“Application of Artificial Intelligence in Computer Science”

Arpit Jain
(arpit.jain@iiitdmj.ac.in)

Abstract

AI (Pronounced as AYE-EYE) or Artificial Intelligence is the ability of the computers to behave like humans. It’s basically a simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using the rules to reach approximate or definite conclusions), and self-correction. Particular applications of AI include expert systems, speech recognition, and machine vision. AI can be categorized in any number of ways, but here are two examples.

The first classifies AI systems as either weak AI or strong AI. Weak AI, also known as narrow AI, is an AI system that is designed and trained for a particular task. Virtual personal assistants, such as Apple's Siri, are a form of weak AI. Strong AI, also known as artificial general intelligence, is an AI system with generalized human cognitive abilities so that when presented with an unfamiliar task, it has enough intelligence to find a solution. The Turing Test, developed by mathematician Alan Turing in 1950, is a method used to determine if a computer can actually think like a human, although the method is controversial.

I want to demonstrate the recent uplift of AI in almost every domain imaginable. AI is employed in Healthcare, Business, Education, Finance, Law, Manufacturing and even in Consumer Products. A small example of AI you might have witnessed is the rise of chatbots. Chatbots on messaging apps allow brands to personally engage at scale with people on the platforms where they’re already spending a lot of time. They are computer programs that use machine learning to pick up on conversation patterns and mimic human conversation when reacting to spoken or written prompts. Chatbots can be connected to a variety of data sources via APIs to deliver information and services on demand, from weather forecasts to order requests.

Introduction

The aim of this project was to realize the potential of AI in imitating the Human Activities. Utilisation of Human Effort on simple activities like answering questions and responding to the queries is not so much productive. The better option is to employ machines for these purposes. Chatbots reduce human effort by performing these simple tasks with better efficiency and precision. Chatbots can handle situations in a versatile manner.

Chatbots have been part of our communication platforms for quite some time. You’ve definitely come across these conversational computer programs: they’ve helped you order a meal, book a cab, or even plan a vacation. According to the experts, private messaging apps are taking over social media platforms. For instance, Messenger, Facebook’s messaging platform, has over 900 million users and 100,000 monthly active bots. With one-on-one and group chat becoming the dominant method of communication, there are almost unlimited opportunities for bots to contribute.

IBM Conversation

IBM cloud offers a service called IBM conversation which allows to Create and deploy customised chatbots for specific applications. The service provides users a special tool for training chatbots. In the Workspaces section, you can find the list of learned domains for your chatbots. Three important components of this service are – Intents, Entities, and Dialog.

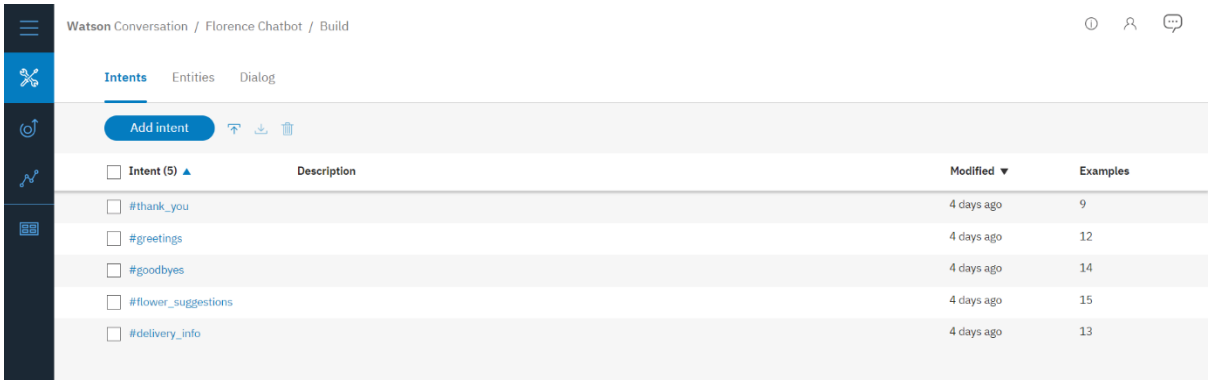
- **Intents.** The list of learned and programmed actions.
- **Entities.** The list of subjects that Watson will recognize during the conversation.
- **Dialog.** The tool for modeling the schema of a dialog. It allows you to specify sequences of questions and typical answers to these questions.

