

# **PROJECT REPORT**

## Intelligent Customer Help Desk with Smart Document Understanding - SB19956

**Kick-off Date:** 5 May, 2020

**Name:** Agranya Pratap Singh

**Email-ID:** [agranyasingh5@gmail.com](mailto:agranyasingh5@gmail.com)

**Category:** Artificial Intelligence

### **DEVELOPMENT ENVIRONMENT**

**GitHub account:** <https://github.com/agranya99>

**IBM Academic Initiative Account:** agranyapratap.singh2017@vitstudent.ac.in

**IBM ID (IBM Cloud):** agranyapratap.singh2017@vitstudent.ac.in

## **CONTENTS**

1. INTRODUCTION
2. ABOUT THE PROJECT
3. SCOPE OF WORK
4. LITERATURE SURVEY  
EXISTING SYSTEM
5. PROPOSED SYSTEM
6. FLOWCHART OF THE MODEL
7. DESIGNING SERVICES USED
8. NODE-RED STARTER APPLICATION
9. IBM WATSON USE-CASES
10. WATSON DISCOVERY, ASSISTANT & IBM CLOUD SERVICES
11. RESULT: INTELLIGENT CUSTOMER HELP DESK (FINAL)
12. ADVANTAGES, DISADVANTAGES AND APPLICATION
13. CONCLUSION AND FUTURE SCOPE
  
- A-I. APPENDIX
- A-II. REFERENCES

# **INTRODUCTION**

Watson is an IBM supercomputer that combines artificial intelligence (AI) and sophisticated analytical software for optimal performance as a "question answering" machine. The supercomputer is named for IBM's founder, Thomas J. Watson. The Watson supercomputer processes at a rate of 80 teraflops (trillion floating point operations per second). To replicate (or surpass) a high-functioning human's ability to answer questions, Watson accesses 90 servers with a combined data store of over 200 million pages of information, which it processes against six million logic rules. The system and its data are self-contained in a space that could accommodate 10 refrigerators.

Applications for Watson's underlying cognitive computing technology are almost endless. Because the device can perform text mining and complex analytics on huge volumes of unstructured data, it can support a search engine or an expert system with capabilities far superior to any previously existing.

## **Overview**

We will be able to write an application that leverages multiple Watson AI services like Discovery, Assistant, Cloud Function and Node Red. By the end of the project we will learn best practices of combining Watson Services, and how they can build interactive retrieval system with discovery + assistant.

- Project Requirements: Python, IBM Cloud, IBM Watson
- Functional Requirements: IBM Cloud
- Technical Requirements: Python, Watson AI, ML
- Software Requirements: Watson Assistant, Watson Discovery
- Project Deliverables: Smartinternz Internship
- Project Duration: 19 days

## **ABOUT THE PROJECT**

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

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device's owner's manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owner's manual to help solve our 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 owner's manual is important and what is not. This will improve the answers returned from the queries.

## **SCOPE OF WORK**

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

## EXISTING SYSTEM

At first, Chatbot can look like a normal app. There is an application layer, a database and APIs to call external services. In a case of the chatbot, UI is replaced with chat interface. While Chatbots are easy to use for users, it adds complexity for the app to handle.

There is a general worry that the bot can't understand the intent of the customer. The bots are first trained with the actual data. Most companies that already have a chatbot must be having logs of conversations. Developers use that logs to analyze what customers are trying to ask and what does that mean. With a combination of Machine Learning models and tools built, developers match questions that customer asks and answers with the best suitable answer. For example: If a customer is asking "Where is my payment receipt?" and "I have not received a payment receipt", mean the same thing. Developers strength is in training the models so that the chatbot is able to connect both of those questions to correct intent and as an output produces the correct answer. If there is no extensive data available, different APIs data can be used to train the chatbot.

### HOW CHATBOTS ACTUALLY WORK?

The chatbots work by adopting 3 classification methods:

- **Pattern Matchers:**

Bots use pattern matching to classify the text and produce a suitable response for the customers. A standard structure of these patterns is "Artificial Intelligence Markup Language" (AIML).

A simple pattern matching example:

*The machine then gives and output:*

*Human: Do you know who Abraham Lincoln is?*

*Robot: Abraham Lincoln was the US President during American civil war.*

- **Algorithms:**

For each kind of question, a unique pattern must be available in the database to provide a suitable response. With lots of combination on patterns, it creates a hierarchical structure. We use algorithms to reduce the classifiers and generate the more manageable structure. Computer scientists call it a "Reductionist" approach- in order to give a simplified solution, it reduces the problem.

Multinational Naive Bayes is the classic algorithm for text classification and NLP. For an instance, let's assume a set of sentences are given which are belonging to a particular class. With new input sentence, each word is counted for its occurrence and is accounted for its commonality and each class is assigned a score. The highest scored class is the most likely to be associated with the input sentence.

For example Sample Training set

*class: greeting*

*"How you doing?"*

*"good morning"*

*"hi there"*

Few sample Input sentence classification:

*input: "Hello good morning"*

*term: "hello" (no matches)*

*Term: "good" (class: greeting)*

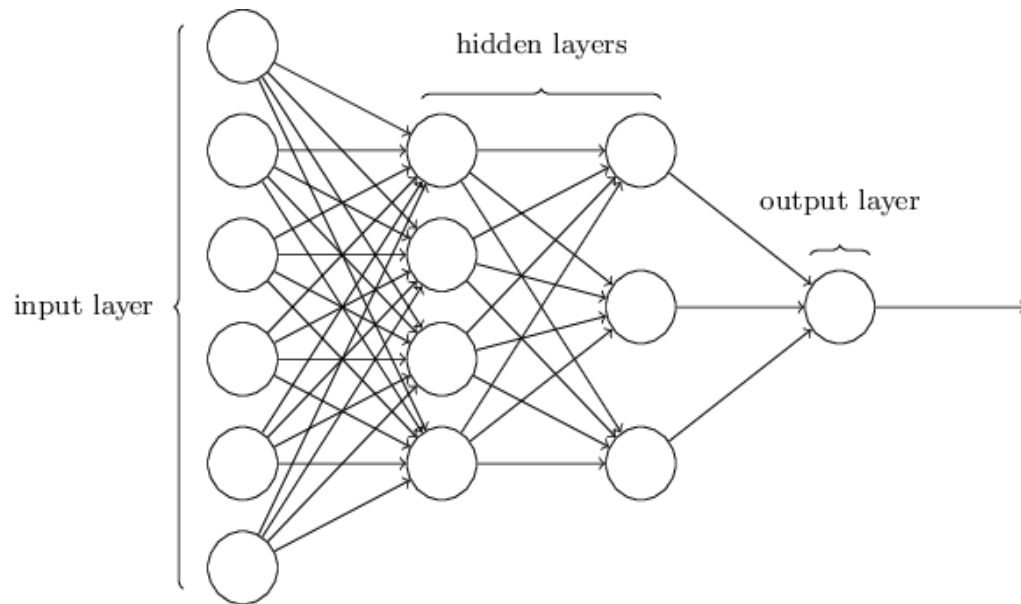
*term: "morning" (class: greeting)*

*classification: greeting (score=2)*

With the help of equation, word matches are found for given some sample sentences for each class. Classification score identifies the class with the highest term matches but it also has some limitations. The score signifies which intent is most likely to the sentence but does not guarantee it is the perfect match. Highest score only provides the relativity base.

