Chatbot to Replace Work of Helpdesk

Chandu Sanjith. T, Pooja H, Nivedita S S, Nikitha V P
Human-to-Machine Conversation Modeling

***Abstract* - A chatbot aims to make a conversation between both human and machine. The machine has been embedded knowledge to identify the sentences and making a decision itself as response to answer a question. The response principle is matching the input sentence from user. From input sentence, it will be scored to get the similarity of sentences, the higher score obtained the more similar of reference sentences. The sentence similarity calculation in this paper using bigram which divides input sentence as two letters of input sentence. We mainly use this chat bot to answer the questions that re asked at the helpdesk. The knowledge of chatbot are stored in the database. The chatbot consists of core and interface that is accessing that core in relational database management systems (RDBMS). The database has been employed as knowledge storage and interpreter has been employed as stored programs of function and procedure sets for pattern-matching requirement. The interface is standalone which has been built using programing language of Pascal and Java.**

Keywords - Bigram; chatbot; database; sentence; similarity

I. INTRODUCTION

The development of the information technology and communication has been complex in implementing of artificial intelligent systems. The systems are approaching of human activities such as decision support systems, robotics, natural language processing, expert systems, etc. Even in the artificial intelligent fields, there are some hybrid methods and adaptive methods those make more complex methods. Not only that, but nowadays there is also a hybrid of natural language and intelligent systems those could understand human natural language. These systems can learn themselves and renew their knowledge by reading all electronics articles those has been existed on the internet. Human as user can ask to the systems like usually did to other human. These systems are often known as internet answering-engines.

In addition the internet answering, currently in the internet also begins many applications of chatter-boot or known as chatbot
which is often aimed for such purposes or just entertainment [1]. This application work is very simpler because the knowledge already programmed in advance [2]. One of methods used in this application is to match the pattern (pattern-matching) [3]. The chatbot would match the input sentence from the speaker or user with pattern that has existed on the knowledge. Each pattern paired with the knowledge of chatbot which taken from various sources. The input sentence prepared as the materials of chat pattern [4]. The chat patterns modeled in the pattern-template stored in a relational database management system (RDBMS) tables. The process of pattern matching is using a sentence similarity measurement scores. The calculation method to achieve the scores of sentence-similarity measurement may apply bigram method as one way of measurement methods, although there are some other methods. The function programs for pattern matching and other support purposes written as program stored in the RDBMS. Other knowledge storage method of chatbot is artificial intelligence markup language (AIML) [5,6].

The AIML has modularly knowledge processes. This system is a web service-based which could be accessed by client. The chat patterns are language knowledge in the format of AIML stored in the database. This system could be added a specific knowledge modules [7,8].

Chatbot can also be developed using IBM Watson and Google dialogue flow. They are the sevices by google and IBM respectively where user can use these services to develop the chatbots based on there quirements.

II. RELATED WORK

Current chatbots use a variety of methods to generate responses, such as machine translation (Ritter et al., 2011), retrieval based response selection (Banchs and Li, 2012), and recurrent neural network sequence generation (Vinyals and Le, 2015). Yet, the databases they use to power their systems have very little variability. Some systems used micro-blogs, such as Twitter conversations (Ritter et al., 2011) and some used movie subtitles (Banchs and Kim, 2014;Ameixaetal.,2014;BanchsandLi,2012),and there is research that used Twitter as a database but switched to ask the human to generate responses in the crowdsourcing platform in real time when the database failed to have an appropriate response (Bessho et al., 2012). Most of the wor kreported above have no real user evaluation or a small group of people for evaluation. Only two kinds of databases have been used, movie subtitles and micro-blogs. In this work, we focus on how to generate appropriate databases for chatbots and conduct evaluations for chatbots by leveraging crowdsourcing resources.

