TRAVEL AGENCY CHATBOT

TEAM MEMBERS AND RESPONSIBILITIES:

NAME	COLLEGE	RESPONSIBILITY	DELIVERABLES	EMAIL ID
ARGHYA MAZUMDAR	N.I.T, ROURKELA	Android Studio Front-End Development with UI Design	Giving the User a highly interactive Interface.	arghyasls@gmail.com
ABHRANIL BHATTACHARJEE	JADAVPUR UNIVERSITY,KOLKATA	Integrating Watson Api with the Android Sdk	Making sure the interface is compatible with Android.	Abhra.tb@gmail.com
ROHIT KAR	I.I.T, BHILAI	Creation and Preparation of the Training Data	Adequate Training Data is Provided..	rohitk@iitbhilai.ac.in

ABSTRACT

From the mighty Himalayas to the greenery of Gangetic planes, from serene beaches to mangrove estuaries - West Bengal is a land of many natural splendours. However, it becomes very difficult to pick a destination for the purpose of travel owing to different requirements of the individual. We propose to create a travel app chatbot which will ask the individual's preferences, monetary constraints and number of days of visit and give a suitable suggestion based upon the training data we give it. We have collected the training data from various travel agencies nearby where we have gained a perspective of the type of questions one may ask and the ideal choices we should provide them with.

ACKNOWLEDGEMENT

We are indebted to the following resources and our project supervisor without whom this would not be possible :

- Prof. Samra Gadepalli, Cognitive Solutions Lead CBDS- Watson CoC.

UNDERLYING SOFTWARE/ HARDWARE REQUIRED FOR THE PRODUCT:

SOFTWARE REQUIREMENTS:

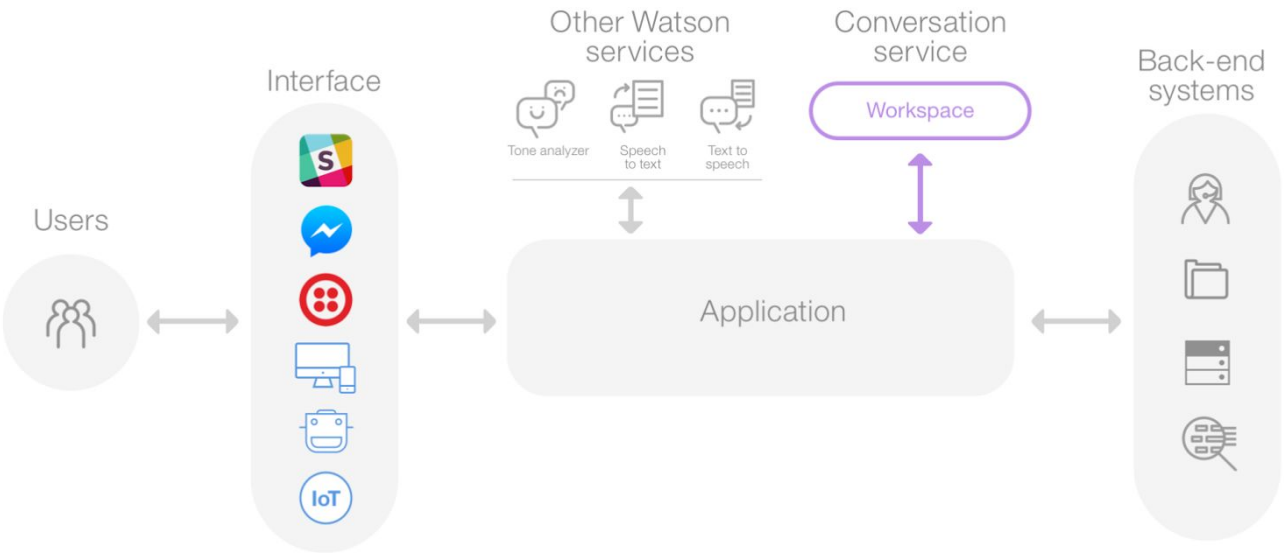
- ANDROID STUDIO
- IBM BLUEMIX JAVA(JAVA SE DEVELOPMENT KIT 8U151)
- JAVA(JAVA SE DEVELOPMENT KIT 8U151)
- ANDROID SDK

