PROJECT REPORT

AISHWARYA CHALLA
Mittuchalla@gmail.com

TITLE - Intelligent Customer Help Desk With Smart Document Understanding

CATOGRY - ARTIFICIAL INTELLIGENCE

INTRODUCTION

OVERVIEW:

Project Requirements: Python, IBM, Watson Assistant
Functional Requirements: IBM Cloud
Software Requirements: Watson Assistant, Watson Discovery
Project Deliverable: Smart bridge
Technical Requirements: AI, ML, PTYHON, WATSON AI
Project Duration: 19 Days

PURPOSE:

The typical customer care chat bot can answer simple questions, such as store locations and hours, directions, and perhaps 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 is not 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 the 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 queries.

SCOPE:

- 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

LITERATURE SURVEY

Existing problem:

Generally Chatbots means getting input from users and getting only response questions and for some questions the output fr om bot will be like “try again ”, “I don’t understand ”, “ will you repeat again ”, and so on... and directs customer to customer agent but a good customer 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 clarifies his doubts with fast chatbots.

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 trained from some insight records based company background so it can answer queries based on the product or related to company. In this project I used Watson Discovery to achieve the above solution. And later including Assistant and Discovery on Node-RED

THEORITICAL ANALYSIS

FLOW DIAGRAM:

- The document is annotated using Watson Discovery.
- The user interacts at the backend with the server, and the front end the chat bot.
- The dialog between the server and chat bot is coordinated with a Watson Assistant skill.
- If the user asks a product operation question, a search query is passed to a predefined IBM Cloud Functions action.
- The cloud function will query the Watson service and return a response.

HARDWARE/SOFTWARE DESIGNING:

- Create all IBM services
- Configure the Watson Discovery with the data set
- Create the cloud function using IBM Cloud
- Configure the Watson Assistant,
- Create the node red-flow
- Deploy the node red-flow

EXPERIMENTAL INVESTIGATION

1. Create all the required IBM Services.
2. Configure the Watson Discovery:

Go to Watson Discovery in IBM Cloud, and create your discovery page and launch the discovery tool.

Upload your data collection by giving it a name, using select your file option, and now annotate your data with SDU by clicking on configure data button and selecting identify fields option.

Then train your data collection by annotating the data.

Text-Light Yellow

Footer-Light Green

Table of contents-Light Blue

Title-Pink

Subtitle-Light Purple

Click on "Apply changes to collection." An "upload your document" dialog box appears, and attach your file. Now in the overview page we are expected to see more than preferably 30 documents and 4 fields that are identified from your data.
Now click on the "Manage your fields" option to improve your query results by splitting the document.
And click on save changes button, then we can give a sample query and run the document.

3. IBM CLOUD FUNCTION

Go to IBM Cloud and create an action. Give a name to it and click select a language that you prefer in the Runtime option.
Write down the code in the space provided and click on save.
Go to parameters tab and create the following parameters:
-> iam_apikey
-> url
->collection_id (copy paste this from the environment details button from your discovery page).
->configuration_id (copy paste this from the environment details button from your discovery page).
-> environment_id (copy paste this from the environment details button from your discovery page).

Click on the endpoints tab, and click on the option "enable as webhooks", and copy the public url.

4. WATSON ASSISTANT

Create an assistant, and click on the "Launch Assistant" tool. Create your skill.
In which you have to create intents and a dialog. After that you have to go to the options button and click on webhooks.
Paste the link by giving a.json extension.
So let your chat bot get trained and try giving a few queries and run it.

5. CREATE NOD RED FLOW

- Drag and drop the assistant node. Give the node a name, and give the API Key, Workspace id, Service endpoint details. Click on done.
- Drag and drop the form node from the dashboard tab, and give a name to it.

Fill in the other details.
Label-Enter the text,
Type-Text
Name-Text
And name the buttons - SUBMIT YOUR QUERY and CANCEL YOUR QUERY.
And click on done.

- Drag and drop the text node from the dashboard tab.
- Drag and drop the function node. Give it a name, and write down the code to get the input passed.
- And connect the form and text node to the function node, the function node to the assistant node.
- Drag and drop the debug node to display an output, and the text node along with the function node.
- In the function node write down the code to display to display the text and click on done.
- And then deploy your flow, and run the chat bot.