III. METHODOLOGY

Bots are rapidly becoming a de facto interface for interacting with software services. This is due partly to the widespread adoption of messaging platforms (for example, Facebook Messenger for social networking and Slack for developer communication), and partly to the advancement of natural-language processing, which many bots leverage. But another driver is the prevalence of big data, along with machine-learning algorithms for analyzing data across many domains. Bots provide a convenient way for developers to generate a UI for interacting with these algorithms and data. Major software companies are recognizing the value bots bring in terms of integrating services, users, and communication channels. Facebook aims to “replace apps” one bot at a time in its messaging platform,2 while Microsoft claims that “conversation as a platform” is the OS of the future.3 Alexa, Siri, IBM Watson, and Google Now all support this shift toward bots. There are also many bots in the platforms software developers use to connect with other developers and services, such as Slack, Microsoft Teams, and HipChat. The transition from commandline interfaces to interacting with bots through messaging tools feels intuitive to most developers. We see new examples of sophisticated and innovative bots stemming from developer’s needs;4 these examples are paving the way for bots in other domains. This situation is inspiring the development of bots for users, who are spending increased time in messaging applications and are embracing bots as an alternative to installing and relying on external apps. However, bot developers must carefully consider not only where to host bots and how to create them but also when not to use them.

A. Creating and Hosting Bots

Although simple bots can be built from scratch and self-hosted, many developers leverage third-party frameworks to streamline creation and distribution. With the explosion of new tools for bot development, we need to distinguish between the tools for building bots (creation platforms) and the platforms on which the bots dwell (distribution platforms).5 Companies such as Microsoft and Facebook offer comprehensive tooling to support bot creation and distribution. Other companies provide customized resources for specific creation and distribution tasks. Table 1 lists some common bot platforms and the creation and distribution services they use, along with other bot technologies.

1. Creation Platforms

Creation platforms provide a variety of software foundations, frameworks, toolkits, APIs, and other advanced features (for example, naturallanguage processing, search, and image processing). These platforms can be distribution-platform-specific or produce bots that are deployable across multiple platforms, such as the Pandorabots. The provided services range from documentation and code templates to no-code-required bot-building interfaces such as Chatfuel. Many popular creation platforms also have vibrant developer ecosystems— developers can connect with these online communities to obtain expertise in the form of tutorials, articles, discussions, and support. Other general bot development communities, such as Botmaker’s Slack group and the Chatbot Magazine community, are hotbeds for discussions on a variety of bot-related topics.

2. AIML

AI is one of the biggest components of chatbot development. In order to make AI useful for your users, it’s important to define the correct flow for your chatbot. AIML or Artificial Intelligence Markup Language is a XML-complaint language designed for creating AI flows that is platform agnostic and portable. Moving towards AIML frees the tight integration between UI Builders, Mockup tools and chatbot platforms opening up choices for developers. This does not lock a developer down to a proprietary syntax and allows a developer to use any mockup tool of choice and allows vendors to work together more closely to bring the best possible solution to build great chatbots.