The intent in general terms means to specify a set of actions or purposes which the chatbot would handle. (Example – Greetings, Asking for information; etc.). Entities are the specific details or objects to which the chatbot will pay special attention when the user enters their message. (Example – Person's Name, Color; etc.). Dialog refers to the flow in which the chatbot proceeds. It simply defines the Order or flow of the chatbot interface.

Florence – Flower Shop Chatbot

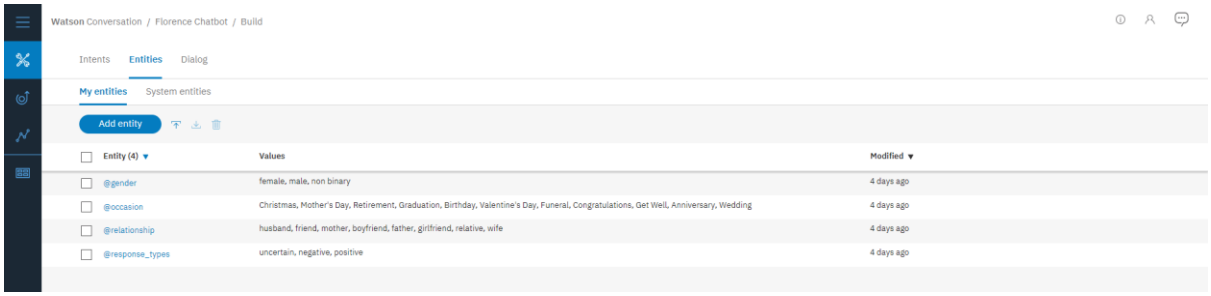
For demonstration purpose, I decided to build a chatbot which might be useful for query answering in a typical flower shop. Florence will attempt to answer the Delivery and suggestion related queries. I identified the following Intents and entities for this Chatbot Interface.

Intents



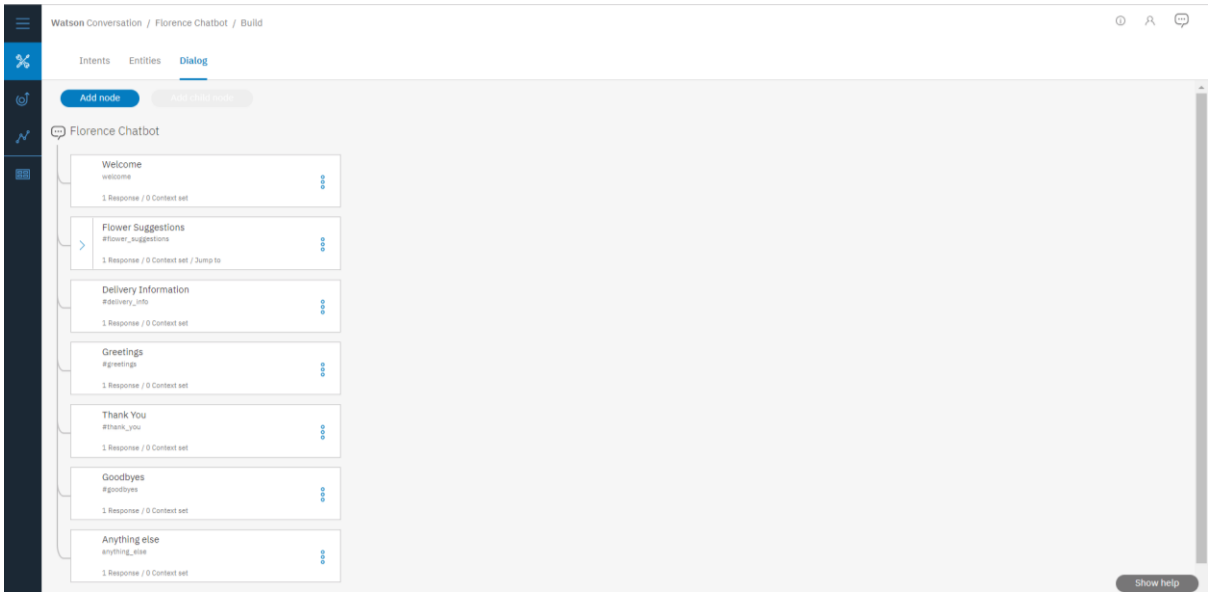
Each Intent Contains several examples of how a user might Use that particular Intent.