- **Artificial Neural Networks:**

Neural Networks are a way of calculating the output from the input using weighted connections which are calculated from repeated iterations while training the data. Each step through the training data amends the weights resulting in the output with accuracy.



As discussed earlier here also, each sentence is broken down into different words and each word then is used as input for the neural networks. The weighted connections are then calculated by different iterations through the training data thousands of times. Each time improving the weights to making it accurate. The trained data of neural network is a comparable algorithm more and less code. When there is a comparably small sample, where the training sentences have 200 different words and 20 classes, then that would be a matrix of  $200 \times 20$ . But this matrix size increases by  $n$  times more gradually and can cause a huge number of errors. In this kind of situations, processing speed should be considerably high.

There are multiple variations in neural networks, algorithms as well as patterns matching code. Complexity may also increase in some of the variations. But the fundamental remains the same, and the important work is that of classification.

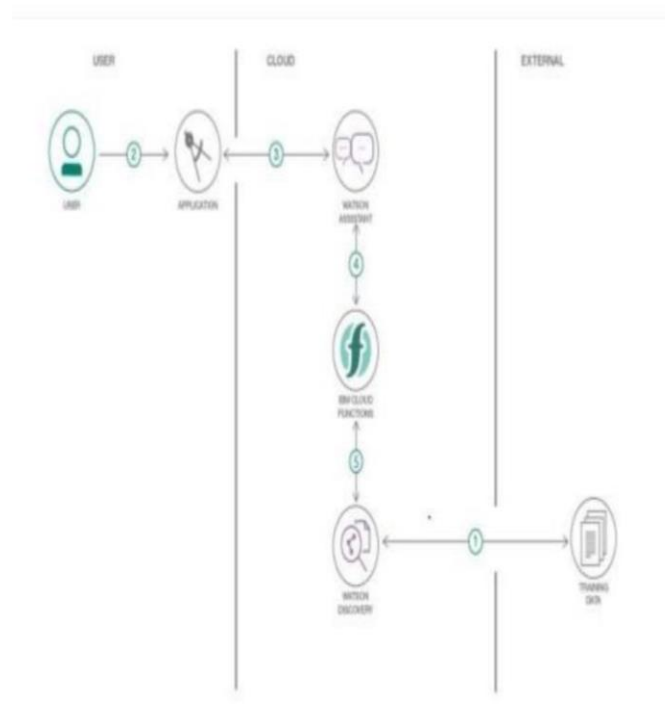
## PROPOSED SYSTEM

In this project, there will be another option. If the customer question is about the operation of a device, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the device's owner's manual. So now, instead of "Would you like to speak to a customer representative?" we can return relevant sections of the owner's manual to help solve our 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 owner's manual is important and what is not. This will improve the answers returned from the queries.

We are using Hukseflux Thermal sensors HTR01 user manual. At the end of the project Watson will be trained with the various user's questions and answer them specifically.

## FLOWCHART OF THE MODEL



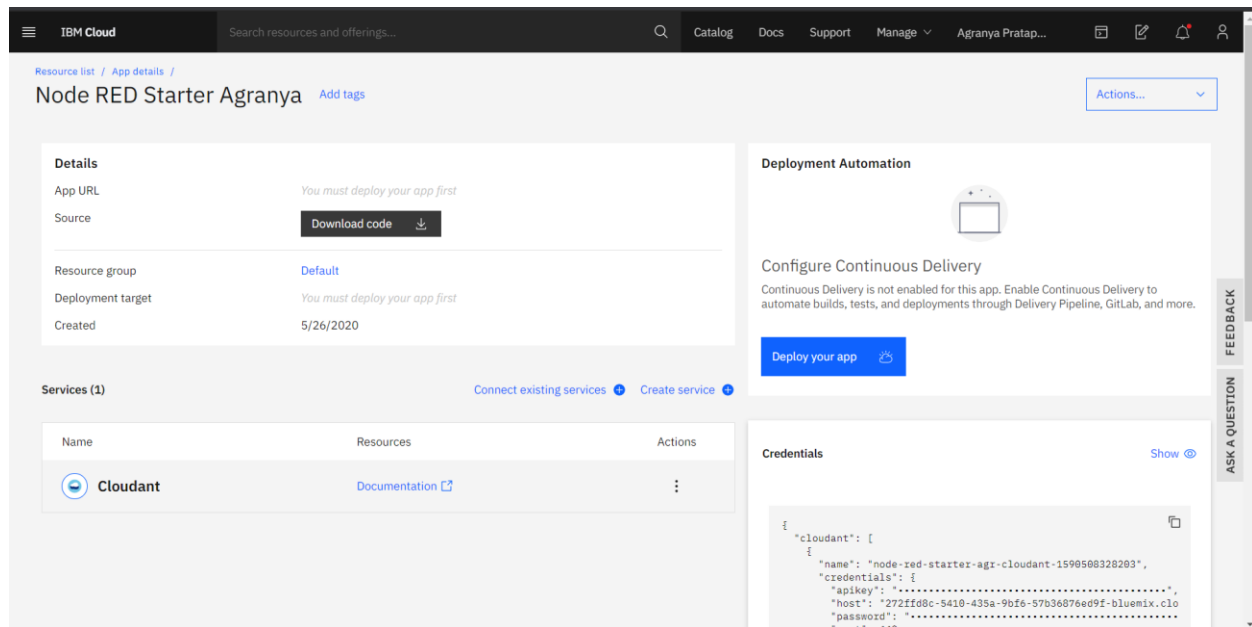
## DESIGNING SERVICES USED



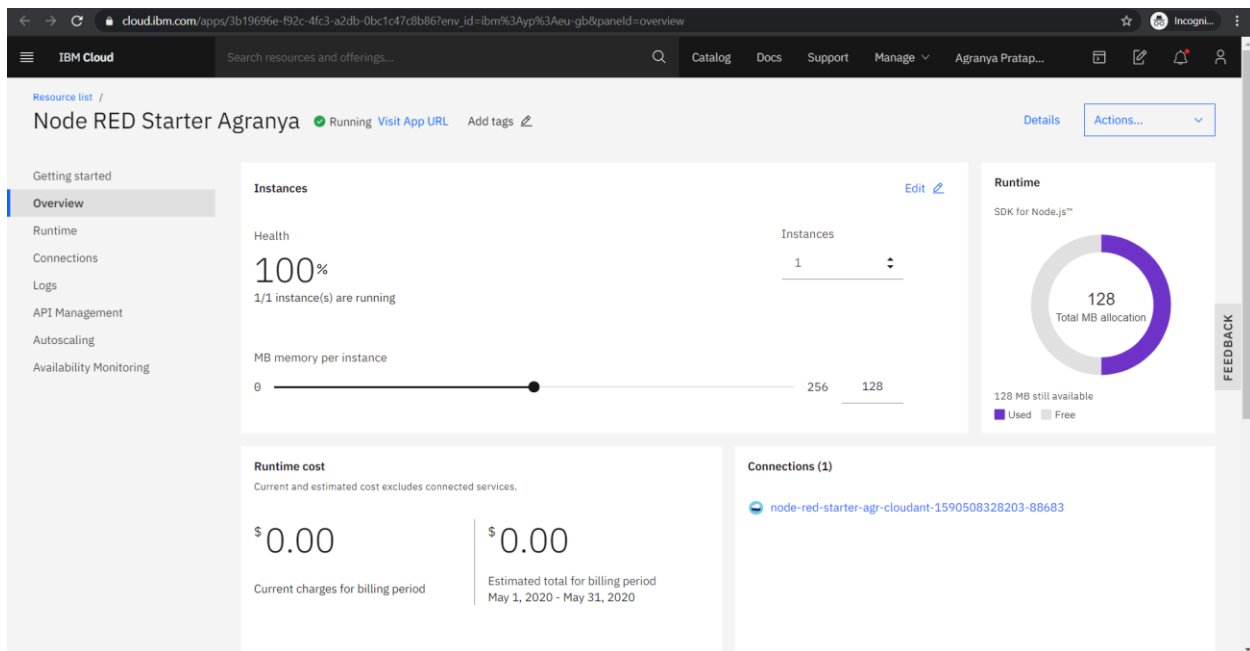
1. IBM Cloud
2. IBM Watson
3. Node-RED
4. Python
5. Watson Assistant

