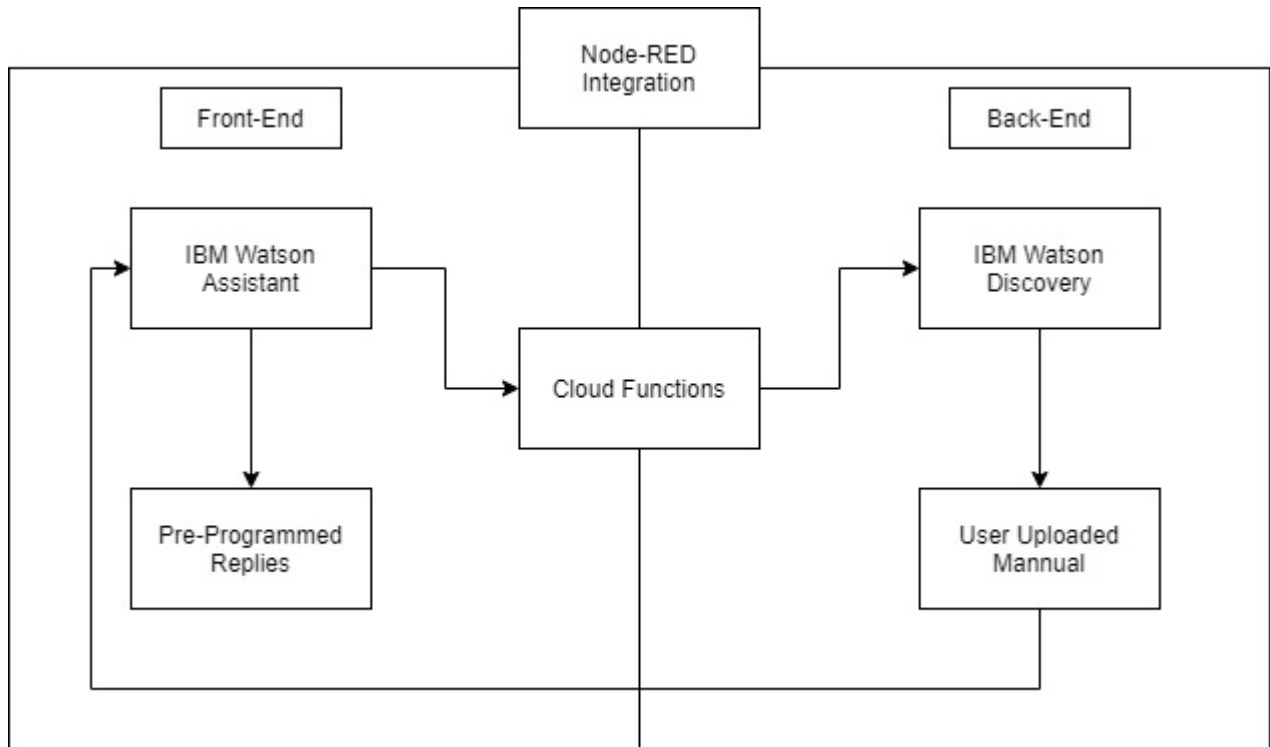
2.2) Proposed Solution:

Therefore, this project proposes the use of Smart Document Understanding feature which increases the horizon of replies by a substantial amount and also increases the quality of the reply.

Incorporating, Smart Document Understanding feature is successful is solving the problem of minimal data stored in assistant as the document's which are uploaded to the assistant provides ardent data to search qualitative data for complicated queries enquired by the user.

THEORITICAL ANALYSIS

3.1) Block Diagram:



3.2) Hardware / Software Designing

The Softwares are listed as follows:

- 5) **IBM Watson Discovery:** is an AI search technology that breaks open data silos and retrieves specific answers to your questions while analysing trends and relationships buried in enterprise data.
- 6) **IBM Watson Assistant:** is a conversation AI platform that helps you provide customers fast, straightforward and accurate answers to their questions, across any application, device or channel.
- 7) **IBM Cloud Function:** is a distributed compute service that executes application logic in response to requests from web or mobile apps. You can set up specific actions to occur based on HTTP-based API requests from web apps or mobile apps.
- 8) **Node-RED Application:** Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

EXPERIMENTAL INVESTIGATIONS

Experimental Procedure for creating the Chatbot:

1. Create IBM Cloud services

Create the following IBM cloud services:

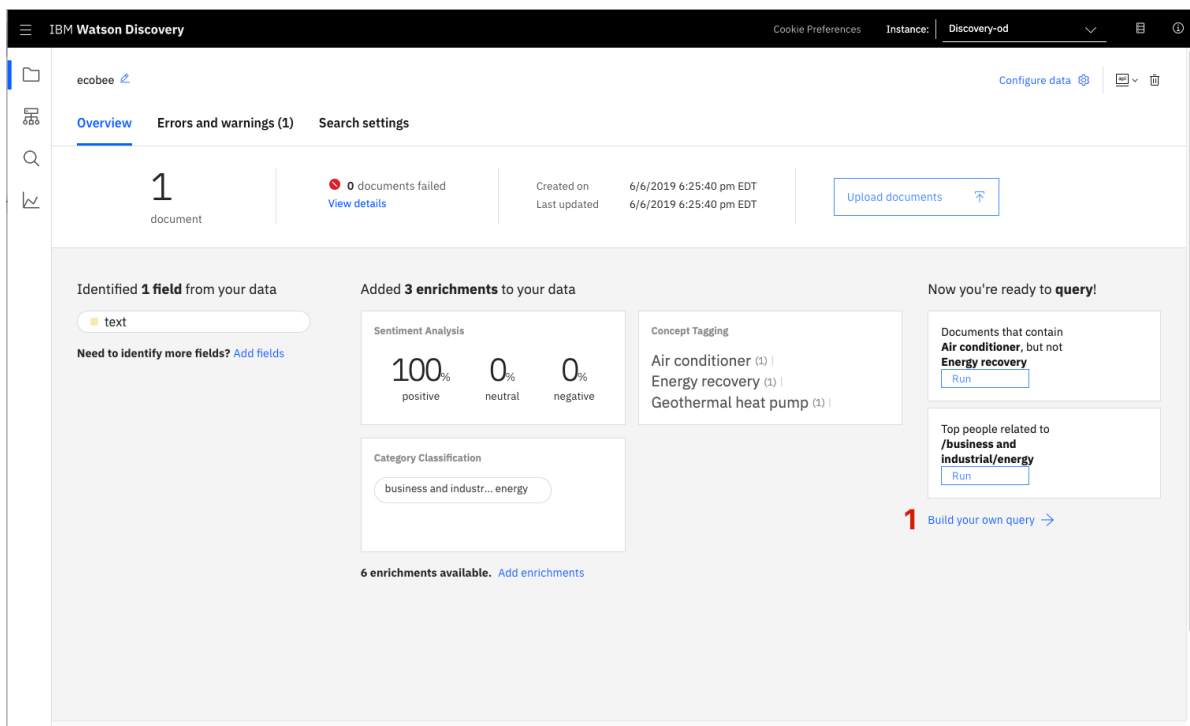
- 1) Watson Assistant
- 2) Watson Discovery