FLOWCHART

At first go to manage pallet and install dashboard. Now, create the flow with the help of the following node:

Inject
Assistant
Debug
Function
Ui_Form
Ui_Text

RESULT

We could make a chat bot that answers relevantly to a give data collection.
https://mittuchat.eu-gb.mybluemix.net/ui/#!/0?socketid=eVMZGiz7KWaUPoDbAAAO

ADVANTAGES & DISADVANTAGES

Advantages:

Companies can deploy chatbots to rectifiy simple and general human queries.
Reduces man power
Cost efficient
No need to divert calls to customer agents and customer agents can look on other works.

Disadvantages:

Some times chat bot can mislead customers
Giving the same answer for different sentiments.
Sometimes they cannot connect to customer sentiments and intentions.

APPLICATIONS

It can deploy in popular social media applications like facebook,slack,telegram.
chat bot can deploy any website to clarify basic doubts of viewers.

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.

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 scopes of this project.

BIBLIOGRAPHY
APPENDIX

NODE RED FLOW:


```
[
{
"id":"c683bbd9.5df328",
"type":"tab",
"label":"WatsonAssisstant",
"disabled":false,
"info":""
},
{
"id":"62a56248.57adbc",
"type":"watson-conversation-v1",
"z":"c683bbd9.5df328",
"name":"Assisstant",
"workspaceid":"da541be7-4f08-4ad3-93c7-2597ae44461e",
"multiuser":false,
"context":false,
"empty-payload":false,
"service-endpoint":"https://api.eu-gb.assistant.watson.cloud.ibm.com/instances/49839
9c1-6148-49bc-9fdc-d6feec9a4f7b",
"timeout":"",
"optout-learning":false,
"x":375,
"y":180,
"wires":[["62015b9e.d93144","c051d27d.eb886"]],
"l":false
},
{
"id":"64122a47.27e344",
"type":"function",
"z":"c683bbd9.5df328",
"name":"F1",
"func":"msg.payload = msg.payload.text; \nreturn msg;",
```

```
"outputs":1,
"noerr":0,
"x":255,
"y":260,
"wires":[["62a56248.57adbc","d51124f.6039bd8"]],
"l":false
},
{
"id":"e57a0dc7.b512f",
"type":"ui_form",
"z":"c683bbd9.5df328",
"name":"",
"label":"form",
"group":"648afa69.071244",
"order":1,
"width":0,
"height":0,
"options":[{"label":"Enter the input",
"value":"text",
"type":"text",
"required":true,
"rows":null}],
"formValue":{"text":""},
"payload":"",
"submit":"submit",
"cancel":"cancel",
"topic":"",
"x":95,
"y":280,
"wires":[["64122a47.27e344"]],
"l":false
},
{
"id":"62015b9e.d93144",
"type":"function",
"z":"c683bbd9.5df328",
"name":"F2",
```

"func":"if(msg.payload.output.error){\n   msg.payload = \"please rephrase\";\n   return msg;\n}\nmsg.payload = msg.payload.output.text[0];\nreturn msg;",
"outputs":1,
"noerr":0,
"x":515,
"y":240,
"wires":[["3db259c6.ab0bd6"]],
"l":false
},
{
"id":"d51124f.6039bd8",
"type":"ui_text",
"z":"c683bbd9.5df328",
"group":"648afa69.071244",
"order":3,
"width":0,
"height":0,
"name":"",
"label":"You",
"format":"{{msg.payload}}",
"layout":"row-spread",
"x":395,
"y":360,
"wires":[],
"l":false
},
{
"id":"3db259c6.ab0bd6",
"type":"ui_text",
"z":"c683bbd9.5df328",
"group":"648afa69.071244",
"order":6,
"width":0,
"height":0,
"name":"",
"label":"bot",
"format":"{{msg.payload}}",

```
"layout":"col-center",
"x":670,
"y":360,
"wires":[]
},
{
"id":"c051d27d.eb886",
"type":"debug",
"z":"c683bbd9.5df328",
"name":"",
"active":true,
"tosidebar":true,
"console":false,
"tostatus":false,
"complete":"false",
"x":540,
"y":80,
"wires":[]
},
{
"id":"648afa69.071244",
"type":"ui_group",
"z":"",
"name":"CHATBOT",
"tab":"25f29110.d110ae",
"order":1,
"disp":true,
"width":13,
"collapse":false
},
{
"id":"25f29110.d110ae",
"type":"ui_tab",
"z":"",
"name":"CUSTOMERCARE",
"icon":"",
"disabled":false,
```

```
"hidden":false
}
]
```

CLOUD FUNCTION:

```
/**
 *
 * @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);
  });
 });
}
```