## CREATING NODE-RED STARTER APPLICATION IN IBM CLOUD

- 1: Login to IBM and go to the catalog
- 2: Search for node-red and select “Node-RED Starter “ Service
- 3: Enter the Unique name and click on create a button



- 5: We have to configure Node red for the first time. Click on next to Continue  
(- Deploy from the Delivery Pipeline -)



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

The screenshot shows the 'Secure your Node-RED editor' configuration screen. It has two main sections:

- Secure your editor so only authorised users can access it:** This section is selected. It contains fields for 'Username' (set to 'agranya') and 'Password' (masked with dots). A password strength indicator shows 'strong'. Below these fields is a checkbox labeled 'Allow anyone to view the editor, but not make any changes', which is checked.
- Not recommended: Allow anyone to access the editor and make changes:** This option is unselected.

At the bottom of the screen, there are 'Previous' and 'Next' buttons.

7: Click Next to continue

8: Click Finish

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

10: After this we have to make the node-red flow, and link everything. We will get a UI from the node.

Node red editor has various nodes with the respective functionality.

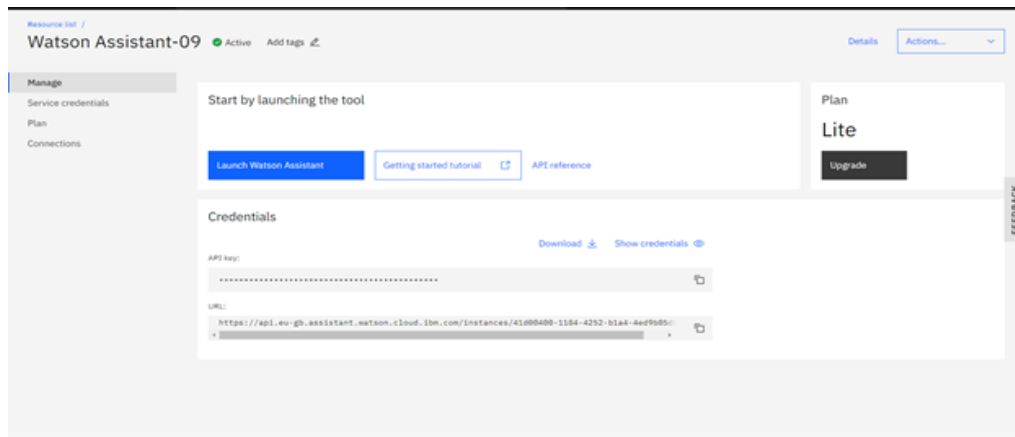
## EXPLORING IBM WATSON USE-CASES

1. AI for customer service: Build a full-service AI assistant that your customers and employees will actually want to use. Seamlessly automate tasks. Address customer requests across channels like digital and voice. Guide employees through internal processes. Best of all: allow your teams to focus on higher value work. Bring Watson Assistant to your data, no matter what cloud it lives on; embed it within any channel – voice or web – to ensure you are meeting your customers and employees where they are; and integrate it within any system, simplifying processes for agents and employees alike.  
95% accuracy rate. 99% increase in response time
2. AI for financial services: IBM RegTech. Accelerate insights, reduce infrastructure costs, improve efficiency for timely, risk-aware decisions with an AI-infused, integrated and cloud-native platform approach on IBM Cloud Pak for Data.
3. AI for enterprise research: Watson Discovery surfaces answers with accuracy and speed. Reduce time spent researching by 75%. AI search augments human research by quickly uncovering information inside your organization's complex knowledge silos and surfacing specific answers – not just the relevant document. With Watson Discovery, deploy AI enterprise search anywhere to reach data stored on-premises or in any private, public, hybrid, or multi-cloud environment. Empower your employees to accurately analyze millions of documents, using best-in-class text analytics – no matter where the content resides.
4. AI for contract governance: Optimize your contract review. Watson Compare & Comply uses AI to identify and understand relationships among headings, bullets, tables, and text – making complex text easily searchable, regardless of whether the exact terms are used or not.

# WATSON DISCOVERY, ASSISTANT & IBM CLOUD SERVICES

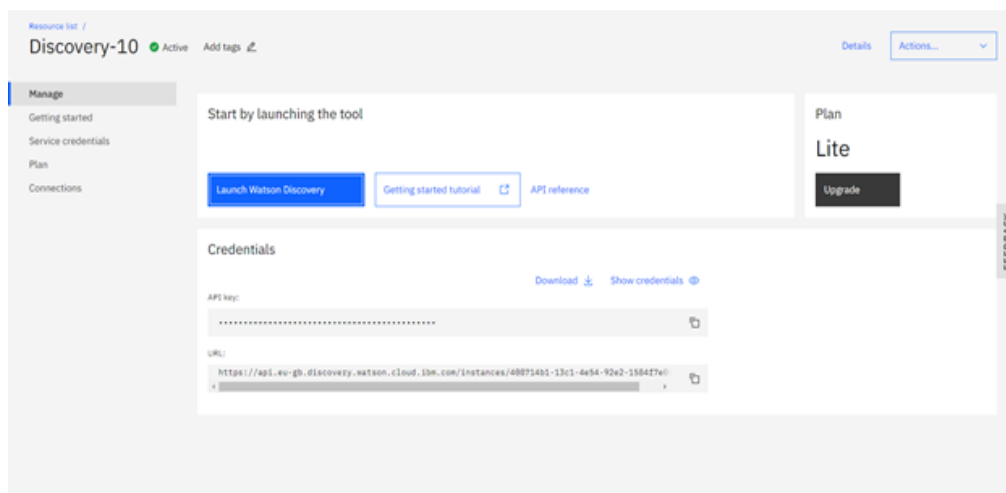
## WATSON ASSISTANT

1. In IBM cloud go to Catalog, search Watson Assistant.
2. Let the details be as default values and click Create.
3. Let the services be activated.



## WATSON DISCOVERY

1. Go to Catalog, search Watson Discovery.
2. Let the details be as default values and click Create.
3. Let the services be activated.