Entities



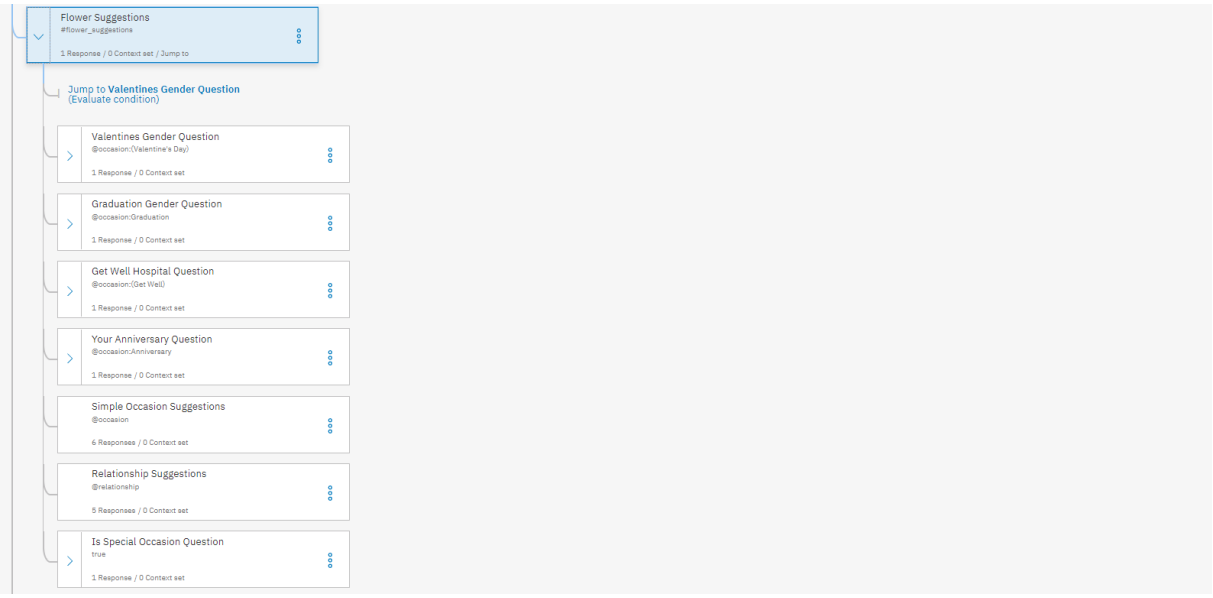
Each Entity contains the specific values that particular intent might have in the user's input.

Dialog Flow



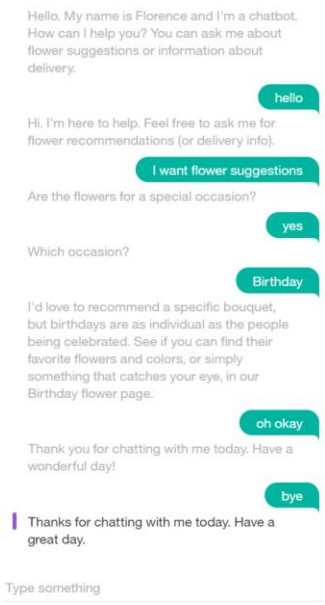
- **Welcome Node gets executed at the beginning of the Chat. (Provides a sort of start to the chat)**
- **Anything Else Node gets executed when no other condition gets matched. (Florence Doesn't recognize the User Input)**

A node might have a Child Node which carries out further Actions in the case when the Parent Node condition matches.



