ADCET Smart Student Care Chatbot

Contents

1. Introduction:

1.1 Overview

The project is titled as "Intelligent Customer Help Desk with Smart Document Understanding". Here a chatbot is created which unlike other chatbots which answers simple questions and if the particular question is out of the scope then the chatbot typically replies in a manner stating that the question is invalid, intends to answer operational questions by using IBM Watson Discovery service and redirect the customer to the owner's manual that is uploaded in the Watson Discovery sevice. This chatbot leads the customer to the desired section in the owner's manual where the solution for the problem exists instead of redirecting to a customer service.

1.1 Purpose

The purpose of the project is to save up on the time and money of the owner or manager of the chatbot by reducing the need to connect to a customer care service person by including Smart document Understanding in the chatbot which answers to most of the operational doubts from the organisation's manual thus saving up on time and increasing the efficiency.

2. LITERATURE SURVEY

2.1 Existing Problem

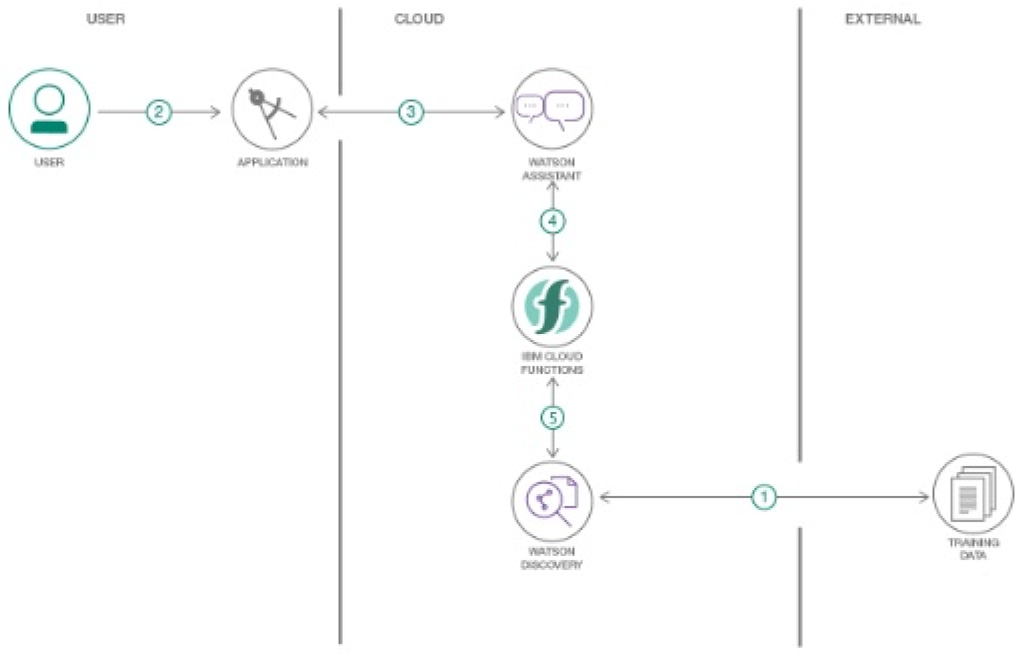
The typical student care chat bot can answer simple questions, such as admission Procedure and hours, directions, and perhaps even making appointments. When a question falls outside of the scope of the pre-determined question set, the option is typically to tell the customer the question isn’t valid or offer to speak to a real person.

2.2 Proposed solution

In this project, there will be another option. If the customer question is about the college Information, the application shall pass the question onto Watson Discovery Service, which has been pre-loaded with the college manual. So now, instead of “Would you like to speak to a customer representative?” we can return relevant sections of the college manual to help solve our student’s problems.

3. THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

Searches the ADCET prospectus and gives back the relavant answer to the user's query by parsing the document using IBM discovery and reture the results in the chatbot.

4. EXPERIMENTAL INVESTIGATIONS

The experimental investigations to get the most appropriate results are as follows

- Structuring the document/manual using discovery service of IBM. It includes omitting the unwanted text as footer and helping the service understand by laying out the title, subtitles, text, tables among others
- Trying to parse the file using the webhook to give the most appropriate answer.
- Modifying and checking which intent works best under which node of the dialog to give appropriate answer.
- Modifying and checking the node red flow so that the text is fitted well in the ui
- Modifying and changing the ui according to the need and layout
- Trying to integrate it with several platforms.

RESULT:

As you can see in the pictures, the user starts off with a normal conversation, the intent of which is recognized by the IBM watson assistant and relevant answer is fetched from the dialog.
When the user asks any operational query (in my case query about the Annasaheb Dange College of Engineering and Technology) the bot would simply parse the document(ADCET Document) using Watson Discovery and give back the section of the document as a reply to the user.

ADVANTAGES & DISADVANTAGES

One can cite several advantages of this chatbot.

- It reduces the time significantly by not making the user wait for getting connected to a customer care person for operational doubts
- Less waiting time for the user rather than waiting for being connected to the customer care
- Saves up cost for the owner of the organization by not having the need to employ more people
- Increases customer satisfaction

Although efficient overall it has some disadvantages.

- The user may not be satisfied by the answer that the chatbot fetches from the manual or document.

APPLICATIONS:

The application of this ADCET Smart Student Care Chatbot is

1. To answer operational queries
2. To give real time answers which are usually answered by connecting to a customer care
3. Helps in knowing the need of the users

CONCLUSION:

Thus, I have developed a chatbot using IBM Watson which uses the smart document understanding to fetch results from the file of college(ADCET Ashta) using Discovery service of IBM and gives returns with the section of the document containing the answer to the user's query.

FUTURE SCOPE

The future scope of the project is:

It can be scaled to answer other operational doubts by the college students about the infrastructure, exam and form filling queries.

Bibliography

(n.d.). Retrieved from Discovery: https://cloud.ibm.com/services/discovery/crn%3Av1%3Abluemix%3Apublic%3Adiscovery%3Aus-south%3Aa%2Fc6bd15308af24210af2c762365192a1b%3Ab301ee81-63f8-4803-ab6b-7a2514d01de3%3A%3A?paneId=manage

how-to-build-a-chatbot. (n.d.). Retrieved from cognitiveclass: https://cognitiveclass.ai/courses/how-to-build-a-chatbot

IBM Watson. (n.d.). Retrieved from https://cloud.ibm.com/services/conversation/crn%3Av1%3Abluemix%3Apublic%3Aconversation%3Aeu-gb%3Aa%2Fc6bd15308af24210af2c762365192a1b%3A7c1aeed4-d6fb-4d47-89b3-cb7d8221f533%3A%3A?paneId=manage

ibm-cloud-essentials. (n.d.). Retrieved from cognitiveclass: https://cognitiveclass.ai/badges/ibm-cloud-essentials

APPENDIX

Source code

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