2.1 Sample AIML code

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aiml>
   <category>
      <pattern>START</pattern>
      <template>
         Hi <bot name="first_name"/>! I am a demo bot and I do awesome things.
      </template>
    </category>
    <category>
      <pattern>HOW ARE YOU</pattern>
      <template>
         I am just a chatbot and I am doing great. Test me often to improve!
      </template>
    </category>
   <category><pattern>*</pattern><template>Sorry I didn't get it =(.</template></category>
</aiml>
```

Here instead of writing the flow in the code, AIML helps me separate the conversation logic and yields a much cleaner solution. If you are into building bots from scratch this is something to consider. However, it is something to keep in mind that in order to build a complex bot or if the bot is the face of your customer service, use the built-in AI features more than going to the DIY route unless you are totally a pro in AIML.

3. Google dialogueflow

Give users new ways to interact with your product by building engaging voice and text-based conversational interfaces powered by AI. Connect with users on the Google Assistant, Amazon Alexa, Facebook Messenger, and other popular platforms and devices.

3.1Getting started with Dialogflow fulfillment

Enable fulfillment for default intents

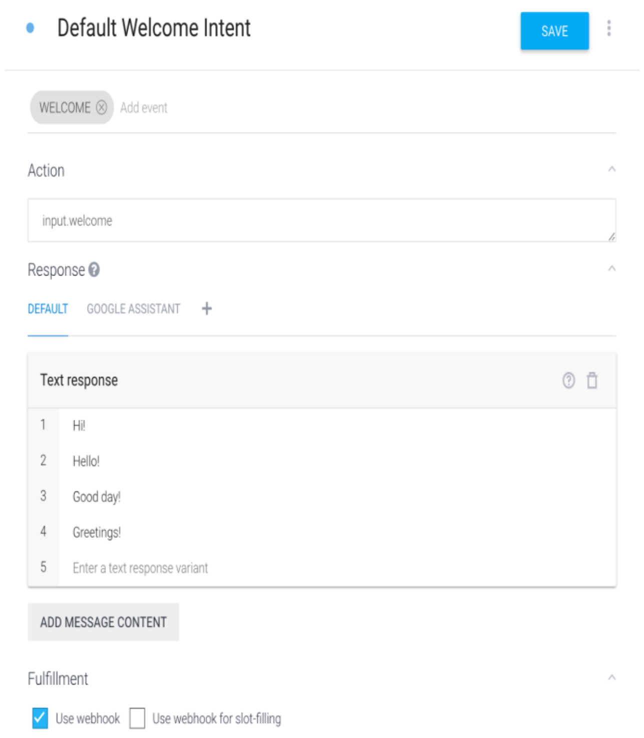
1. Click on **Intents** in the left menu.

2. Click on **Default Fallback Intent**.

3. At the bottom of the page, click **Fulfillment** to reveal the options.

4. Check **Use webhook**.

5. Click the **Save** button.


Default Welcome Intent

Action
input.welcome
Response

Text response
1 Hi!
2 Hello!
3 Good day!
4 Greetings!
5 Enter a text response variant

Fulfillment
✓ Use webhook ☐ Use webhook for slot-filling

The sample code handles the two default intents included when you create an agent, and responds to the user via the Cloud Function for Firebase fulfillment.

If the Default Welcome Intent is matched, the code defined on line 31 (for queries via Google Assistant) or 33 (all other queries) is used. If the Default Fallback Intent is matched, code on line 40 (for queries via Google Assistant) or 42 (all other queries) is used.

3.2 Actions

To make use of the last response, you will need to setup a new intent.

1.Click on the plus icon next to **Intents** in the left menu.