Application for Chatbot

IBM provides a documentation repository to develop a user application using their Sample app. The sample is based on the node.js(Backend) and HTML/CSS(Frontend).



- **Left Side shows the Actual chat between user and Florence.**
- **Right Side shows the inference from the Chat.**

User input


```
{
  "input": {
    "text": "bye"
  },
  "context": {
    "conversation_id": "739821c7-5f62-4fe4-a232-91ce21aab492",
    "system": {
      "dialog_stack": [
        {
          "dialog_node": "root"
        }
      ],
      "dialog_turn_counter": 6,
      "dialog_request_counter": 6,
      "_node_output_map": {
        "Welcome": [
          0
        ],
        "Greetings": [
          0
        ],
        "Is Special Occasion Question": [
          0
        ],
        "A Special Occasion": [
          0
        ],
        "node_22_1502215786664": [
          0
        ],
        "Goodbyes": [
          0
        ]
      },
      "branch_exited": true,
      "branch_exited_reason": "completed"
    }
  }
}
```

Watson understands


```
{
  "intents": [
    {
      "intent": "goodbyes",
      "confidence": 1
    }
  ],
  "entities": [],
  "input": {
    "text": "bye"
  },
  "output": {
    "text": [
      "Thanks for chatting with me today. Have a great day."
    ],
    "nodes_visited": [
      "Goodbyes"
    ],
    "log_messages": []
  },
  "context": {
    "conversation_id": "739821c7-5f62-4fe4-a232-91ce21aab492",
    "system": {
      "dialog_stack": [
        {
          "dialog_node": "root"
        }
      ],
      "dialog_turn_counter": 7,
      "dialog_request_counter": 7,
      "_node_output_map": {
        "Welcome": [
          0
        ],
        "Greetings": [
          0
        ],
        "Is Special Occasion Question": [
          0
        ],
        "A Special Occasion": [
          0
        ],
        "node_22_1502215786664": [
          0
        ],
        "Goodbyes": [
          1
        ]
      },
      "branch_exited": true,
      "branch_exited_reason": "completed"
    }
  }
}
```


Plugging the Florence with Sample App

I made a Node.js app using IBM documentation repository on GitHub which demonstrates the Conversation service in a simple chat interface simulating a Flower Shop.

Before you begin

- Create a Bluemix account
    - Sign up in Bluemix, or use an existing account. Your account must have available space for at least 1 app and 1 service.
- Make sure that you have the following prerequisites installed:

- The Node.js runtime, including the package manager
- The Cloud Foundry command-line client

  Note: Ensure that you Cloud Foundry version is up to date

Installing locally

If you want to modify the app or use it as a basis for building your own app, install it locally. You can then deploy your modified version of the app to the Bluemix cloud.

Getting the files

Use GitHub to clone the repository locally, or download the .zip file of the repository and extract the files. **Link: https://github.com/arpitjainds/Florence-Chatbot**

Setting the Proxy environment variables

Use the following Commands if you are on the proxy network.

```
C:\Users\MJ\Downloads\chatbot\conversation-simple-master>set https_proxy=http://2015047:82564974@172.27.16.154:3128

C:\Users\MJ\Downloads\chatbot\conversation-simple-master>set http_proxy=http://2015047:82564974@172.27.16.154:3128
```

Setting up the Conversation service

You can use an existing instance of the Conversation service. Otherwise, follow these steps.