HARDWARE REQUIREMENTS:

- 8 GB RAM
- INTEL I5/17 PROCESSOR
- NVIDIA GTX/AMD GRAPHICS

- WINDOWS 7/10 OPERATING SYSTEM

ARCH DIAGRAM SHOWING THE WORKING OF THE SERVICES AND APPLICATION:



- Users interact with our application through the user **interface** that we implemented. For example, a simple chat window or a mobile app, or even a robot with a voice interfaces.
- The **application** sends the user input to the Conversation service. The application connects to a *workspace*, which is a container for your dialog flow and training data. The service interprets the user input, directs the flow of the conversation and gathers information that it needs.
- The application can interact with our **back-end systems** based on the user's intent and additional information. For example, answer question, open tickets, update account information, or place orders. There is no limit to what you can do.

THEORIES AND LOGIC:

There are different classes of Chatbots in the industry. The simplest kind are the purposeless mimicry agents which only provide the illusion of a conversation. These are based on deep learning sequence-to-sequence models. The second class comprises intention-based agents such as Apple's Siri. These agents have a simple understanding and can do real stuff. The Chatbot implemented in this project is an intention-based agent.

COGNITIVE SEARCH VS. TRADITIONAL SEARCH :

Traditional search uses Boolean queries, text relevance and other methods to get hits from search queries whereas in cognitive search we use machine learning techniques to do the same techniques. In cognitive search we train the engine on a few example queries and the engine learns from those queries. The engine gets better with every search as it learns from every query and gives better results. Cognitive search uses Natural Language Processing and machine learning to understand and organize data, predict the intent of the search query, improve relevance of data which eliminates the hassles of keyword based traditional search.

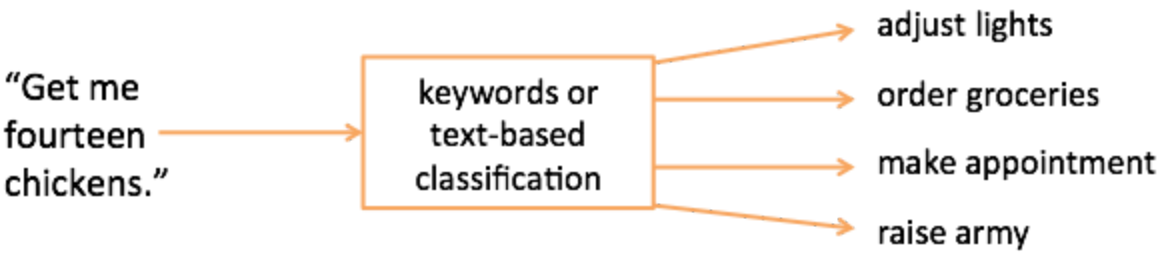
INTENTION BASED AGENTS

Intention-based agents understand language as commands and they use that understanding to perform tasks in the world. Understanding what we say as command language requires solving two problems:

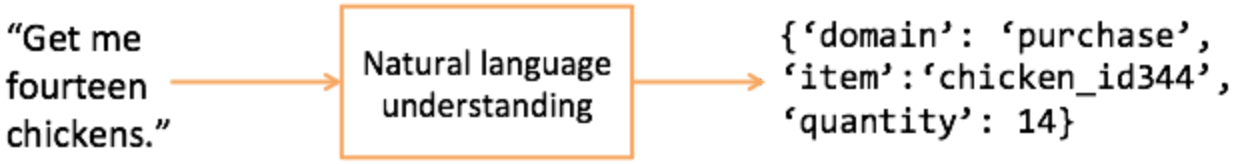
1. Identifying what the user wants the machine to do (the “intent”).
2. Figuring out the details of the intent so the machine can take action.

In our project we used sample “intents” - what the user is asking for – and used sample “dialogues” - the response that the bot should give – and trained the Watson Conversation api on these sample responses. Once the model is trained it can handle a variety of user inputs and not just the ones that are used in the sample intents.