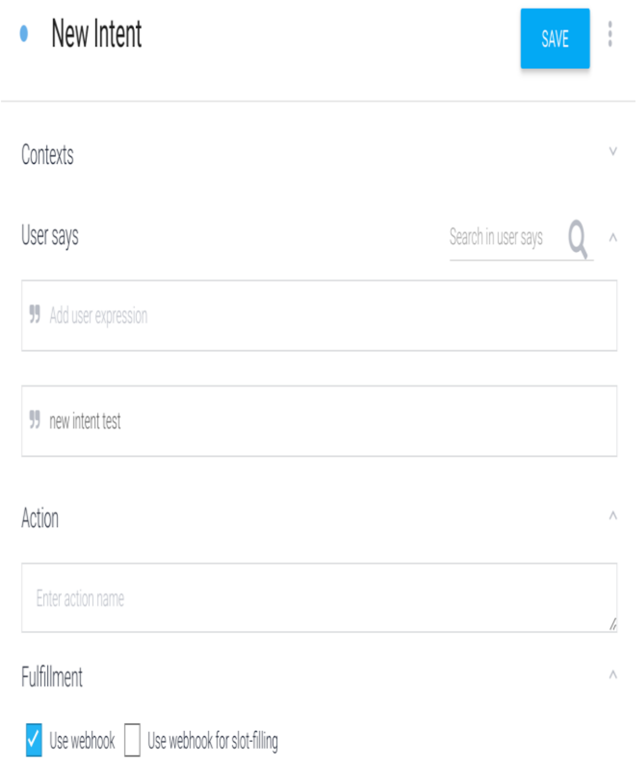
2. Name the intent whatever you want.

3. Enter the following **Training Phrase**: new intent test

S 4.scroll down and click on **Fulfillment**.

5. Check the **Use webhook** option.

6. Click the **Save** button.


New Intent

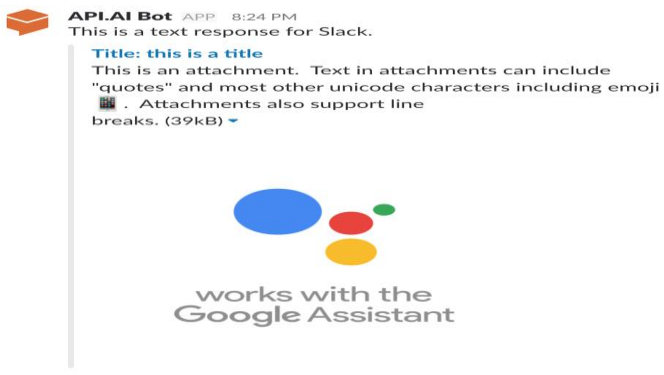
Contexts
User says

Add user expression
new intent test
Action
Enter action name
Fulfillment
✓ Use webhook ☐ Use webhook for slot-filling

A

fter your agent is finished training (the settings gear will stop spinning), type new intent test in the simulator on the right, and hit enter.

3.3 Rich responses


API.AI Bot APP 8:24 PM
This is a text response for Slack.
Title: this is a title
This is an attachment. Text in attachments can include "quotes" and most other unicode characters including emoji . Attachments also support line breaks. (39kB)

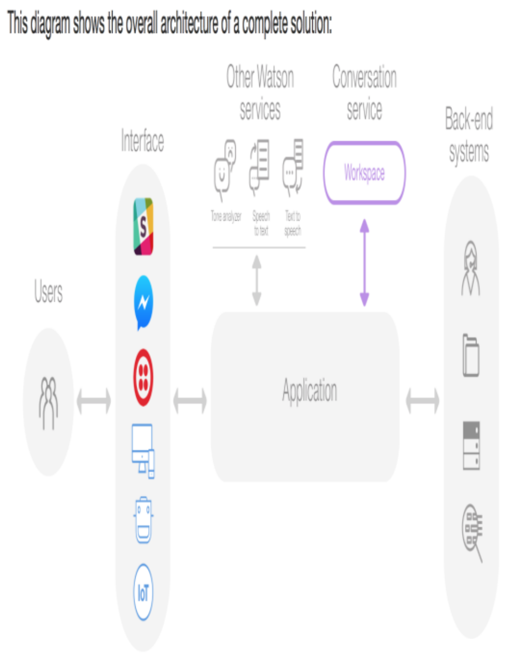
Contexts and parameters

To apply outgoing contexts to your webhook requests, uncomment lines 51 and 59 and redeploy the function. Now your function will apply an outgoing context of weather with a lifespan of 2 and a city parameter with the value of Rome to any matched intents with the action default.

4. IBM Watson

Creation of Watson Assistant (formerly Conversation) Service

Watson Assistant (formerly Conversation) combines a number of cognitive techniques to help you build and train a bot - defining intents and entities and crafting dialog to simulate Watson Assistant (formerly Conversation).



You use the Watson Assistant (formerly Conversation) tool to create workspaces by either creating a new workspace from scratch, or by importing a workspace from a sample JSON file. You can also duplicate an existing workspace within the same service instance.

1. If the Service Details page is not already open, click your Watson Assistant (formerly Conversation) service instance on the IBM Cloud console. (When you create a service instance, the Service Details page displays.)

2. On the "Service Details" page, scroll down to **Watson Assistant (formerly Conversation) tooling** and click **Launch tool**.

3. Click **Create** to create a new workspace.

4. Specify the details for the new workspace:

○ **Name**: A name no more than 64 characters in length. This value is required.