2. Configure Watson Discovery

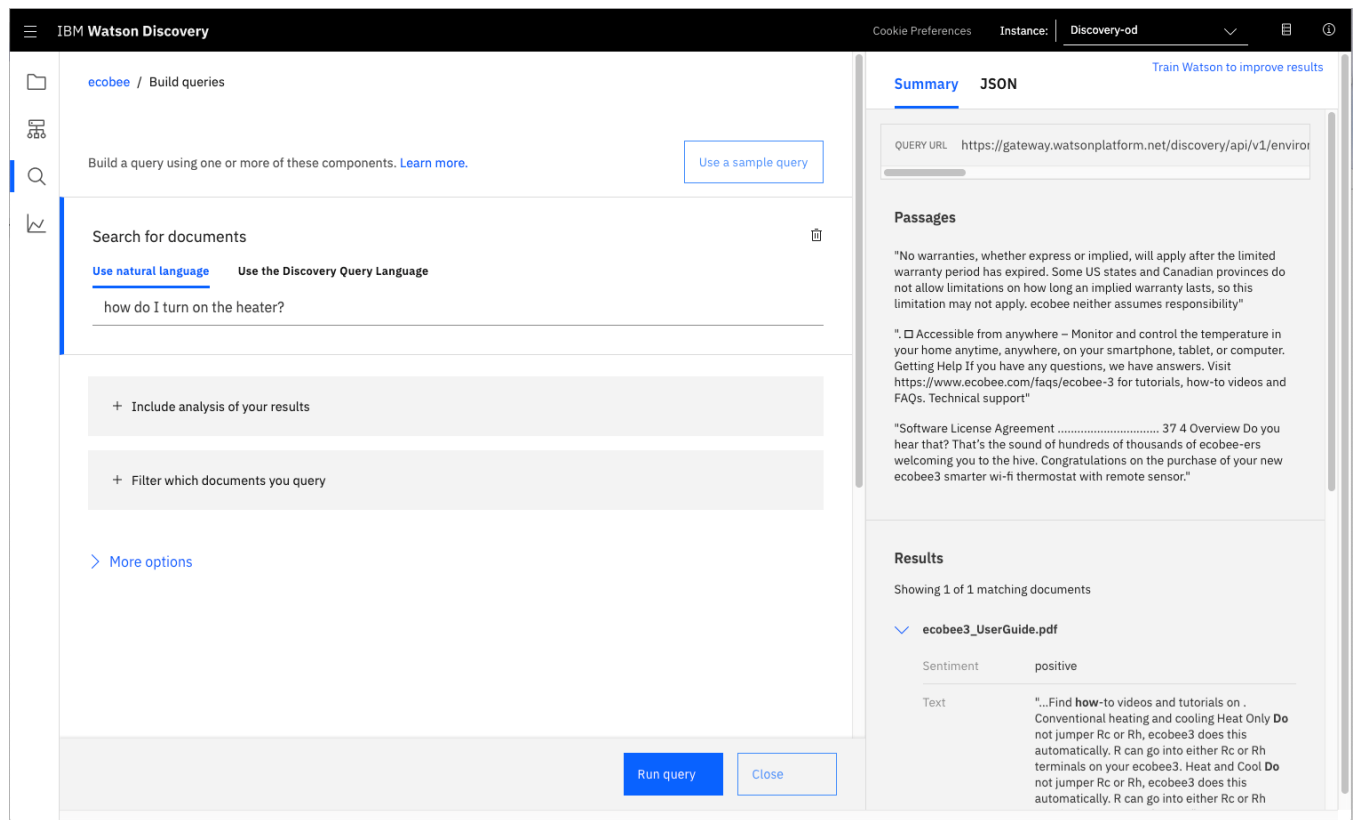
Import the document

Launch the Watson Discovery tool under resources tab 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 document which you want as the back-end manual.

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.



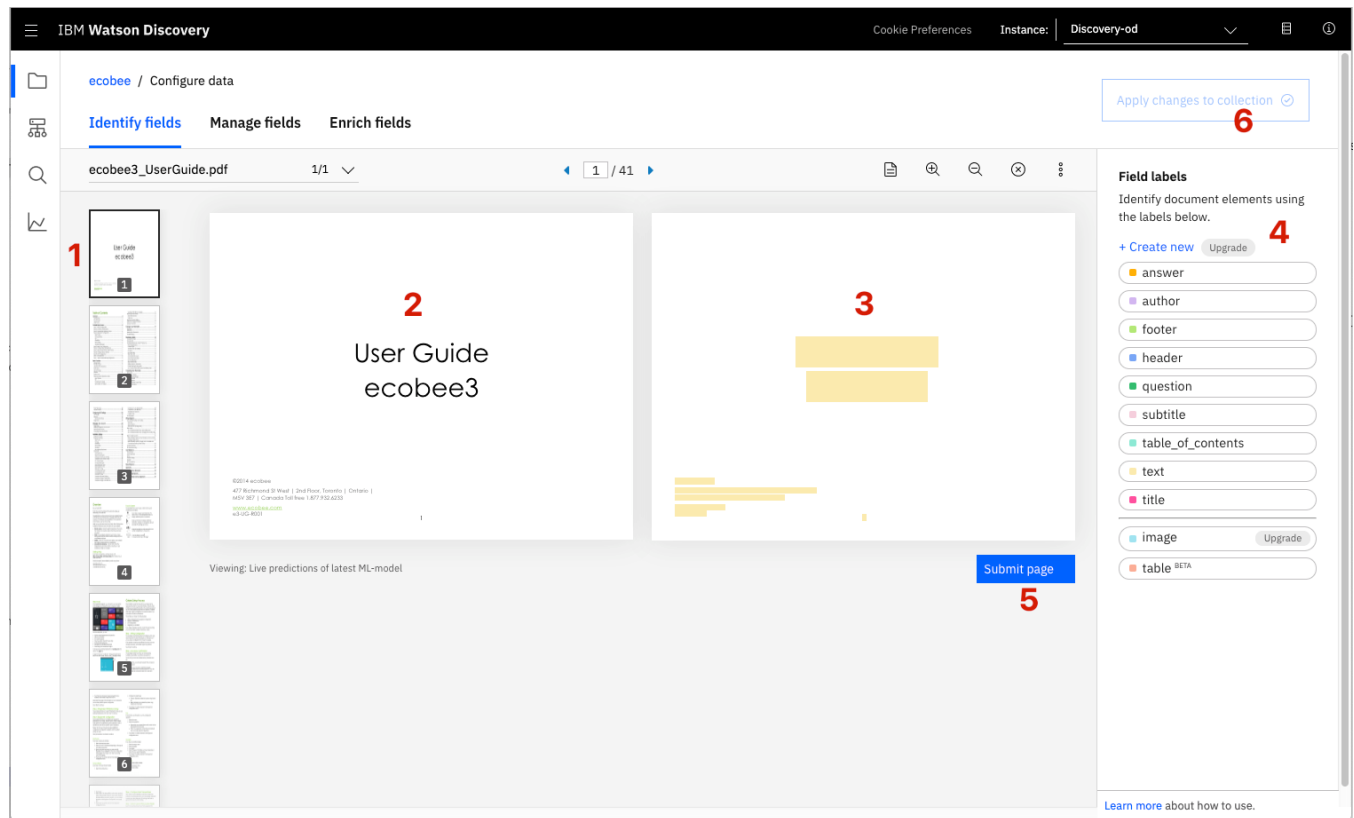
Enter queries related to the operation of the thermostat and view the results. As you will see, the results are not very useful, and in some cases, not even related to the question.

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 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.