1. At the command line, go to the local project directory (Florence-Chatbot).
2. Connect to Bluemix with the Cloud Foundry command-line tool. Use one of these regions - eu-gb,ng,eu-de,su-syd,us-east

   ```
   cf login -a https://api.eu-gb.bluemix.net
   ```

3. Create an instance of the Conversation service in Bluemix.(If you already have an instance, don't follow this step) For example:

   ```
   cf create-service conversation free my-conversation-service
   ```

```
C:\Users\MJ\Downloads\chatbot\conversation-simple-master>cf login -a https://api.eu-gb.bluemix.net
API endpoint: https://api.eu-gb.bluemix.net

Email> arpit.jain@iiitdmj.ac.in

Password>
Authenticating...
OK

Targeted org arpit.jain@iiitdmj.ac.in

Targeted space dev



API endpoint:   https://api.eu-gb.bluemix.net (API version: 2.92.0)
User:           arpit.jain@iiitdmj.ac.in
Org:            arpit.jain@iiitdmj.ac.in
Space:          dev
```

Importing the Conversation workspace

1. In your browser, navigate to [your Bluemix console] (https://console.ng.bluemix.net/dashboard/services).

2. From the **All Items** tab, click the newly created Conversation service in the **Services** list.

3. On the Service Details page, click **Launch tool**.

4. Click the **Import workspace** icon in the Conversation service tool. Specify the location of the workspace JSON file in your local copy of the app project:

   <project_root>/training/Florence-Chatbot.json

5. Select **Everything (Intents, Entities, and Dialog)** and then click **Import**. The car dashboard workspace is created.

Configuring the app environment

1. Copy or rename the envexample.env file to .env (nothing before the dot). Create a service key in the following format:

   ```
   cf create-service-key <service_instance> <service_key>
   ```

   For example:

   ```
   cf create-service-key my-conversation-service myKey
   ```

2. Retrieve the credentials from the service key using the command cf service-key <service_instance> <service_key>. For example:

   ```
   cf service-key my-conversation-service myKey
   ```

   The output from this command is a JSON object, as in this example:

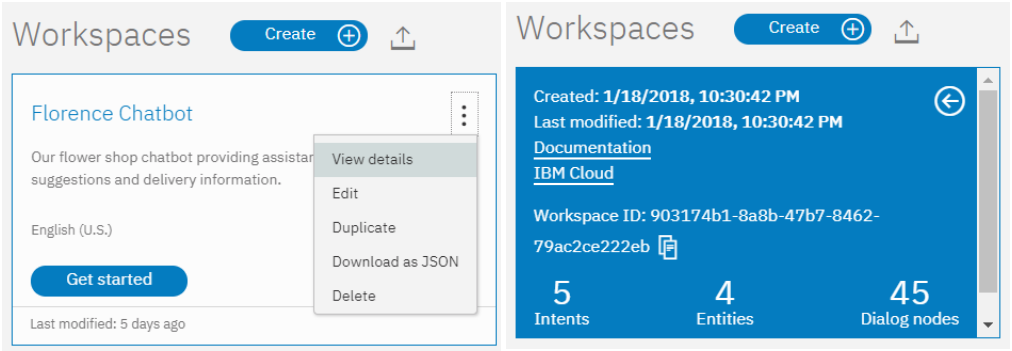
   ```
   {
    "password": "87iT7aqpvU7l",
    "url": "https://gateway.watsonplatform.net/conversation/api",
    "username": "ca2905e6-7b5d-4408-9192-e4d54d83e604"
   }
   ```

3. Paste the password and username values (without quotation marks) from the JSON into the CONVERSATION_PASSWORD and CONVERSATION_USERNAME variables in the .env file. For example:

   ```
   CONVERSATION_USERNAME=ca2905e6-7b5d-4408-9192-e4d54d83e604
   CONVERSATION_PASSWORD=87iT7aqpvU7l
   ```

4. In your Bluemix console, open the Conversation service instance where you imported the workspace.

5. Click the menu icon in the upper-right corner of the workspace tile, and then select **View details**.



6. Click the  icon to copy the workspace ID to the clipboard.

7. On the local system, paste the workspace ID into the WORKSPACE_ID variable in the. env file. Save and close the file.

Installing and starting the app

1. Install the demo app package into the local Node.js runtime environment:

```
npm install
```

2. Start the app:

```
npm start
```

3. Point your browser to http://localhost:3000 to try out the app.

Testing the app

After your app is installed and running, experiment with it to see how it responds.