○ **Description**: A description no more than 128 characters in length.

○ **Language**: The language of the user input the workspace will be trained to understand. The service supports Brazilian Portuguese, English, French, Italian, and Spanish.

5. Click **Create**. The new workspace is created and now appears as a tile on the Workspaces page.

4.1 Creating an intent

You use the Watson Assistant (formerly Conversation) tool to create intents. The number of intents and examples you can create in a single service instance depends on your Watson Assistant (formerly Conversation) service plan:
Create some intents.

1. In the Watson Assistant (formerly Conversation) tool, open your workspace and then click the **Intents** tab.

2. Click **Create new**.

3. In the Intent name field, type a descriptive name for the intent. The intent name can contain letters (in Unicode), numbers, underscores, hyphens, and dots. Intent names cannot contain spaces and must not exceed 128 characters. The following are examples of intent names:

○ #weather_conditions
○ #pay_bill
○ #escalate_to_agent

Tip: Don't include the # character in the intent names when you create them in the Watson Assistant (formerly Conversation) tool.

4. In the **User example** field, type the text of a user example for the intent. An example can be any string up to 1024 characters in length. The following might be examples for the #pay_bill intent:

○ I need to pay my bill.

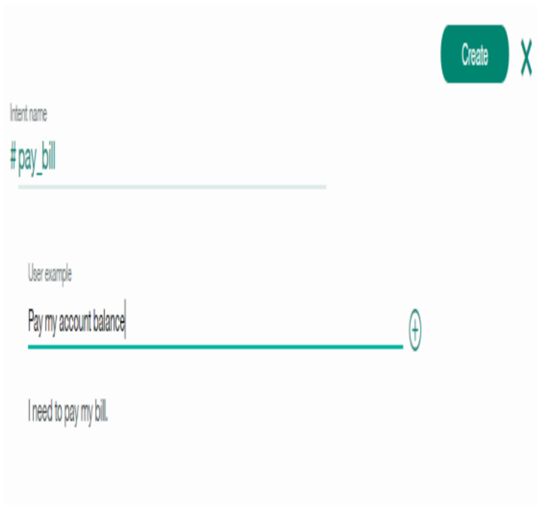
○ Pay my account balance

○ make a payment

Important: Intent names and example text can be exposed in URLs when an application interacts with the service. Do not include sensitive or personal information in these artifacts.

Press Enter or click **+** to save the example.

5. Repeat the same process to add more examples.
Provide at least 5 examples for each intent. The more examples you provide, the more accurate your application can be.



6. When you have finished adding examples, click **Create** to finish creating the intent.

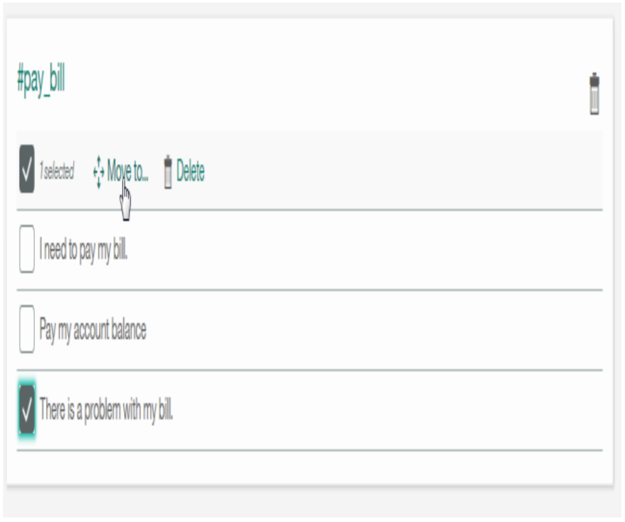
4.2 Results

The intent you created is added to the Intents tab, and the system begins to train itself on the new data.

You can click any intent in the list to open it for editing. You can make the following changes:

- Rename the intent.
- Delete the intent.
- Add, edit, or delete examples.