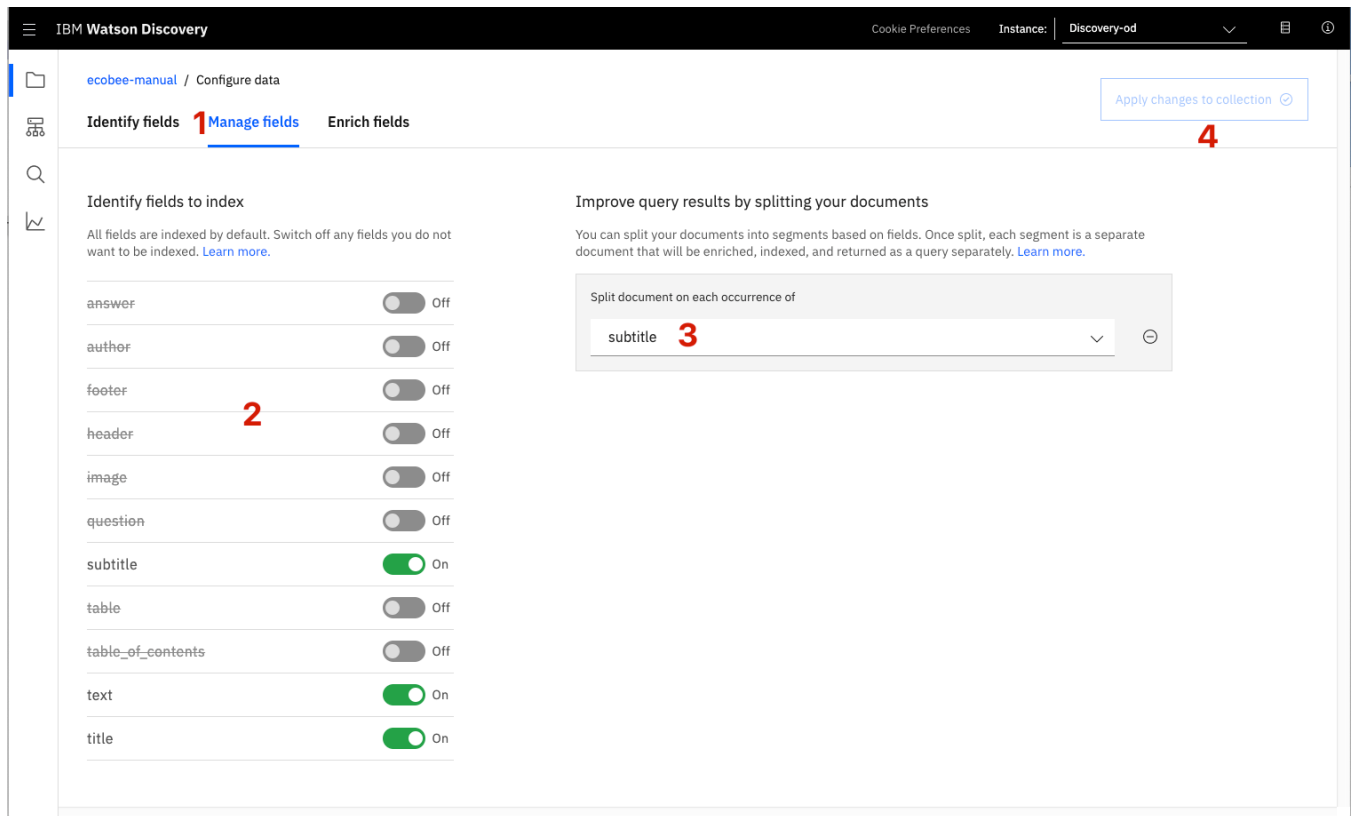
For this specific owner's manual, at a minimum, it is suggested to mark the following:

- 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:

IBM Watson Discovery

Cookie Preferences Instance: Discovery-od

ecobee-manual

Configure data

Overview Errors and warnings (130) Search settings

130 documents

0 documents failed View details

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

Upload documents

Identified 5 fields from your data

- footer
- subtitle
- table_of_contents
- text
- title

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

Added 4 enrichments to your data

Entity Extraction

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

Concept Tagging

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

5 enrichments available. [Add enrichments](#)

Sentiment Analysis

37% positive 26% neutral 36% negative

Category Classification

technology and com... operating systems

Now you're ready to query!

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

Run

Documents that contain **Heat**, but not **Internet**

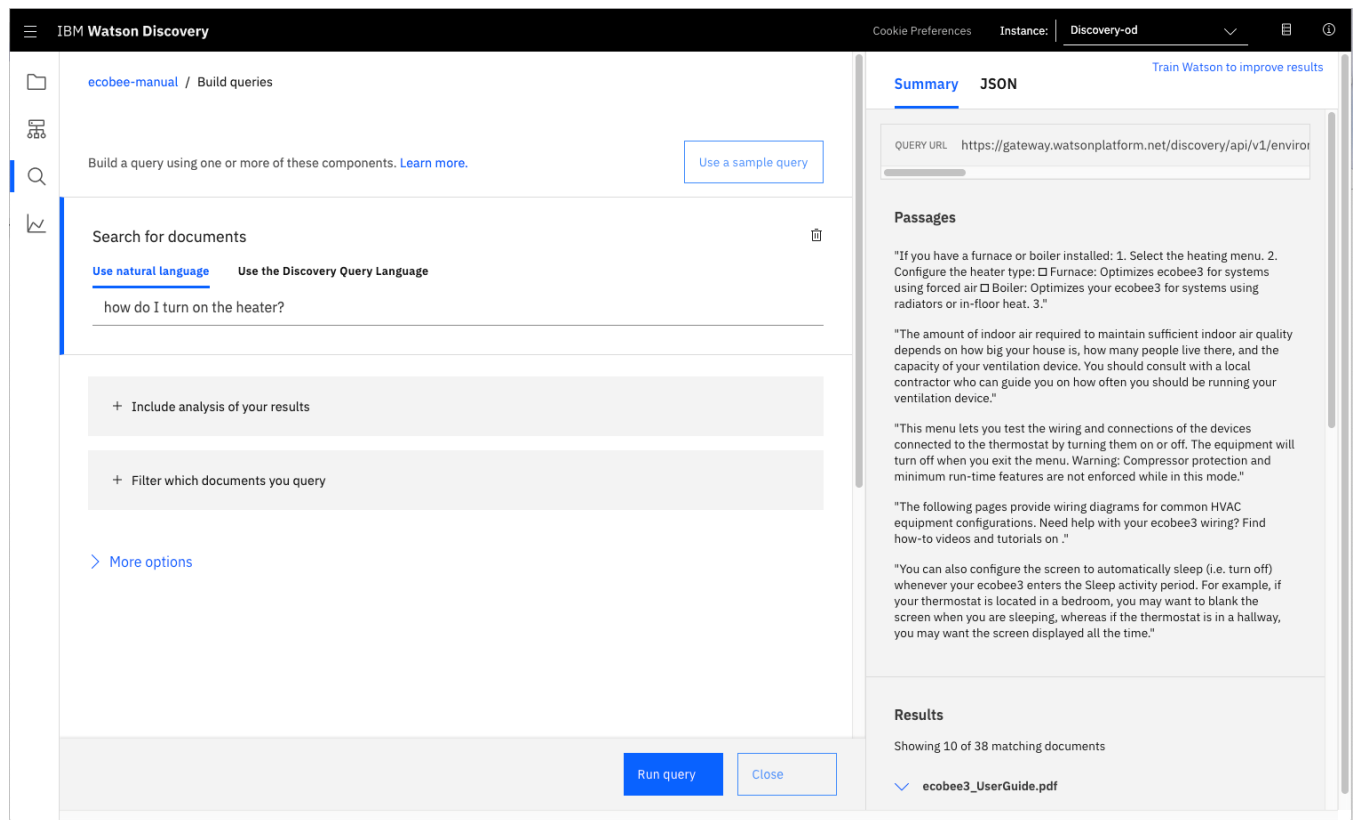
Run

Top entities with their average, min, max sentiment score

Run

[Build your own query](#)

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



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:

Cookie Preferences

Instance:

Discovery-od

Configure data

api

Collection ID

561d0cd7--8f2f-49361a56340c

Configuration ID

171876d1--a1f5-fe2aec037dca

Environment ID

b08a83ff--a02c-74f793d952b8

pm EDT

pm EDT

is

0% neutral

0% negative

Entities of type **Person** which have positive sentiment

Run

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

IBM Cloud

Search resources and offerings...