The chat interface is on the left, and the JSON that the JavaScript code receives from the Conversation service is on the right. Your questions and commands are interpreted using a small set of sample data trained with the following intents:

```
buy flowers for some occasion
buy flowers for someone
suggest flowers
buy flowers
greetings
goodbyes
```

Type a request, such as **buy flowers** or **I want to buy some flowers for an anniversary**. The system understands your intent and responds. You can see the details of how your input was understood by examining the JSON data in the **Watson understands** section on the right side.
To see details of how these intents are defined, including sample input for each intent, launch the Conversation tool.

Modifying the app

After you have the app deployed and running, you can explore the source files and make changes. Try the following:

- Modify the .js files to change the app logic.
- Modify the .html file to change the appearance of the app page.

Deploying to Bluemix

You can use Cloud Foundry to deploy your local version of the app to Bluemix.

1. In the project root directory, open the **manifest.yml** file:

- In the **applications** section of the **manifest.yml** file, change the **name** value to a unique name for your version of the demo app.
- In the **services** section, specify the name of the Conversation service instance you created the demo app. If you do not remember the service name, use the **cf services** command to list all services you have created.

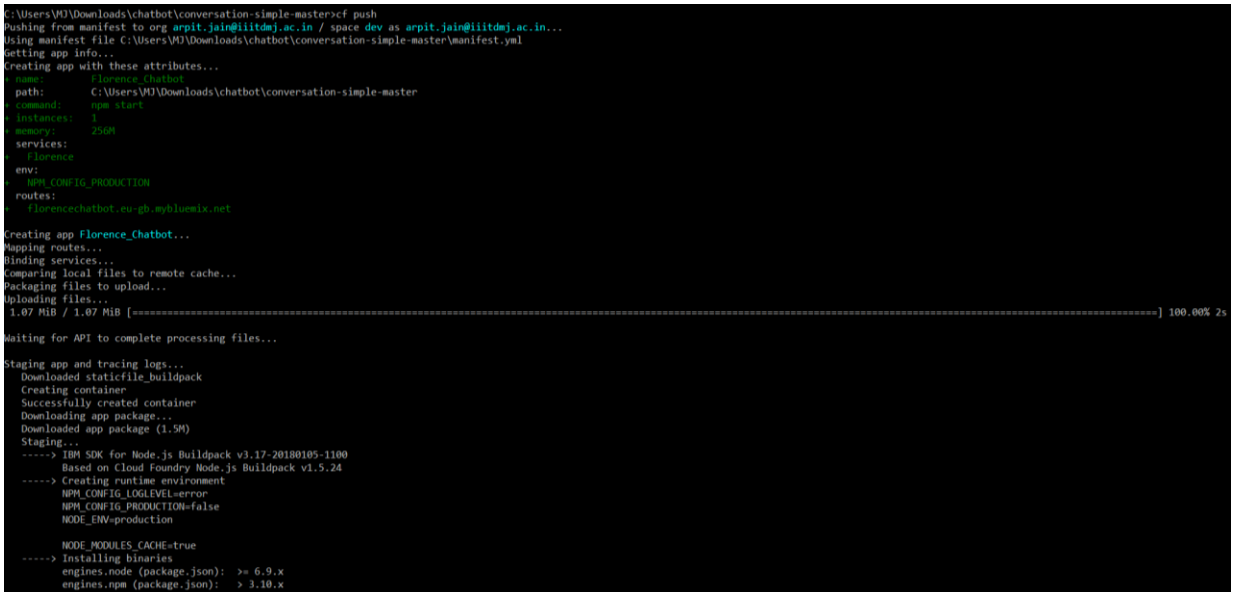
The following example shows a modified manifest.yml file:

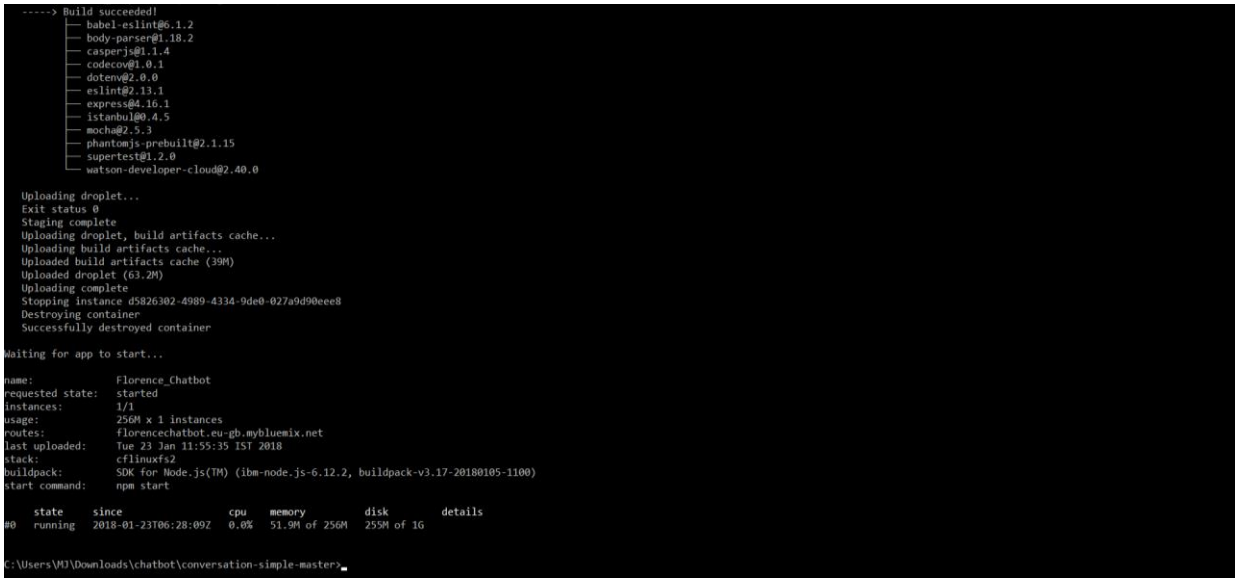
```
---
declared-services:
 conversation-service:
   label: conversation
   plan: free
applications:
- name: Florence_Chatbot
 command: npm start
 path:
 memory: 256M
 instances: 1
 services:
 - Florence
 env:
   NPM_CONFIG_PRODUCTION: false
```

1. Push the app to Bluemix:

```
cf push
```

Access your app on Bluemix at the URL specified in the command output.