- Move an example to a different intent.

To move an example, select the example by clicking the check box and then click **Move to**.

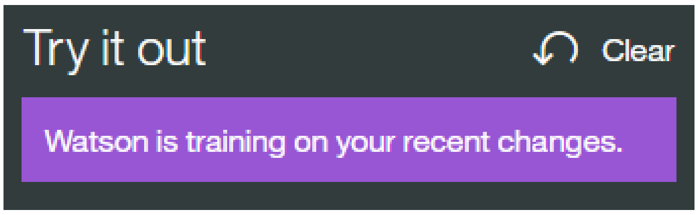


4.3 Testing your intents

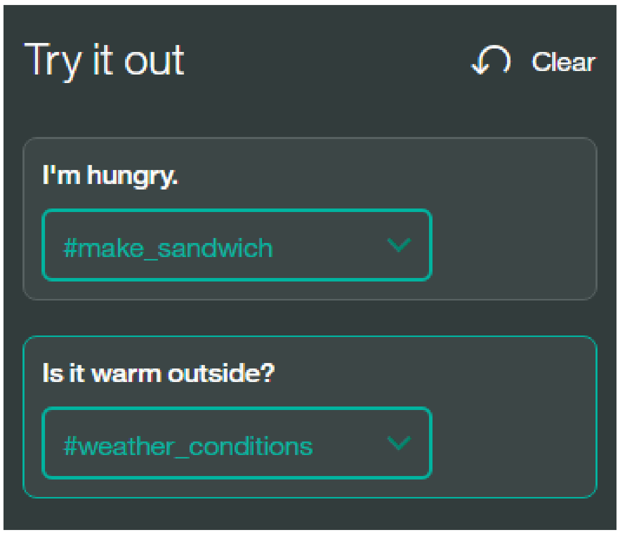
After you have finished creating new intents, you can test the system to see if it recognizes your intents as you expect.

1. In the Watson Assistant (formerly Conversation) tool, click the icon.

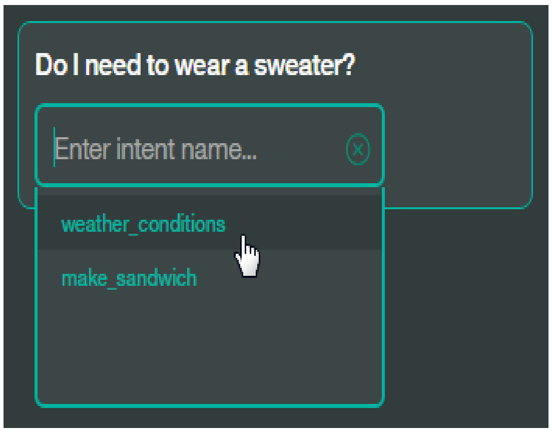
2. In the Try it out panel, enter a question or other text string and press Enter to see which intent is recognized. If the wrong intent is recognized, you can improve your model by adding this text as an example to the correct intent.**Tip**: If you have recently made changes in your workspace, you might see a message indicating that the system is still retraining. If you see this message, wait until training completes before testing:



The response indicates which intent was recognized from your input.



If the system did not recognize the correct intent, you can correct it. To correct the recognized intent, click the displayed intent and then select the correct intent from the list. After your correction is submitted, the system automatically retrains itself to incorporate the new data.



If your intents are not being correctly recognized, consider making the following kinds of changes:

o Add the unrecognized text as an example to the correct intent.

o Move existing examples from one intent to another.

o Consider whether your intents are too similar, and redefine them as appropriate.

4.4 Creating an entity

You use the Watson Assistant (formerly Conversation) tool to create entities. The number of entities, entity values, and synonyms you can create in a single service instance depends on your Watson Assistant (formerly Conversation) service plan:

1. In the Watson Assistant (formerly Conversation) tool, open your workspace and then click the **Entities** tab.

2. Click **Create new**.

3. In the **Add the entity name** field, type a descriptive name for the entity.The entity name can contain letters (in Unicode), numbers, underscores, and hyphens. For example:

o @location
o @menu_item
o @product

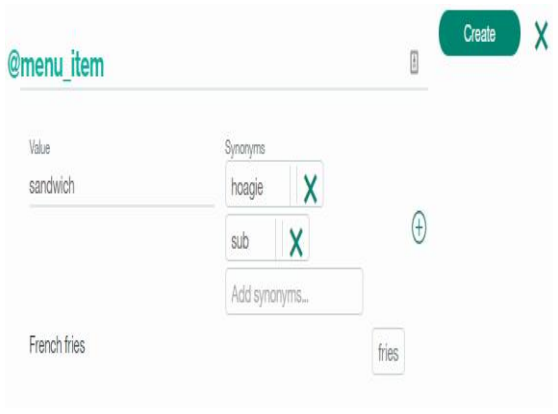
Tips:

o Don't include the @ character in the entity names when you create them in the Watson Assistant (formerly Conversation) tool.