Resource list /

Discovery-od

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

Service credentials

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

Service credentials

New credential

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

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

```
{
  "apikey": "dryBf3aITnsy; [REDACTED] AHiau8bkoAfu10",
  "iam_apikey_description": "Auto generated apikey during resource-key operation for Instance - crn:v1:bluemix:public:discovery:us-south:a/bc1bd51c396536dc7d5f81d5a4e19533:acf2871-3bbd-4e04-a0f9-0daa59770852::",
  "iam_apikey_name": "auto-generated-apikey-f5f36cdd-d1d2-4a17-b41d-8ca5d1f1c7a6",
  "iam_role_crn": "crn:v1:bluemix:public:iam:::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/bc1bd51c396536dc7d5f81d5a4e19533::serviceid:ServiceId-016b8efa-a050-4708-a191-0b71f43cbddb",
  "url": "https://gateway.watsonplatform.net/discovery/api"
}
```

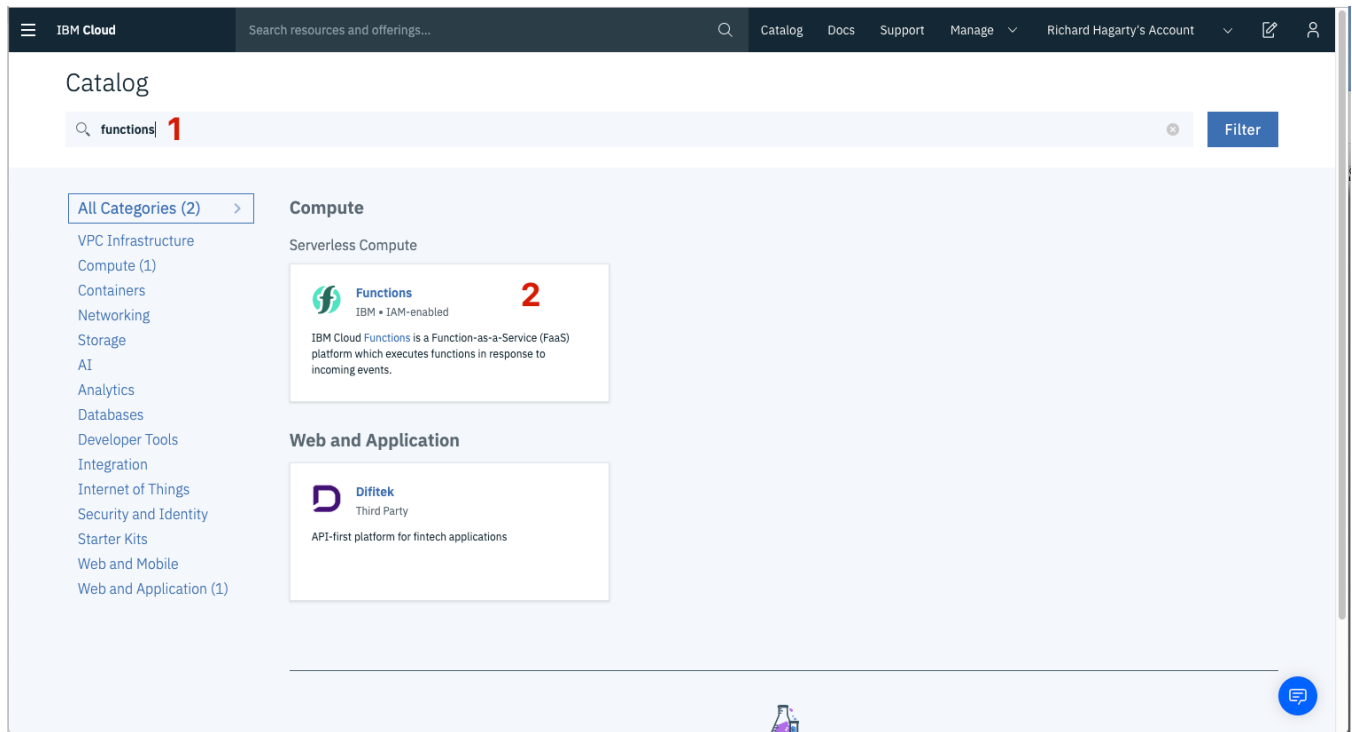
Click the View credentials [2] drop-down menu to view the IAM apikey [3] and URL endpoint [4] for your service.

3. Create IBM Cloud Functions action

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

Start the IBM Cloud Functions service by selecting Create Resource from the IBM Cloud dashboard.

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



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

From the Create panel, select the Create Action option.

On the Create Action panel, provide a unique Action Name [1], keep the default package [2], and select the Node.js 10 [3] runtime. Click the Create button [4] to create the action.

IBM Cloud

Search resources and offerings...

Catalog

Docs

Support

Manage

Richard Hagarty's Account

Functions

Getting Started

Actions

Triggers

APIs

Monitor

Logs

Namespace Settings

Create Action

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

[Learn more about Actions](#)

[Learn more about Packages](#)

Action Name

disco-action-2

Enclosing Package

(Default Package)

Create Package

Runtime

Node.js 10

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

Cancel

Previous

Create

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

disco-action

Web Action

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Code

Node.js 10

Change Input

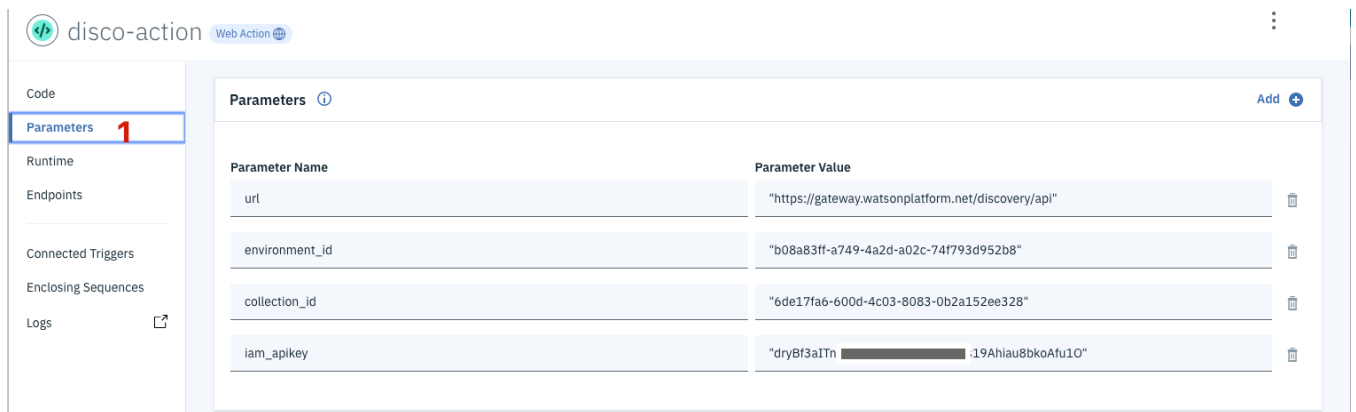
Invoke

```
1 - /**
2  *
3  * @param {object} params
4  * @param {string} params.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,
```

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

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

Select the Parameters tab [1]:



Add the following keys:

- url
- environment_id
- collection_id
- iam_apikey

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

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

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

The screenshot shows the IBM Cloud Functions console for the 'disco-action' function. The 'Code' panel displays the following Node.js code:

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

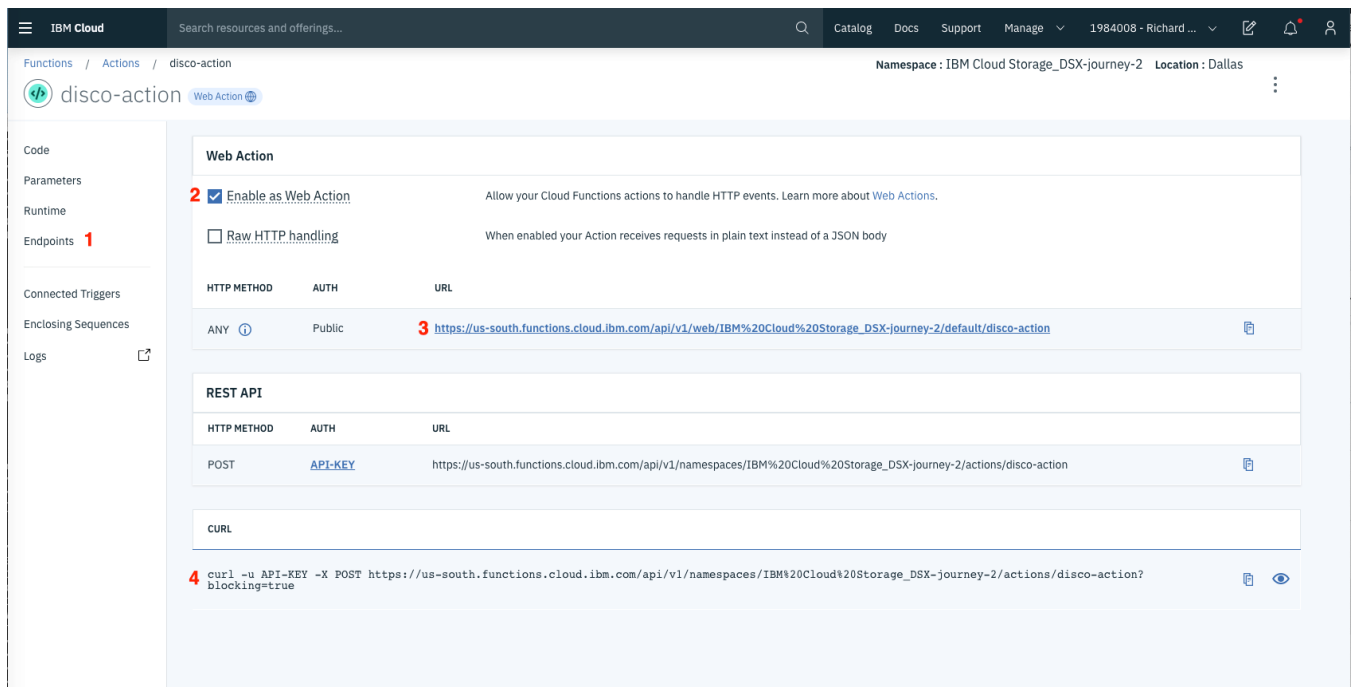
The 'Activations' panel shows the results of a successful invocation:

Activation ID: e1bfc0ff21544c85bfc0ff21549c85a1

Results:

```
{
  "matching_results": 14,
  "passages": [],
  "results": [
    {
      "enriched_text": {
        "categories": [
          {
            "label": "/technology and computing/operating systems",
            "score": 0.842265
          },
          {
            "label": "/technology and computing/hardware/computer",
            "score": 0.835879
          },
          {
            "label": "/technology and computing/hardware/computer peripherals/computer monitors",
            "score": 0.832254
          }
        ],
        "concepts": [
          {
            "dbpedia_resource": "http://dbpedia.org/resource/IPhone",
            "relevance": 0.917306,
            "text": "IPhone"
          },
          {
            "dbpedia_resource": "http://dbpedia.org/resource/Personal_digital_assistant",
            "relevance": 0.887088,
            "text": "Personal digital assistant"
          }
        ]
      }
    }
  ]
}
```

Next, go to the Endpoints panel [1]:



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

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

To verify you have entered the correct Discovery parameters, execute the provided curl command [4].

If it fails, re-check your parameter values.

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

4. Configure Watson Assistant

Launch the Watson Assistant tool under resources tab 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 centre 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 instructions from the manual you uploaded .

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 configuration page for a new intent named `#Product_Information`. The interface includes a header with a back arrow, the intent name, and a 'Try it' button. Below the header, there are fields for the intent name, description, and user examples. The 'Add user example' section is active, showing a list of three examples: 'How do I access the settings', 'How do I set the time', and 'How do I turn on the heater'. Each example has a checkbox and a timestamp of '2 hours ago'. There are also buttons for 'Add example' and 'Show recommendations'.

Intent name
Name your intent to match a customer's question or goal. For example, `#pay_bill` or `#open_account`.

`#Product_Information`

Description (optional)
User wants help using the thermostat

Add user example
Type a user example here

[Add example](#) [Show recommendations](#)

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

Create new dialog node

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

The screenshot shows the IBM Watson Assistant interface. At the top, there's a dark blue header with "IBM Watson Assistant". Below it, a breadcrumb "Skills /" is visible. The main title is "Customer Care Sample Skill copy" with a subtitle "Sample simple customer service skill to get you started." Below the title is a navigation bar with tabs: "Intents", "Entities", "Dialog" (marked with a red '1'), "Analytics", "Options", "Versions", and "Content Catalog". The "Dialog" tab is active, showing a list of dialog nodes. The nodes are: "Directions and location" (with a blue chevron icon), "Make an appointment", "Transfer to agent" (with a blue chevron icon), "Small Talk" (with a blue folder icon), and "anything_else". The "Small Talk" node is selected, and its dropdown menu is open, showing options: "Add node to folder", "Add node above", "Add node below" (marked with a red '3'), "Add folder", "Move", "Duplicate", "Jump to", and "Delete". A red '2' is placed next to the dropdown arrow of the "Small Talk" node.

IBM Watson Assistant

Skills /

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

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

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

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

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

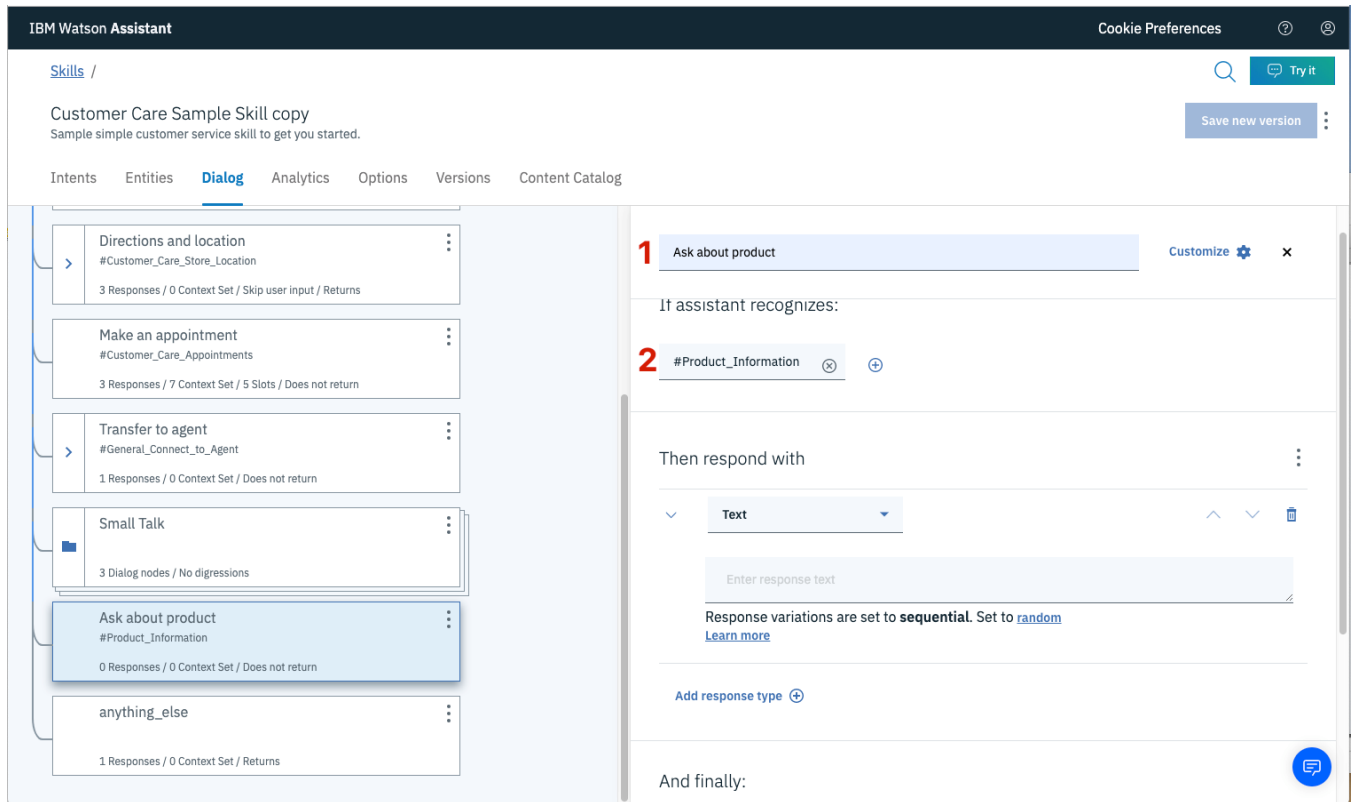
2 Small Talk
3 Dialog nodes / No digressions

anything_else
1 Responses / 0 Context Set / Returns

3

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

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]:

IBM Watson Assistant

Skills /

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

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

Webhooks

Autocorrection

System Entities

Webhooks

A webhook is a mechanism that allows your dialog skill to call an external API when specific dialog nodes are triggered. Specify the request URL for the external API you want to be able to invoke. You will then be able to access this URL from within the dialog editor.
[Learn more](#)

URL

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

Headers

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

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

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

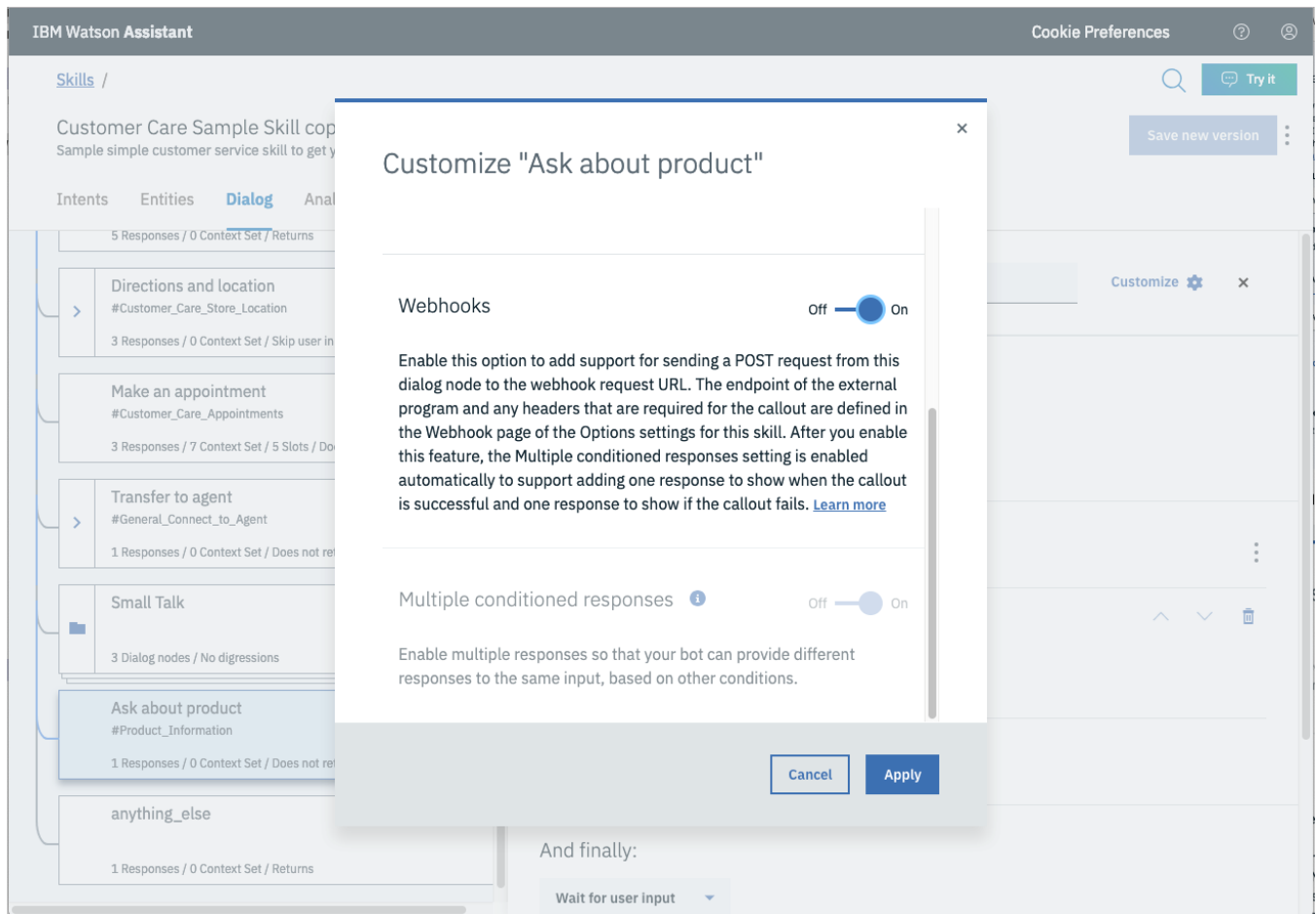
Next step

To trigger this webhook from an individual dialog node, enable the webhook from the Customize page in node details. [Go to dialog](#).

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

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

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



Click Apply.

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

IBM Watson Assistant

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

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

Ask about product Customize X

If assistant recognizes:

#Product_Information X +

Then callout to my webhook: [Learn more](#)

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

Add parameter +

Return variable

1 \$webhook_result_1

Then respond with

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

"<?input.text?>"

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

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

Return variable

\$webhook_result_1

Then respond with

	IF ASSISTANT RECOGNIZES	RESPOND WITH		
1	\$webhook_result_1	\$webhook_result_1		
2	anything_else	Try again later		

Add response

Test in Assistant Tooling

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

Enter some user input:

Try it out

Clear

Manage Context 3



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



hello

#General_Greetings



Hello. Good evening



how do I turn on the heater?

#Product_Information



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



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

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

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

Context variables ⓘ

\$Enter variable name

\$timezone

⊖

"America/Los_Angeles"

\$webhook_result_1

⊖

{"matching_results":9,"passages":[{"do

\$no_reservation

⊖

true

5. Creating User Interface and Webpage:

We will achieve this by Integration of Watson assistant in Node-RED.

Open the Node-RED from IBM catalogue. Open the app by clicking on visit App URL.



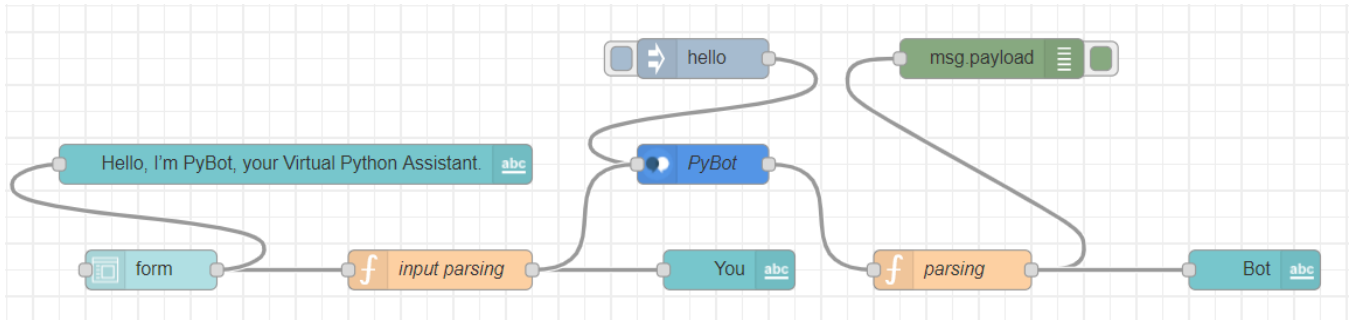
1) Import the flow from flow.json present in the Github repository.


The screenshot shows the "Import nodes" dialog in Node-RED. The "Clipboard" tab is selected, and the "Paste flow json or" button is active. A text area contains the following JSON flow definition:

```
[
  {
    "id": "ba712b04.c1fcb8",
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": ""
  },
  {
    "id": "cf6cf6f8.cb9908",
    "type": "inject",
    "z": "ba712b04.c1fcb8",
    "name": "",
    "topic": "",
    "payload": "hello",
    "payloadType": "str",
    "repeat": ""
  }
]
```

At the bottom, the "Import to" section has two buttons: "current flow" (selected) and "new flow". At the very bottom, there are "Cancel" and "Import" buttons.

The Flow will appear as following:



The Node-RED  node provides a very easy wrapper node to calls the Watson Assistant service and interact with the chat bot in Node-RED.

2) Enter the Watson Assistant credentials in the following window by double clicking the node.

Edit assistant node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖨

🔑 Name

PyBot

👤 Username

apikey

🔑 Password

.....

🔑 API Key

.....