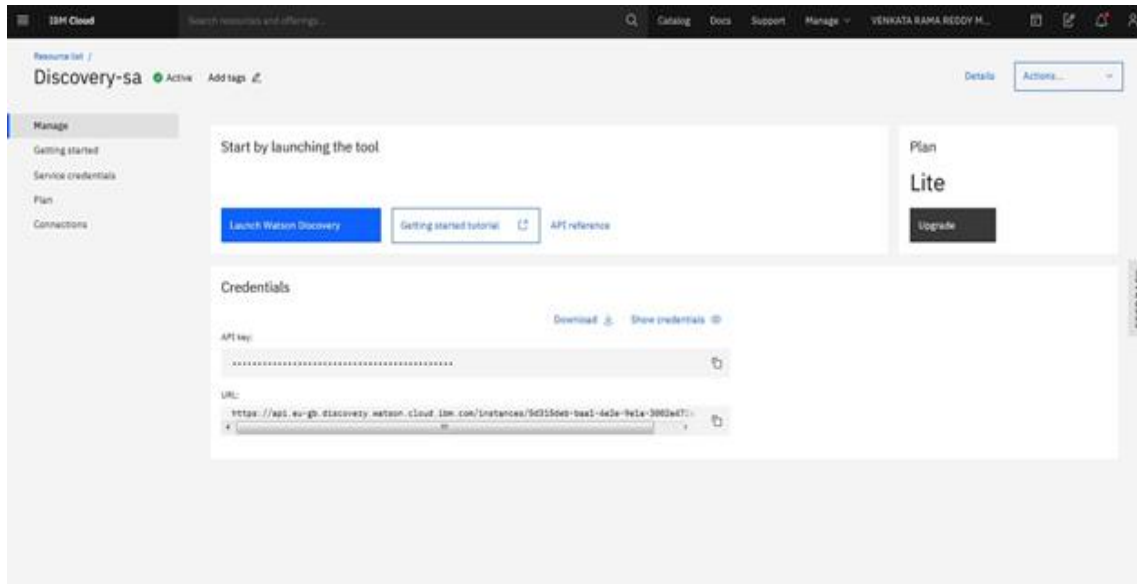
## CONFIGURATION OF WATSON DISCOVERY

After creating and launching the discovery from the resources we have to attach the document i.e., **Heater user document** from the local documents.

It is generally the basic user manual in which it describes the features and uses of the different components and the keys. So without the smart document the result won't be that accurate enough. So with this SMART DOCUMENT the user can get the accurate results. This can be done easily by clicking on the configure setting and then labelling each word or element present in the document as their respective label such as title, subtitle, text, image and Footer. Some of the labels are not present in the lite plan. In the lite plan we are provided with limited content of IBM Watson, the labels help us in segmentation of the document which helps the discovery to understand the document better and provide better results. The results provided by the discovery can be improved, all the results are shown in assistant in which the discovery finds the sentiment to be positive i.e. matching between the question or query entered by the user and the data of the document.

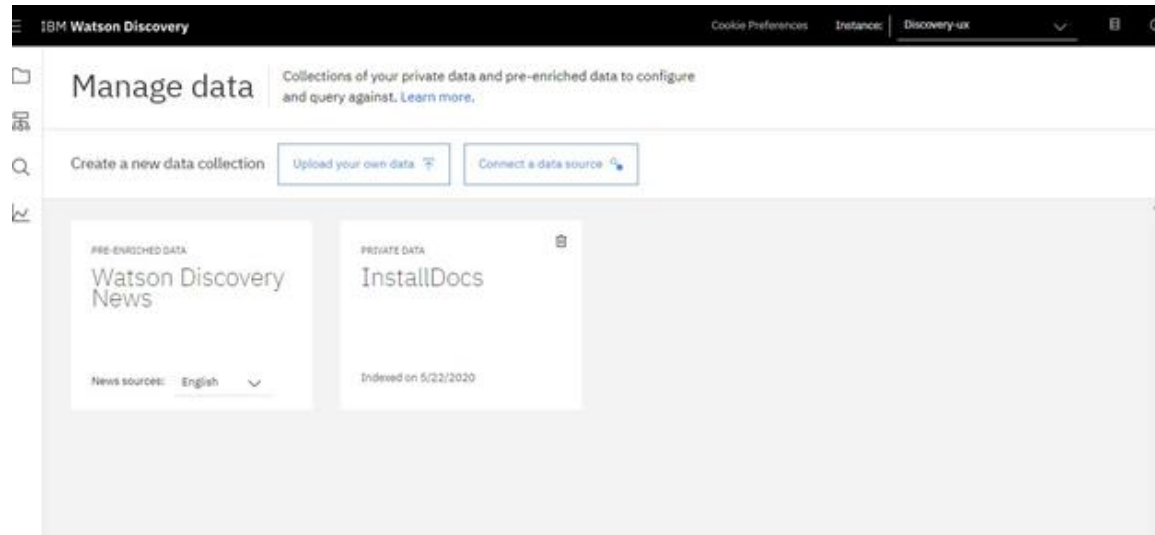
Follow the below mentioned steps:

After creating the discovery from the catalog, we will be redirected to this base page of discovery where the name of the discovery along with its API Key and URL are mentioned. These credentials will be used in further steps.