o Entity names can't contain spaces or be longer than 64 characters. And entity names can't begin with the string sys-, which is reserved for system entities.

4. In the **Value** field, type the text of a possible value for the intent. An entity value can be any string up to 64 characters in length.**Important**: Don't include sensitive or personal information in entity names or values. The names and values can be exposed in URLs in an app.

5. In the **Synonyms** field, type any synonyms for the entity value. A synonym can be any string up to 64 characters in length. Press Enter to save each synonym.


@menu_item

Create

Value: sandwich
Synonyms: hoagie, sub
Add synonyms...

French fries
fries

6. Click + and repeat the process to add more entity values.

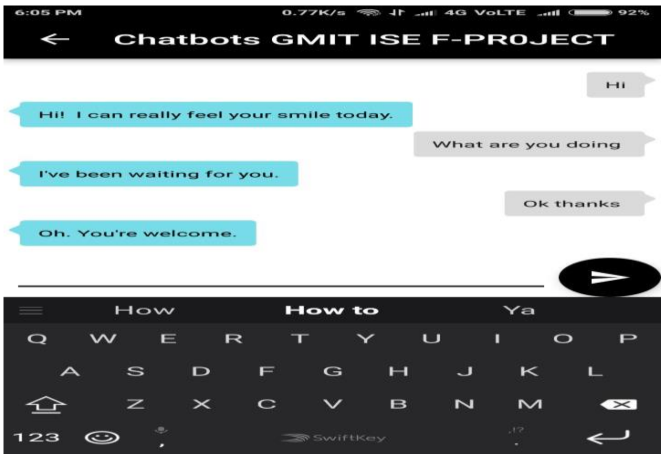
7. When you are finished adding values and synonyms, click **Create**.

4.5 Building a Dialog

The dialog component of the Watson Assistant (formerly Conversation) service uses the intents and entities that are identified in the user's input to gather required information and provide a useful response. Your dialog is represented graphically as a tree; create a branch to process each intent that you define.

IV. Testing The Bots

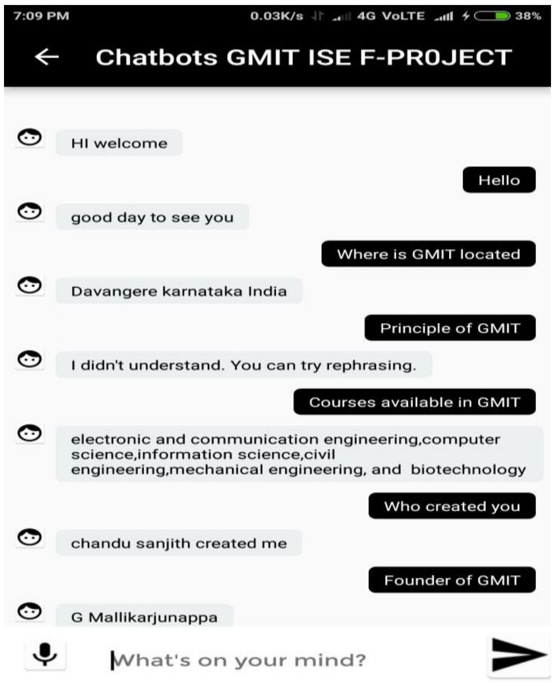
Once we successfully deploy these bots, Here we are deploying them in an android Application.


6:05 PM 0.77K/s 4G VoLTE 92%

Chatbots GMIT ISE F-PROJECT

Hi
Hi! I can really feel your smile today.
What are you doing
I've been waiting for you.
Ok thanks
Oh. You're welcome.

So here the Interface is created for Pandorabot which was Created using AIML.The AIML files will be presnt in the Local storage of the divice.


7:09 PM 0.03K/s 4G VoLTE 38%

Chatbots GMIT ISE F-PR0JECT

HI welcome
Hello
good day to see you
Where is GMIT located
Davangere karnataka India
Principle of GMIT
I didn't understand. You can try rephrasing.
Courses available in GMIT
electronic and communication engineering,computer science,information science,civil engineering,mechanical engineering, and biotechnology
Who created you
chandu sanjith created me
Founder of GMIT
G Mallikarjunappa

What's on your mind?

So here the IBM Watson service is working fine as it is replying the user about the college information so here we can say that out Chatbot has successfully replaced the work of an help desk.