We can explain intent determination using the following example.



When the user asks about chickens, the machine has to figure out which of its four capabilities matches the intent of the person. The assistant can determine the intent using either keywords or text-based classification. To use keywords, you simply associate words and phrases with intents. To do text-based classification, you label a bunch of statements with the correct intents and then train a classifier over them. The modeled is trained using convolutional neural network.
Once the agent has determined the intent, in our example it is to order groceries, it needs to convert the statement details into a machine-readable form, such as a Python dictionary.



Having the command in this kind of dictionary form is called a frame and slot semantics, where the frame is the topic and the slots are the individual features and values. Once we have the machine readable form, the computer can do with it what it wants, just like any other instruction. This is natural language understanding.

NATURAL LANGUAGE UNDERSTANDING

Natural Language Understanding is done using context-free grammars (compositional semantics). Context-free grammars consist of production rules that have a single nonterminal on the left and a string on the right side that can consist of terminals and nonterminals. To each of these production rules, we add a semantic attachment. An example set of rules is shown below.

```
$Order → $Purchase $ItemAmount, dunion(s[0],s[1])
$Purchase → is_purchase(tokens), {'domain': 'purchase'}
$ItemAmount → $Amount $Item, dunion(s[0],s[1])
$Amount → is_number(tokens), {'amount': get_number(tokens)}
$Item → is_item(tokens), {'item': get_item(tokens)}
```

Nonterminals begin with '$'. The first rule says that an order consists of an intent to purchase and an item and amount of that item. The semantics of that rule are combined via a dictionary union of the semantics of $Purchase (s[0]) and the semantics of $ItemAmount (s[1]). The next rule says that an intent to purchase is determined by the function is_purchase, as shown below. (This, of course, is a compact way to describe a set of rules. I.e., one with "get me" on the right-hand side, and another with "buy", and another with "grab me some.")

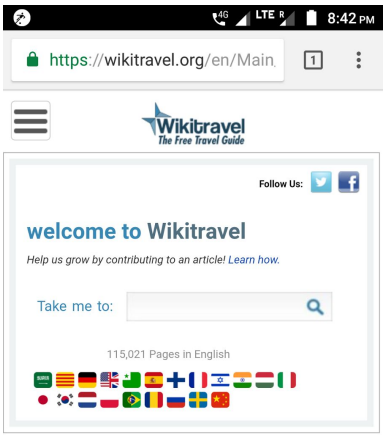
The next rule says that $ItemAmount consists of an $Amount and an $Item, which are defined in the following two rules. Each of those rules has an associated semantics at the end. We use parsing to apply those production rules to text to get semantics. Parsing is done using bottom-up dynamic programming. Dynamic programming consists of solving sub problems and then reusing those solutions over and over again. The parsing process builds up the parse by looking at pairs of words, and for each pair, it loops over all *n* possible words in between. Because the parsing algorithm deals with pairs of sub-parses, the production rules above each have either a single terminal or two non-terminals on the right-hand side.

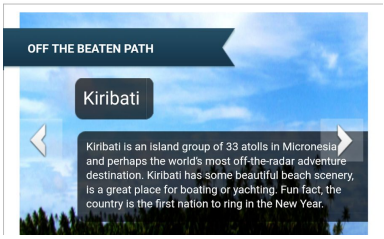
LIBRARIES AND API'S USED:

- *WATSON CONVERSATION API'S:* With the IBM Watson™ Conversation service, you can build a solution that understands natural-language input and uses machine learning to respond to customers in a way that simulates a conversation between humans.
- *ANDROID STUDIO AND SDK TOOLS:* Android Studio provides the fastest tools for building apps on every type of Android device.

DATA SOURCE USED:

The data source used for displaying the description, mode of transport and hotels to a place have been scrapped from wikitravel.org using JSoup, i.e Java Based Web Scraping Tool. This ensures that as people visit the website and update their reviews of the place on the web page the data will be automatically updated in our application. Thus in this manner , the dynamic nature of the dataset used is beneficial to the working of the application.