🔑 Service Endpoint

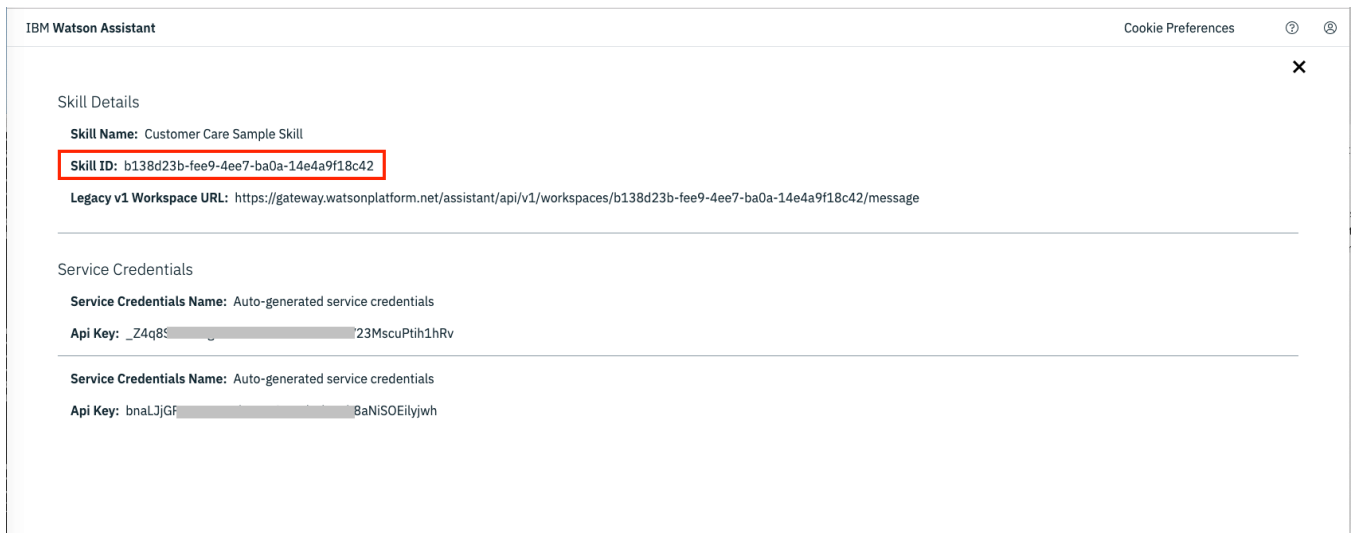
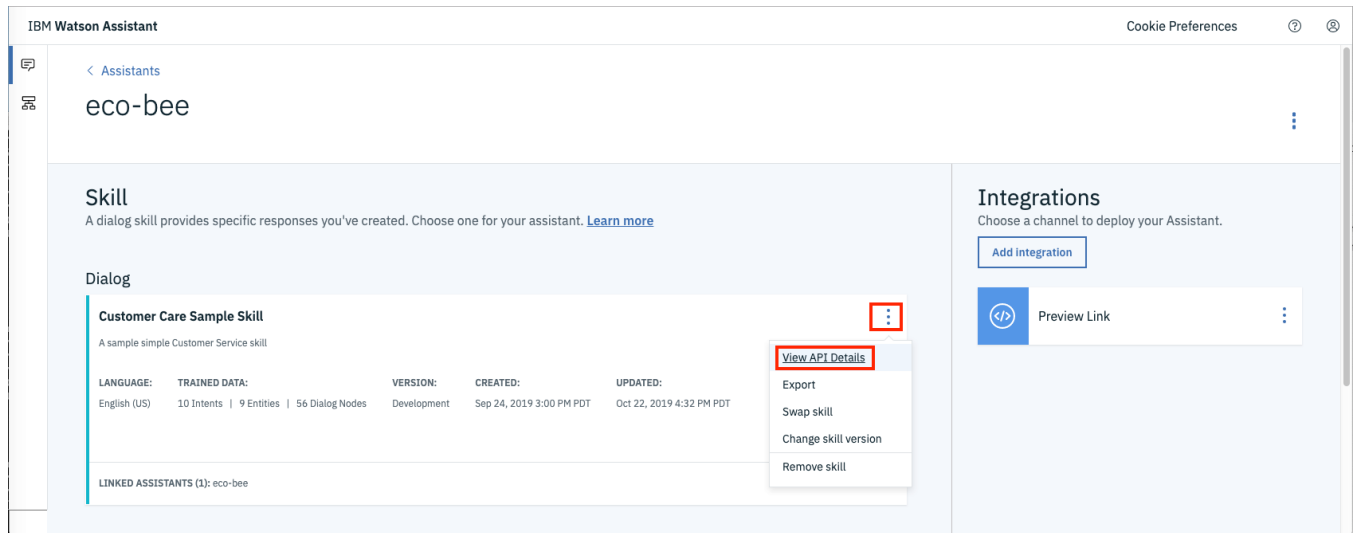
https://api.us-south.assistant.watson.cloud.ibm.co

🔑 Workspace ID

cd6377f2-0ac7-4310-ab99-3620258fdd43

Credentials can be found by clicking the Service Credentials tab, then the View Credentials option from the panel of your created Watson service.

An additional ASSISTANT_SKILL_ID value is required to access the Watson Assistant service. To get this value, select the Manage tab, then the Launch tool button from the panel of your Watson Assistance service. From the service instance panel, select your Assistant to display the assigned skills. For this code pattern, we used the dialog skill named Custom Skill Sample Skill that comes with the service:

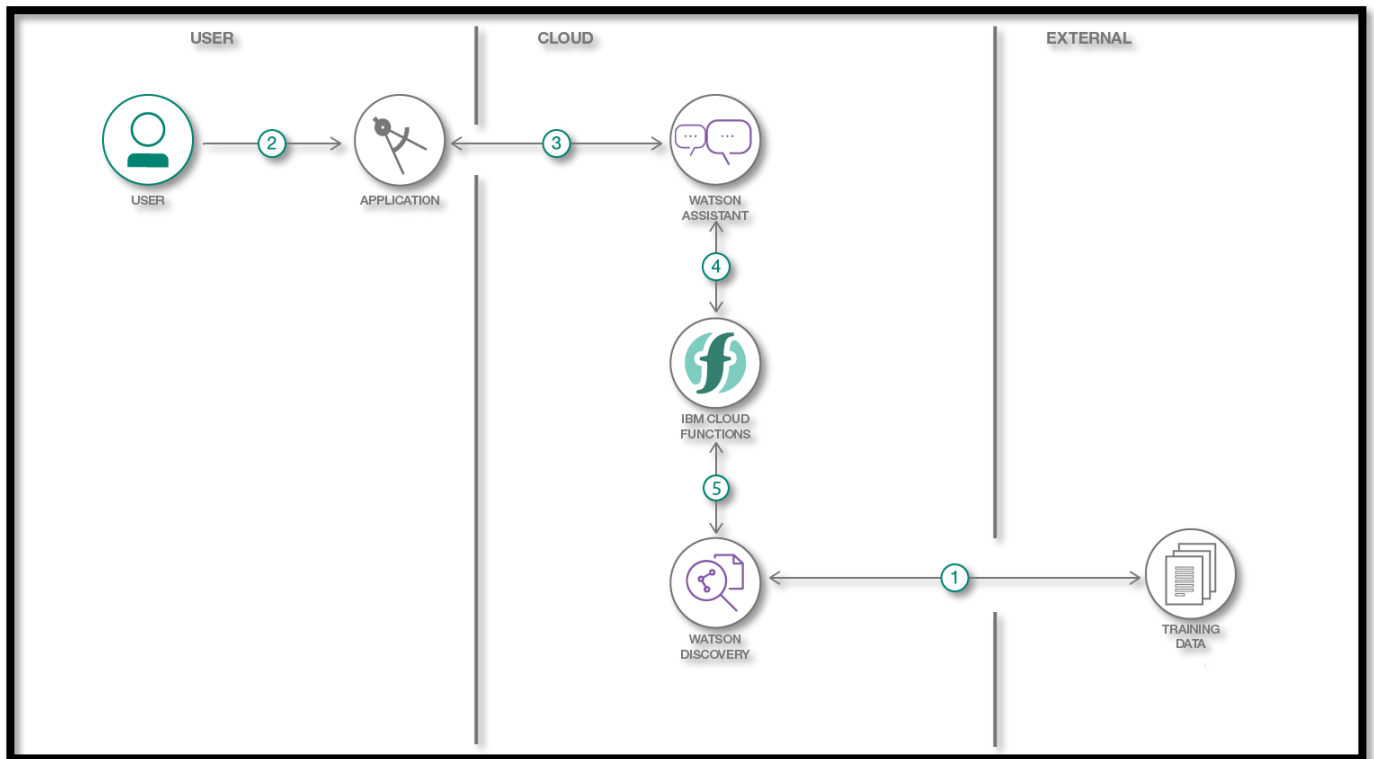


Click the option button (highlighted in the image above) to view the skill API Details. Here you will find the Skill ID value.