V.CONCLUSION

Chatbots are low in cost compared to websites and they tend to train themselves so the maintenance becomes low Watson is an exceptional question answering technology that leverages the power of cognitive computing. It encompasses many different innovations in the field of natural language processing and question answering system. However, training IBM Watson could be tedious and if it is not trained properly then the answers may not be accurate.
Bots are rapidly becoming pervasive: we interact with them in cars, at home, in entertainment devices, and at work. And, as we discuss in the sidebar, bots play a sophisticated and increasingly significant role in software development projects. We need to learn from these early adoption experiences to find out not only what works well but also what might go wrong. Here are insights we've gained from our research that developers should consider when creating and using bots.
In feature we can use these chatbots in medical fields as of now IBM Watson medical assist is currently being developed. Chatbots.

REFERENCES

[1] R. High, Jr., "IBM Watson, How Watson Works", power point presentation, March 21, 2015 .
[2] IBM Guide, "IBM Watson Ecosystem Getting Started Guide", V 1.7, July 2014.
[3] IBM White Paper, "IBM Systems and Technology. Watson – A System Designed for Answers: the Future of Workload Optimized Systems Design." February 2011. [Online]: http://public.dhe.ibm.com/common/ssi/ecm/en/pow03061usen/ POW03061USEN.PDF.
[4] Watbot by IBM watson: https://github.com/watbot.
[5] A. Lally, J. M. Prager, M.C. McCord, B. K. Boguraev, S. Patwardhan, J. Fan, P. Fodor, and J. Chu-Carroll, "Question Analysis: How Watson Reads a Clue." IBM Journal of Research and Development, Vol. 56, No. 3.4, 2012, pp. 2:1— 2-14.
[6] B. Anbarasan, "Using the Computer Brain Cognitive Computing" Special Issue (NCRICA-14) of International Journal of Emerging Technologies in Computational and Applied Sciences, Vol. 7, No. 1, 2014, pp. 22-26.
[7] D. A. Ferrucci, "Introduction to "This is Watson"." IBM Journal of Research and Development, Vol. 56, No. 3.4, 2012, pp. 1:1— 1:15.
[8] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, N. Schlaefer, C. Welty, "Building Watson: An Overview of the DeepQA Project." AI Magazine, Vol. 31, No. 3, 2010, pp. 59-79.