DEPLOYMENT:

The interfacing has been done in the form of an app using java in android studio.

Front End

For the UI/UX design of the app, three xml files have been added:

activity_main: Contains the List View to display all the chat bubbles and the enter text window at the bottom of the screen in which the user enters his/her query.

item_mine_message: Contains the chat bubble of the user's input.The himanshusoni package of chat bubbles have been used for this alongside with a message icon as an image view.

item_other_message:Contains the chat bubble of the bot output.The himanshusoni package of chat bubbles have been used for this alongside with abot icon drawn in vector graphics as an image view.

Back End

The back end has three files namely:

ChatMessage.java: This contains the getter and setter method of the chat bubbles.

ChatMessageAdapter.java:This is an adapter which links the chat bubbles to the List View of the UI design. The class checks whether the message is an user input or a bot generated output and depending on the result gives the proper of orientation of display.

MainActivity.java: This is the class where the main activity starts. A conversation Service has been created using the user id and password in the bluemix account. This conversation service is then given an input by the user in the form of a message request. The response to this request is then displayed using the OnResponse() method. The context of the response is stored in a list and this context is then fed into further requests so that Watson does not have to start the conversation from start. The conversation is initialized with a blank input and a null context.

The build grade of the app has used the com.ibm.watson.developer_cloud:java-sdk:3.7.2 package for the integration.

The entire code of the project can be found at link**:**

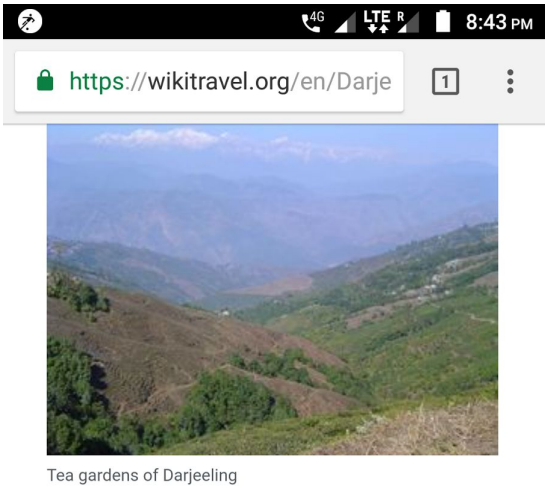
https://github.com/arghyasls/ibm_travelbot_conversation

The apk file can be found here:

https://drive.google.com/open?id=12SUIm4Uz8nBUEPlNt0D5yHU9TCj-sH9E

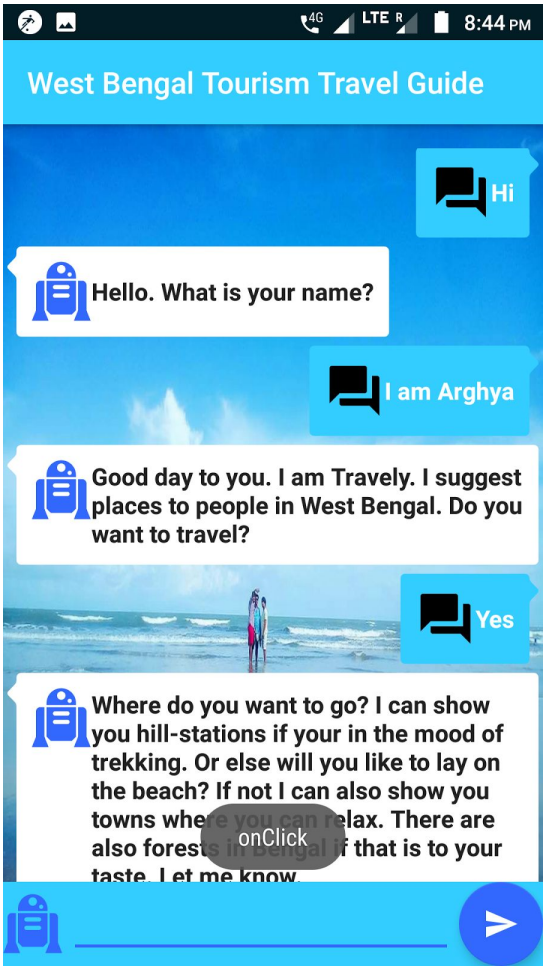
WORKING MODEL:

Some screenshots of the app are shared below for reference: The wikitravel.org page is being scraped.