3) Deploy the Node-RED flow.

- 4) Copy the URL of your Current tab till .net. Go to the web browser paste the URL and add /ui to open the webpage application of the Chatbot.

FLOWCHART



RESULTS

After the Node-RED integrates all Watson services the final webpage output can be procured. To access this resultant webpage, you need to copy the web URL of the Node-RED till .net and edit further with /ui.

For example the Node-RED editor URL for my program is

<https://node-red-omklj.eu-gb.mybluemix.net/>

After editing

<https://node-red-omklj.eu-gb.mybluemix.net/ui>

PyBot

Hello, I'm PyBot, your Virtual Python Assistant.

Enter Your Question Here "

What is Python

SUBMIT

CANCEL

You **What is Python**

Bot [" Here's a very brief summary of what started it all, written by Guido van Rossum: I had extensive experience with implementing an interpreted language in the ABC group at CWI, and from working with this group I had learned a lot about language design. This is the origin of many Python features, including the use of indentation for statement grouping and the inclusion of very-high-level data types (although the details are all different in Python)."]

ADVANTAGES & DISADVANTAGES

Advantages	Disadvantages
The main advantage offered by chatbots from the point of view of customer service is automation.	The main drawback of the use of chatbots in this context is that the user can easily detect the presence of robotic answers.
It can help you establish your online presence outside of your working hours. If a client has a question in the middle of the night, and your business does not provide 24/7 customer support, a chatbot is a solution.	It may cost quite a lot to make a chatbot as specialists in the area are not that easy to find. This especially relates to applications based on AI.
Humans have a limit to the number of clients they can handle at once. However, with chatbots, there is no such constraint and they can handle as many queries as required at once.	It is one of the significant limitations of chatbots. These chatbots are programmed in a way that they only know what they are taught. They cannot understand the context of humans, and this is a massive gap that can even lead to an irate customer.

APPLICATIONS

- 1) Messaging apps
- 2) Company internal platforms
- 3) Customer Service
- 4) Healthcare
- 5) Politics
- 6) Toys

CONCLUSION

In this Project we have established the basis of an automated Chatbot based on AI which examines a pre uploaded document side by side for efficient and variety of results which increases the precision of the so procured, output provided by the Chatbot. Further, more we have concluded that how an amalgamation of programmes has made the result of the project possible, therefore stating that these software's have more horizon to reach than we suggest.

FUTURE SCOPE

It would be wrong or ignorant to say that chatbot is evolving and their evolution will become complete by the end of this project. Chatbot are being evolved and are more intelligent as well as human than ever. There is no denying to this fact. The successful adoption of chatbots by end users has led to the use of more and more bots in advanced artificial intelligence technologies and their usage by a custom software development company.

BIBLIOGRAPHY

APPENDIX

A. Source Code

1) Cloud Function (Integrate_Discovery.js)

```
/**
 *
 * @param {object} params
 * @param {string} params.iam_apikey
 * @param {string} params.url
 * @param {string} params.username
 * @param {string} params.password
 * @param {string} params.environment_id
 * @param {string} params.collection_id
 * @param {string} params.configuration_id
 * @param {string} params.input
 *
 * @return {object}
 *
 */

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

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

```

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

    let discovery;

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

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

```

2) Node-RED flow (flow.json)

```
[
  {
    "id": "ba712b04.c1fcb8",
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": ""
  },
  {
    "id": "cf6cf6f8.cb9908",
    "type": "inject",
    "z": "ba712b04.c1fcb8",
    "name": "",
    "topic": "",
    "payload": "hello",
    "payloadType": "str",
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "x": 230,
    "y": 80,
    "wires": [
      [
        "e712f9a.05b4208"
      ]
    ]
  },
  {
    "id": "d97aab5.a738e58",
    "type": "debug",
    "z": "ba712b04.c1fcb8",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 890,
    "y": 80,
  }
]
```

```

    "wires": []
  },
  {
    "id": "e712f9a.05b4208",
    "type": "watson-conversation-v1",
    "z": "ba712b04.c1fcb8",
    "name": "PyBot",
    "workspaceid": "cd6377f2-0ac7-4310-ab99-3620258fdd43",
    "multiuser": false,
    "context": false,
    "empty-payload": false,
    "service-endpoint": "https://api.us-south.assistant.watson.cloud.ibm.com",
    "timeout": "0.5 ",
    "optout-learning": false,
    "x": 490,
    "y": 200,
    "wires": [
      [
        "49fe6c94.e7d224"
      ]
    ]
  },
  {
    "id": "49fe6c94.e7d224",
    "type": "function",
    "z": "ba712b04.c1fcb8",
    "name": "parsing",
    "func": "msg.payload=msg.payload.output.text[0];\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 700,
    "y": 240,
    "wires": [
      [
        "d97aab5.a738e58",
        "7c30a70f.fde918"
      ]
    ]
  },
  {
    "id": "8ebee192.d443b",
    "type": "ui_form",
    "z": "ba712b04.c1fcb8",

```

```

    "name": "",
    "label": "",
    "group": "9d205537.7113a8",
    "order": 2,
    "width": 0,
    "height": 0,
    "options": [
      {
        "label": "Enter Your Question Here",
        "value": "text",
        "type": "text",
        "required": true,
        "rows": null
      }
    ],
    "formValue": {
      "text": ""
    },
    "payload": "",
    "submit": "submit",
    "cancel": "cancel",
    "topic": "",
    "x": 70,
    "y": 220,
    "wires": [
      [
        "ac05fad6.de22a8",
        "7db381d2.d5346"
      ]
    ]
  },
  {
    "id": "ac05fad6.de22a8",
    "type": "function",
    "z": "ba712b04.c1fcb8",
    "name": "input parsing",
    "func": "msg.payload=msg.payload.text;\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 290,
    "y": 260,
    "wires": [

```

```

        "57882ceb.0ec584",
        "e712f9a.05b4208"
    ]
]
},
{
    "id": "57882ceb.0ec584",
    "type": "ui_text",
    "z": "ba712b04.c1fcb8",
    "group": "9d205537.7113a8",
    "order": 4,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "You",
    "format": "{{msg.payload}}",
    "layout": "row-left",
    "x": 170,
    "y": 420,
    "wires": []
},
{
    "id": "7c30a70f.fde918",
    "type": "ui_text",
    "z": "ba712b04.c1fcb8",
    "group": "9d205537.7113a8",
    "order": 6,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "Bot",
    "format": "{{msg.payload}}",
    "layout": "row-left",
    "x": 770,
    "y": 400,
    "wires": []
},
{
    "id": "7db381d2.d5346",
    "type": "ui_text",
    "z": "ba712b04.c1fcb8",
    "group": "9d205537.7113a8",
    "order": 1,

```

```

        "width": 0,
        "height": 0,
        "name": "",
        "label": "Hello, I'm PyBot, your Virtual Python Assistant.",
        "format": "",
        "layout": "row-center",
        "x": 670,
        "y": 320,
        "wires": []
    },
    {
        "id": "9d205537.7113a8",
        "type": "ui_group",
        "z": "",
        "name": "PyBot",
        "tab": "98a11fd7.73f55",
        "order": 1,
        "disp": true,
        "width": "20",
        "collapse": false
    },
    {
        "id": "98a11fd7.73f55",
        "type": "ui_tab",
        "z": "",
        "name": "Virtual Python Services",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]

```