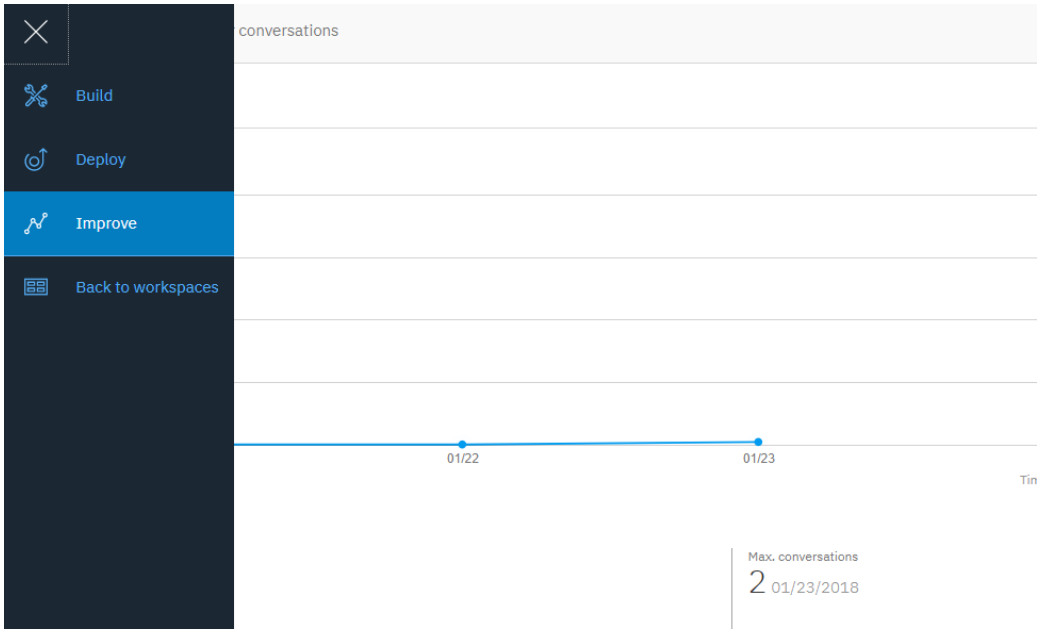
Troubleshooting

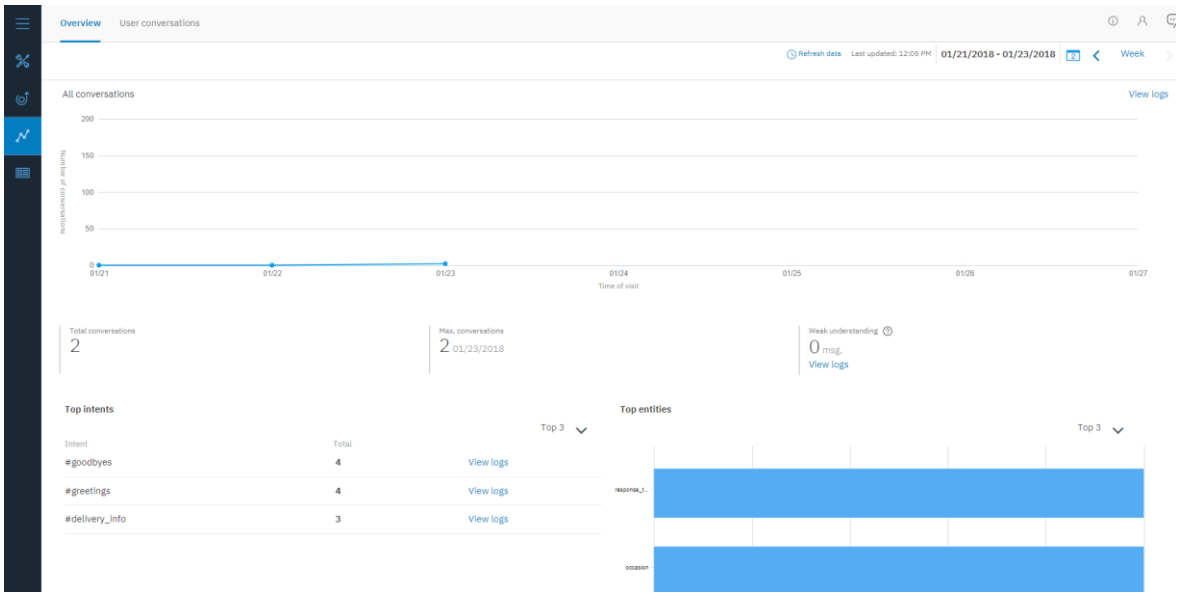
If you encounter a problem, you can check the logs for more information. To see the logs, run the following command:

```
cf logs <application-name> --recent
```

Monitoring the Chatbot

Chatbot can be monitored and trained further on the Improve panel of the IBM Conversation workspace.





Conclusion

We are in the middle of a technological revolution. Artificial intelligence (AI) has the potential to fundamentally transform the world around us -- the way we communicate, work, and conduct our daily lives. The interaction between machines and people will become more fluid and personalized as artificial intelligence becomes more advanced and adapts to individual needs. While the applications based on AI are already visible in healthcare diagnostics, targeted treatment, transportation, public safety, service robots, education and entertainment, it will inevitably gain prominence in other fields in the years ahead. Chatbots are a special application of AI which provides personal interaction with a human. The growth of Chatbots has increased due to the fact that they are extremely close to human understanding and thus provide Human-like conversations.