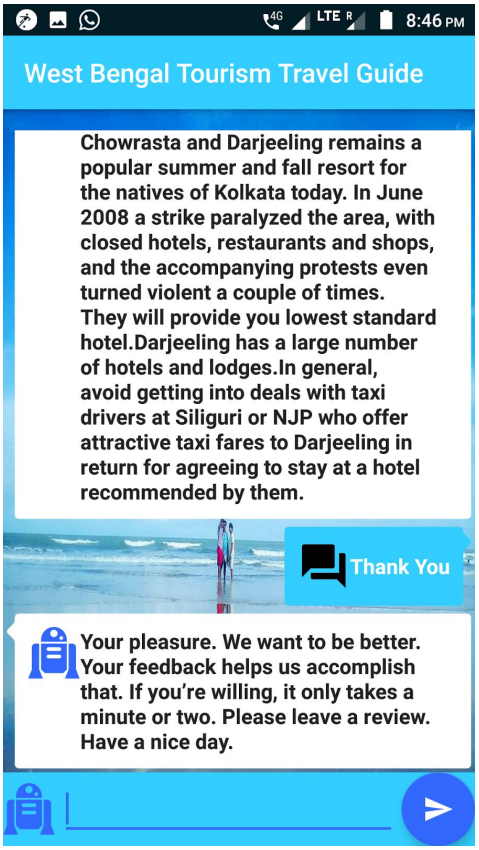
Tea gardens of Darjeeling

Originally just a cluster of villages that was administered intermittently by Nepal and Sikkim, Darjeeling grew in prominence during the mid 19th century when, because of its climate, the British first established a hill station there after leasing it from the Chogyal of Sikkim and later discovered that the area was particularly well suited for tea plantations. In 1849, the British annexed the area and Darjeeling became a part of British India. The Darjeeling Himalayan Railway was opened in 1881 (it is now a UNESCO World Heritage Site) and the town became the de-facto summer capital of India during the days when the Raj was governed from Calcutta.




8:45 PM
West Bengal Tourism Travel Guide
Show me a hill station
You can visit the tea gardens of Darjeeling. You can climb the hills of Kalimpong. You can see the grandeur of Kurseong Mirik and Siliguri. Let me know which place do you want to visit.
Tell me about Darjeeling
Darjeeling is a small city in West Bengal, India.Originally just a cluster of villages that was administered intermittently by Nepal and Sikkim, Darjeeling grew in prominence during the mid 19th century when, because of its climate, the British first established a hill station there after leasing it from the Chogyal of Sikkim and later discovered that the area was particularly well suited for tea plantations. In 1849, the
onClick


8:46 PM
West Bengal Tourism Travel Guide
how do I go there
The Darjeeling Himalayan Railway was opened in 1881 (it is now a UNESCO World Heritage Site) and the town became the de-facto summer capital of India during the days when the Raj was governed from Calcutta.5 hours by road from Darjeeling and 2 hours by air from Delhi, and 50 minutes by air from Kolkata. One can catch a share jeep to Siliguri and a bus to the Nepalese border, changing buses in Nepal.
suggest me a hotel
Because it was a popular hill station during the days of the Raj, a lovely Victorian town was built among the Himalayan foothills, the remnants of which are still visible around the



SOURCES AND REFERENCES:

BOOKS:

- NATURAL LANGUAGE PROCESSING WITH PYTHON by Steven Bird, Ewan Klein and Edward Loper.
- SPEECH AND LANGUAGE PROCESSING by Daniel Jurafsky and James Martin.
- DEEP LEARNING by Andrew Ng (2018).

ONLINE SOURCES:

- The Watson Conversation API and its documentation.
- IBM Bluemix cloud platform and its documentation.
- Stackoverflow.com
- Youtube.com
- Tutorialspoint.com