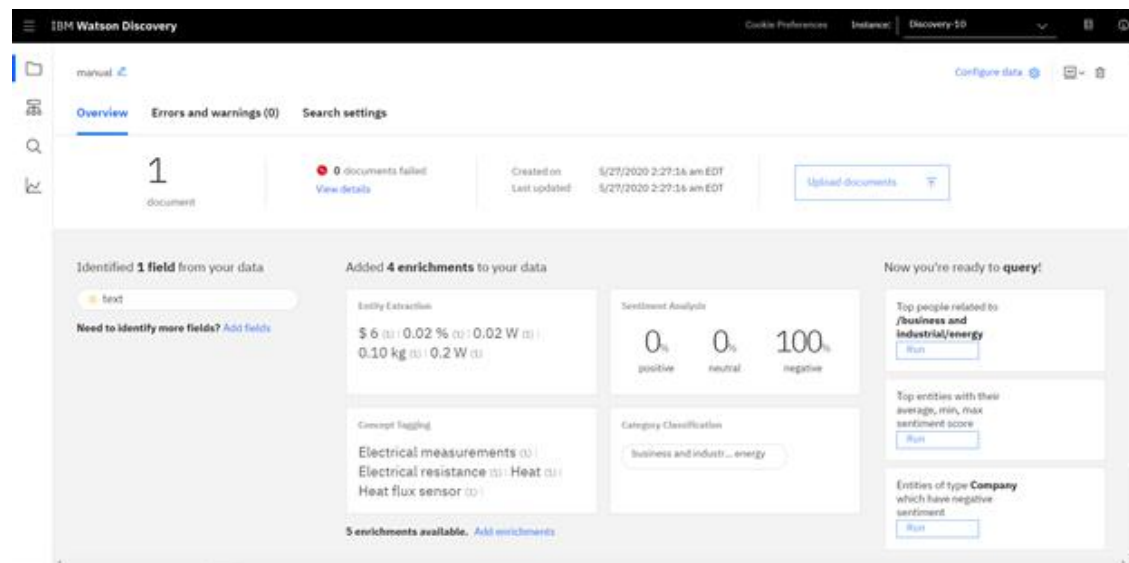


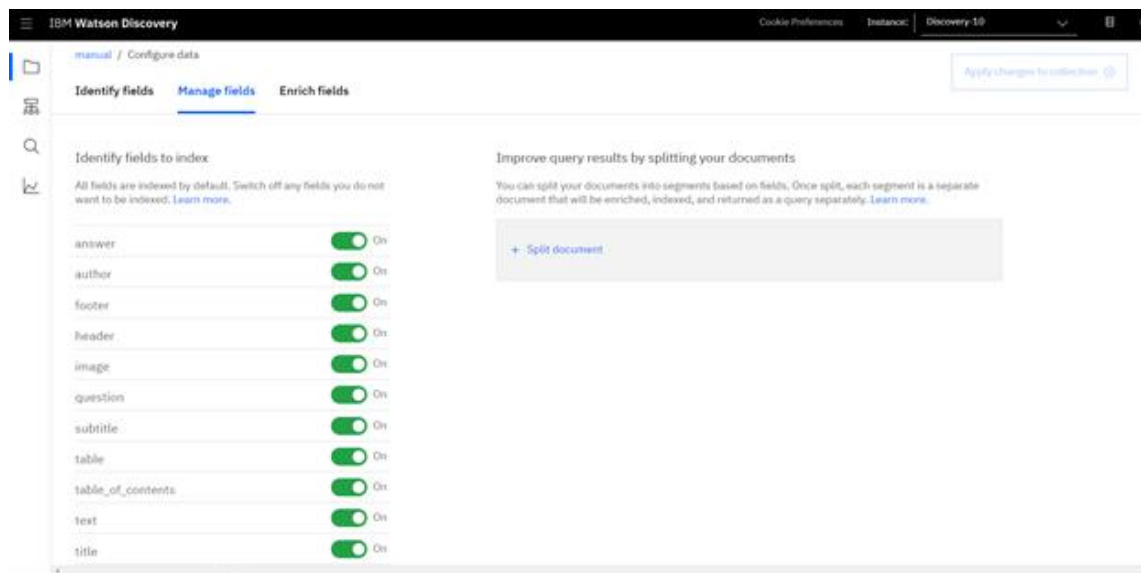
Click on the Launch Watson Discovery to launch the discovery.

Upload the data

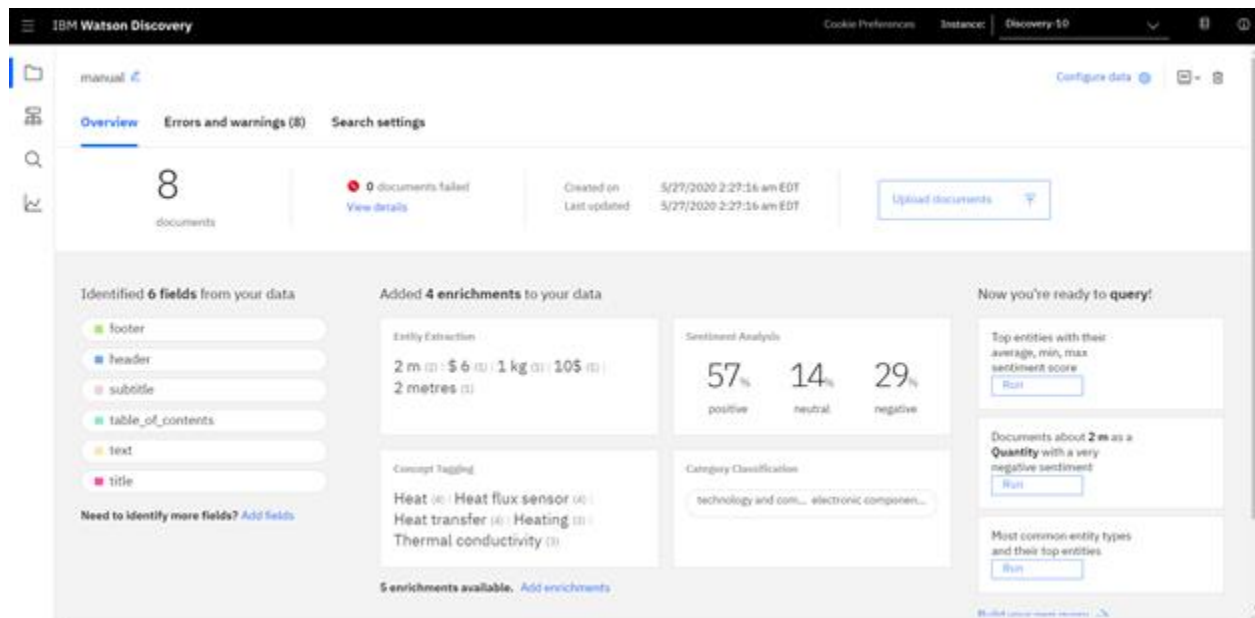


After uploading the document we will get a page where we can neglect the warnings and then we have to train the data by customizing it.





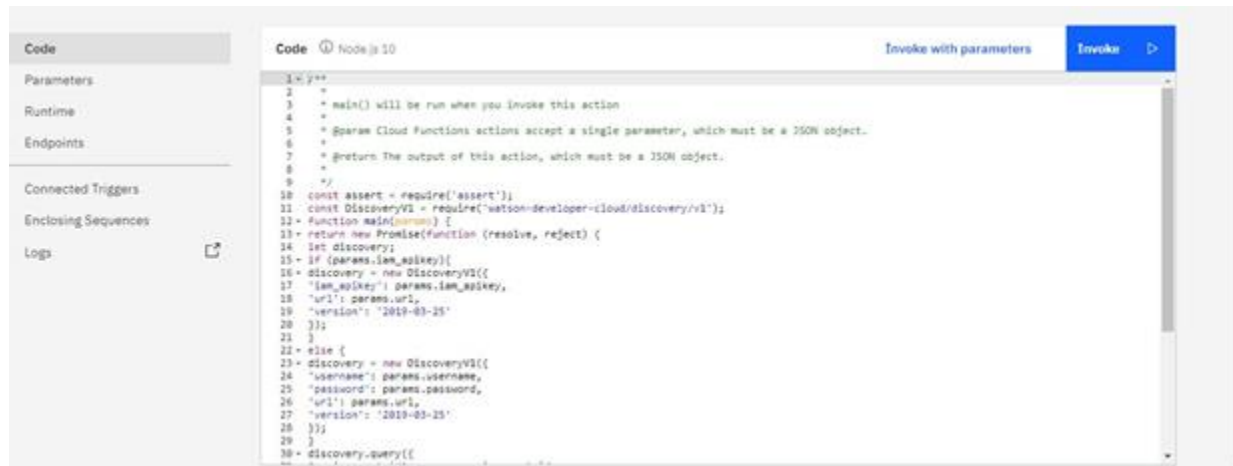
Then after customizing the data, the data need to be split according to different types here we consider it as subtitles format.



We already have the API key and the URL.

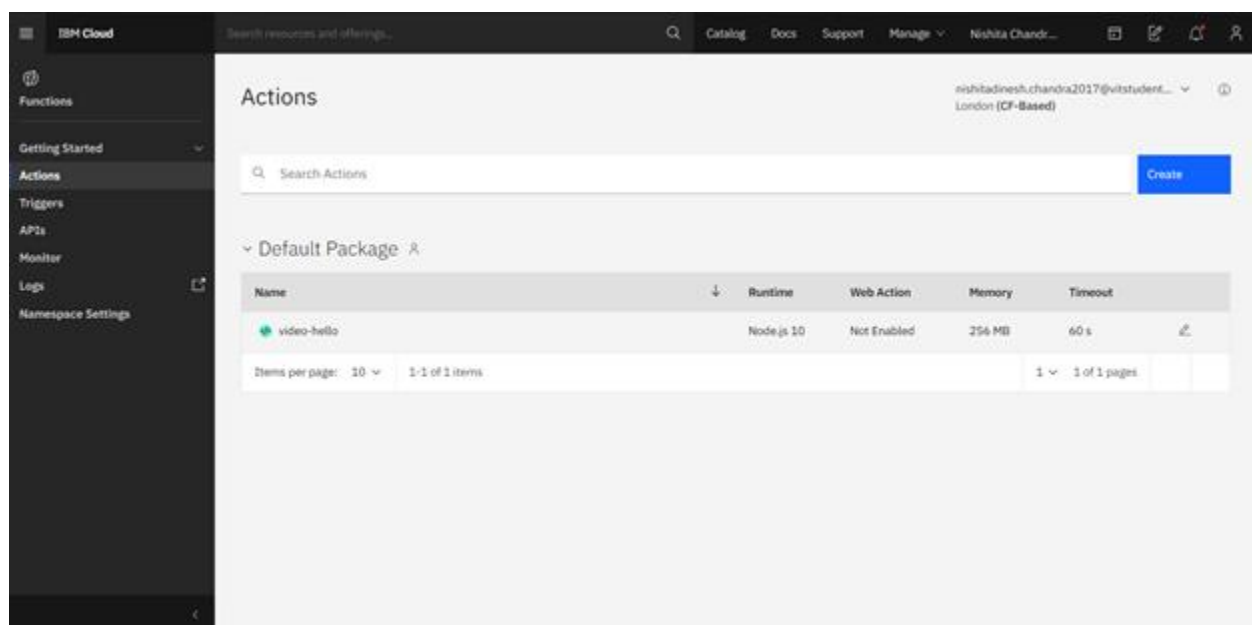
Next, we have to create the IBM Cloud Function Action: It is used to link the discovery with assistant, so that our queries can be answered by the discovery. After selecting the action from the IBM catalog, we have to click on the action tab as shown on the left

menu. Here we made the Information function. Then we can post the code which will help us to link the discovery.



The screenshot shows the IBM Cloud Functions code editor. On the left, there is a sidebar with tabs: Code, Parameters, Runtime, Endpoints, Connected Triggers, Enclosing Sequences, and Logs. The 'Code' tab is active, displaying a JavaScript function. The function is a Node.js 10 action that uses the 'watson-developer-cloud/discovery/v1' API. It takes parameters like 'iam\_apikey', 'url', and 'version'. The function logic includes an 'if' statement to check for 'iam\_apikey' and a 'else' block for other parameters. It uses the 'discovery' module to perform a query. At the top right, there are buttons for 'Invoke with parameters' and 'Invoke'.

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



The screenshot shows the IBM Cloud Functions 'Actions' page. The left sidebar contains navigation links: Functions, Getting Started, Actions, Triggers, APIs, Monitor, Logs, and Namespace Settings. The 'Actions' page has a search bar and a 'Create' button. Below the search bar, there is a section for 'Default Package' with a table listing actions. The table has columns: Name, Runtime, Web Action, Memory, and Timeout. One action is listed: 'video-hello' with runtime 'Node.js 10', 'Not Enabled' web action, '256 MB' memory, and '60 s' timeout. At the bottom, there is a pagination bar showing 'Items per page: 10' and '1 of 1 items'.

Name	Runtime	Web Action	Memory	Timeout
video-hello	Node.js 10	Not Enabled	256 MB	60 s

We can make the parameters as per the code and paste the parameter value from the discovery credentials. After that we have to click on the endpoint and enable the web action which will generate a public URL and it will be further used.



IBM Cloud Search resources and offerings...

Functions / Actions / video-hello

video-hello Web Action

Namespace: nishitadinesh.chandra2017@vitstudent.ac.in\_dev(London)

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Parameters

Parameter Name	Parameter Value
url	"https://api.eu-gb.discovery.watson.cloud.ibm.com/instances/408714b1-13c1-4e5"
environment_id	"b28f8eb0-9cf3-4ec9-b07d-14ef9a2aaa6f"
collection_id	"14e553ba-3e4b-4179-88b2-e7352e523aa9"
iam_apikey	"HL5NfGXHacp-EG5bqGf2u5EKUoh_Kvu1e40na,mWS_"

Add Parameter

Code

Parameters

Runtime

Endpoints

Connected Triggers

Enclosing Sequences

Logs

Web Action

☒ Enable as Web Action

Allow your Cloud Functions actions to handle HTTP events. Web Actions allow to control the response data and type by using a set of URL extensions, such as .json or .html. Learn more about [Web Actions](#). Note: The Web Action URL below requires to return a dict object that contains a body property.

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

HTTP Method	Auth	URL
ANY	Public	https://eu-gb.functions.cloud.ibm.com/api/v1/web/heddyrama311%40gmail.com_dev/default/disco-action-1

REST API

HTTP Method	Auth	URL
POST	API KEY	https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/heddyrama311%40gmail.com_dev/actions/disco-action-2

CURL

```
curl -u API-KEY -X POST https://eu-gb.functions.cloud.ibm.com/api/v1/namespaces/heddyrama311%40gmail.com_dev/actions/disco-action-2?looking=true
```

## CONFIGURATION OF WATSON ASSISTANT

Next, we have to make the Watson assistant and use the sample customer care skill for convenience. We can add intent related to product information and the related entities and dialog flow.

IBM Watson Assistant Lite Upgrade

Customer Care Sample Skill

Save new version Try it

Intents

Intents (10) ↑	Description	Modified T1	Examples T1
#Cancel	Cancel the current request	5 minutes ago	7
#Customer_Care_Appointments	Schedule or manage an in-store appointment.	5 minutes ago	20
#Customer_Care_Store_Hours	Find business hours.	5 minutes ago	48
#Customer_Care_Store_Location	Locate a physical store location or an address.	5 minutes ago	25
#General_Connect_to_Agent	Request a human agent.	5 minutes ago	47
#General_Greetings	Greetings	5 minutes ago	30
#Goodbye	Good byes	5 minutes ago	6
#Help	Ask for help	5 minutes ago	8
#Product_Information		a few seconds ago	5

Showing 1-10 of 10 intents

1 1 of 1 pages

The contents in the document need to be given as example in the intents so that when the customer needs any help the answer can be resulted using the keywords i.e., the examples.

IBM Watson Assistant Lite Upgrade

Customer Care Sample Skill

Save new version

Intents

Entities

Dialog

Options

Analytics

Versions

Content Catalog

Ask about product

Customize

Node name will be shown to customers for disambiguation so use something descriptive.

Settings

If assistant recognizes

#Product\_Information

Intents

#Product\_Information How to turn on heater

Text

Enter response text

Response variations are set to **sequential**. Set to **random** | **multiline**

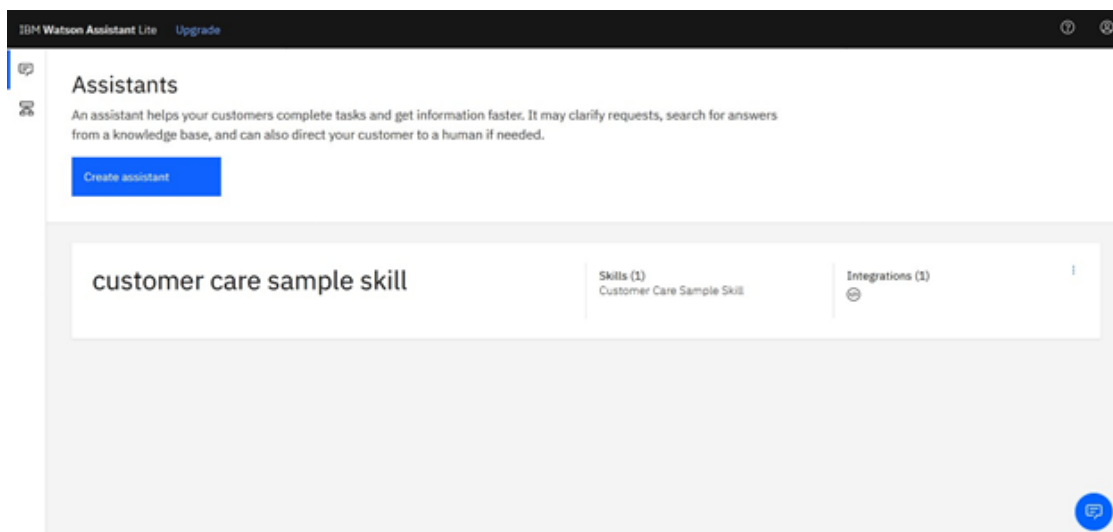
Learn more

Add response type

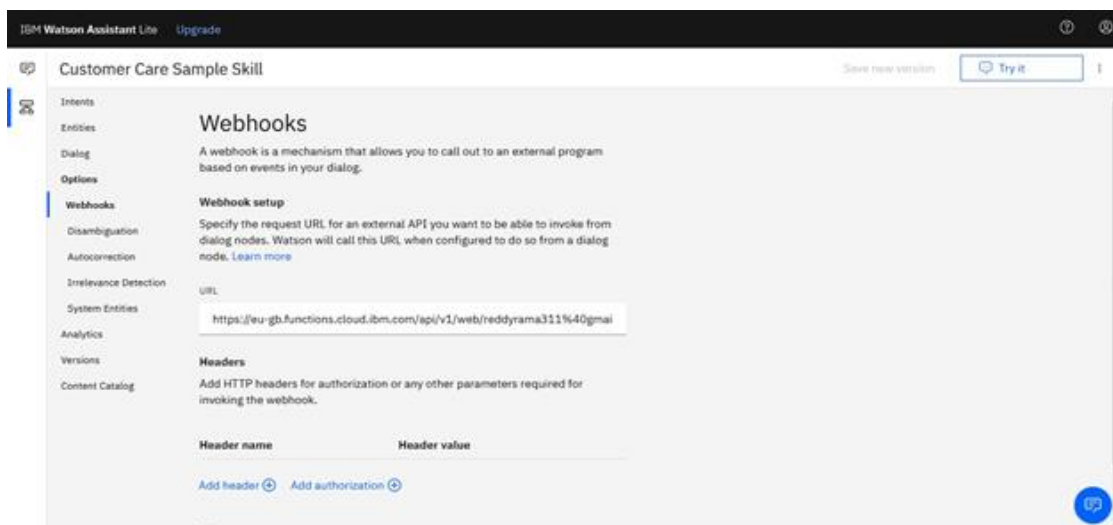
After adding all the examples, we have to add a node in the dialog i.e., Ask about product where the webhook url is given so that it takes the document from the intent and searches the result.

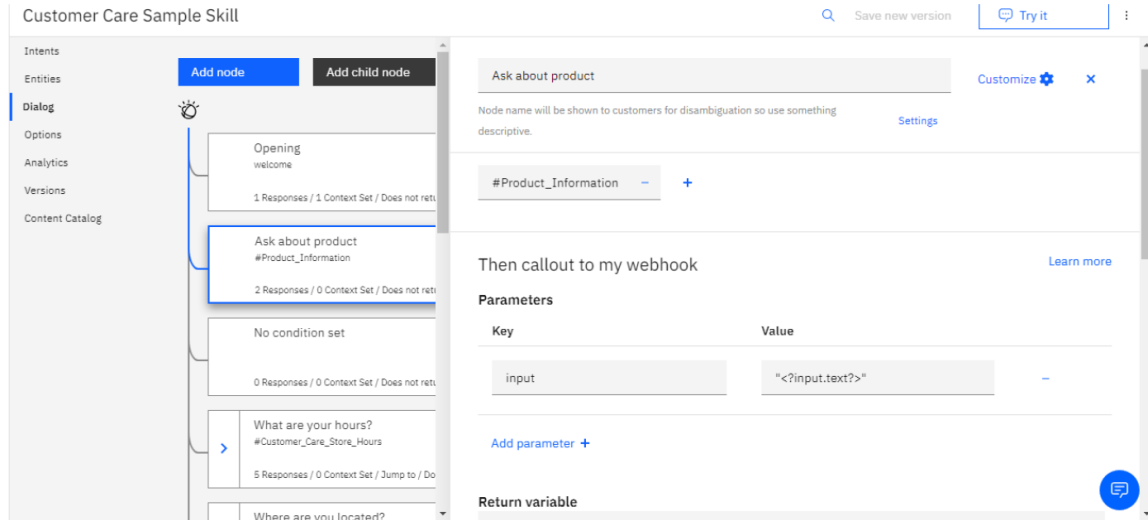
We have to enable the webhooks which enables our dialog to send a POST request to the webhook URL.

If the assistant recognizes the intent then the webhook is to be called and returns the value as `webhook_result_1` by taking the input parameters.



Add the URL generated in cloud to the Webhooks.





## BUILDING NODE-RED FLOW TO INTEGRATE ALL SERVICES

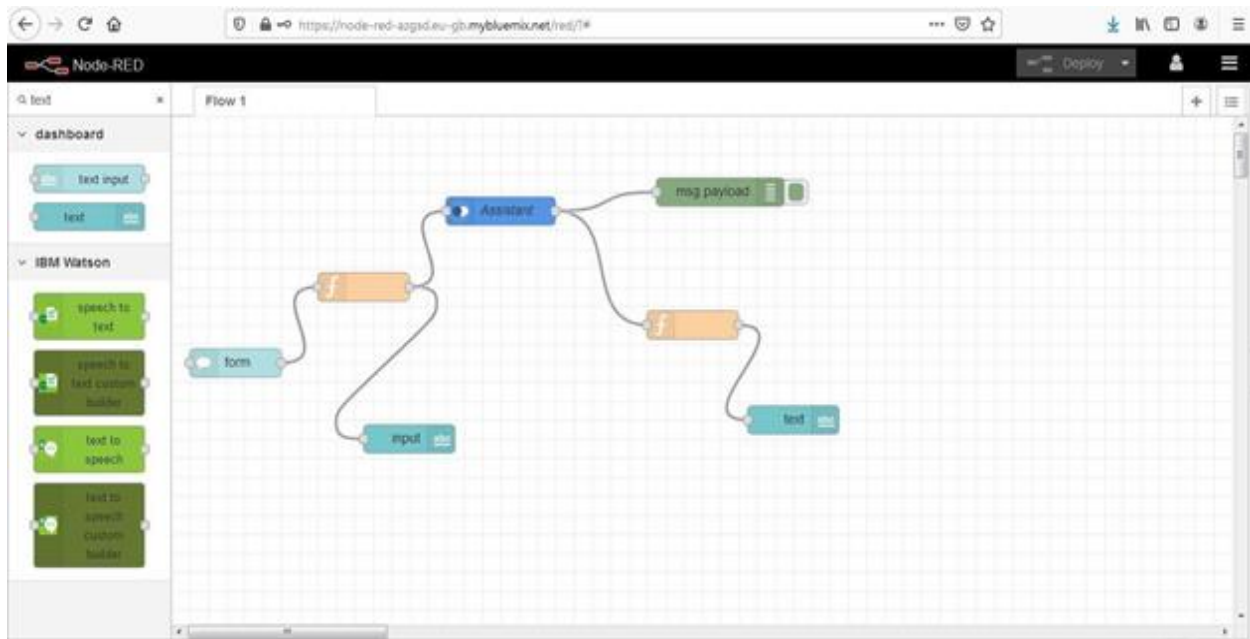
After this we have to make the **NODE-RED** flow, and link everything. We will get a UI from the node.

Go to the Node-RED editor and create your own flow.  
Install the node-red dashboard palette.



Create nodes and edit them.

The final flow will look like as shown below.



Edit the forms by clicking on them. Once done click Deploy.

Take the dashboard URL and add /ui to open the chatbot.

The UI we have is the basic one but can be improved by writing the HTML code in the template node. We can vary the background colour also from the node-red. This is how the initial flow looked like but we imported another flow just to improve the UI This basic flow can be extended to what we have used next, we have imported the flow to make things easy and to have an interesting UI.

# RESULT: INTELLIGENT CUSTOMER HELP DESK

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

The image displays two screenshots of a chatbot interface, likely from a Node-RED dashboard. Each screenshot shows a chat window with a title bar, an input field, and a response area.

**Top Screenshot:**

- Chatbot** (Title)
- Enter your input \*** (Label)
- Hello** (User Input)
- SUBMIT** (Button)
- CANCEL** (Button)
- You : Hello** (User Message)
- Bot** (Label)
- Hi there. How can I help you this morning?** (Bot Response)

**Bottom Screenshot:**

- Chatbot** (Title)
- Enter your input \*** (Label)
- How to turn on heater** (User Input)
- SUBMIT** (Button)
- CANCEL** (Button)
- You : How to turn on heater** (User Message)
- Bot** (Label)
- "Please refer to section 2.1 for more details on how to compute the heater power and applied heat flux. Suggested HTR01 wiring is shown in figure 4.2.1. The heater serial number and resistance are shown on the HTR01 product certificate and on the sticker on the strain relief block."** (Bot Response)

## Chatbot

Enter your input \*

How to change time of heater

SUBMIT

CANCEL

You : **How to change time of heater**

Bot

**"Upon installing the heat flux sensor and HTR01, a reference measurement should be made. A time trace of the heater power, the measured heat flux and the measured sensor temperature should be stored as reference data."**

## Chatbot

Enter your input \*

How to set time in heater

SUBMIT

CANCEL

You : **How to set time in heater**

Bot

**"This can be done accurately by using the four heater wires to conduct a four-point resistance measurement. 3. Record a time trace of the heater power, the measured heat flux and the sensor temperature; the same parameters as in the reference data. Normalise the data by the heater power."**

## Chatbot

Enter your input \*

What is a heater

SUBMIT

CANCEL

You : **What is a heater**

Bot

"If a variable heat flux is required, the heater can either be connected via a solid state relay controlled by a pulse-width modulated (PWM) signal or to a programable power supply (see figure 4.2.1). The HTR01 electrical connections are explained in table 4.2.1 When connecting HTR01, always observe the rated heater voltage."



# **ADVANTAGES, DISADVANTAGES AND**

## **APPLICATIONS**

### **ADVANTAGES**

- More efficient than the previous manuals.
- Reduces work load on the employees.
- Results in accurate and takes less time.
- Solves issues faster.
- Deliver faster, smarter, more personalized service.

### **DISADVANTAGES**

- Sometimes it may result in wrong answer where there may be issue in the sentimental analysis.
- It may take some time to display the result when there is multiple occurrence of the keyword in the document.
- Giving same answer for different sentiments.
- If the data is not trained properly then the result will not be accurate.

### **APPLICATIONS**

- Chatbot can be applied in various fields to help the customer in finding the result in larger documents.
- It can be used to find the data in social medias and in any communication channel.

## **CONCLUSION & FUTURE SCOPE**

By using this process, we can create a basic chatbot that helps the customer to clear their basic needs and issues, with the help of IBM WATSON ,WATSON ASSISTANT and WASTON DISCOVERY for the data to be imported from the local computers and NODE-RED application to show the flow of the data which results in the output with the usage of the webhooks. And we have successfully created the smart help desk using these applications.

We can update the data by uploading the pre-built node red flow and then modifying it and then deploying it. The data in the functions can be modified once if we change the documents that need to be processed. We can also improve the results of discovery by enriching it with more fields. We can also include Watson text to audio and Speech to text services to access the chatbot handsfree. These are few of the future scopes which are possible.

# APPENDIX

## CODE

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

```
const assert = require('assert');
const DiscoveryV1 = require('watson-developer-cloud/discovery/v1'); function
main(params) {
  return new Promise(function (resolve, reject) {
    let discovery;
    if (params.iam_apikey) {
      discovery = new DiscoveryV1({
        'iam_apikey': params.iam_apikey, 'url': params.url,
        'version': '2019-03-25'
      });
    }
    else {
      discovery = new DiscoveryV1({
        'username': params.username, 'password': params.password, 'url':
params.url,
        'version': '2019-03-25'
      });
    }
    discovery.query({
      'environment_id': params.environment_id, 'collection_id':
params.collection_id, 'natural_language_query': params.input, 'passages':
true,
      'count': 3,
      'passages_count': 3
    }, function (err, data) {
      if (err) {
        return reject(err);
      }
      return resolve(data);
    });
  });
}
```

## **REFERENCES**

1. <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
2. <https://github.com/watson-developer-cloud/node-red-labs>
3. <https://www.ibm.com/watson/products-services>
4. <https://developer.ibm.com/components/watson-assistant/series/learning-path-watson-assistant>
5. <https://developer.ibm.com/articles/introduction-watson-discovery/>