Amica (The Chatterbot)

Shubham Gaba[1], Ms. Vandana Choudhary[2]
[1] Student, Dept of Information technology
[2]Assistant Professor, Dept of Information technology
[1, 2] Maharaja Agrasen Institute of Technology,Delhi

Abstract- *In this demo, we present the AMICA, a information-graph-driven chatbot outlined to optimize community interaction. The bot was planned for integration into community computer program to encourage the replying of repetitive questions. Four primary challenges were tended to when building the chatbot, to be specific (1) understanding client questions, (2) getting pertinent data based on the queries, (3) fitting the reactions based on the measures of each yield stage as well as (4) creating sub- sequent client intelligent with the AMICA.*

I. INTRODUCTION

There's a developing fervor around chatbots. From tech giants' center items such as Apple Siri, Amazon Alexa, to various startup companies, numerous are compelled by the thought of progresses in fake insights combined with a characteristic frame of interactions. Be that as it may, sometime recently this wave of promoting buildup, inquire about on chatbots has come a long way within the past half century.Two focuses of feedback have been frequently raised for considers of chatbots. One may be a need of understanding on real-life client experience and consideration to the hole between client intuitive within the lab and those within the wild] Another point is the center on barely compelled specialist started discussions for the errand spaces, which provides little data approximately client interface in speaking with chatbots for future framework development.

AMICA is actually an Italian word meaning friend.It is python based project . It is unlike other in the sense that it is not a specific chatbot designed for a particular company rather it is trainable chatbot which can cater to the needs of any type of company requiring it's services be it government organization or e-commerce website. Previously existing chatbots were usually built using js and if extra queries were required to be in the chatbots that were being frequently asked the programmer was required to make the altercations but in case AMICA any non-programmer can feed the particular query to the AMICA using two lines of code that are required to just activate the trainer. It can also maintain a count variable to provide information about the queries that are being frequently asked by the users as it flags the particular query so it answer can be feeded to the trainer.

II. RELATED WORK

One of the early chatbots was ELIZA , which was planned around a design pattern matching approach. ELIZA was expecting to sound like diverse people, e.g. like a psychotherapist. Afterward, the ELIZA- propelled chatbot ALICE was presented. ALICE was implemented based on a scripting dialect called AIML2, which is following to RiveScript3, which is as of now among the foremost utilized dialects for chatbots. More laterchatbots, such as Rose, combine chat scripts with address understanding to superior imitate space specialists. Later approaches have moreover considered handling the combination of RDF and chatbots.

III. APPROACH

Our architecture is based on a modular approach to account for distinctive sorts of client inquiries and reactions seen . The AMICA is competent of reacting to clients by means of (1) basic brief content messages or (2) through more expand intuitively messages utilizing the basic information chart. Clients can communicate with the AMICA through (3) content but moreover through (4) intuitive such as clicking on buttons or joins as well as giving criticism. Moreover, we centered on planning this bot so that it can be associated to a huge number of stages to extend outreach to the community.

(1)PROCESS FLOW DIAGRAM

3.1 RESPONDING TO A SPECIFIC INPUT

Ifyouwantaparticularlogicadaptertoonlyrespondtoauni quetypeofinput,thebestwaytodothisisbyoverriding the can_processmethod on your own logic adapter.
Here is a simple example. Let's say that we only want this logic adapter to generate a response when the input statement starts with "Hey AMICA". This way, statements such as

```
def can_process(self, statement):
    if statement.text.startswith('Hey AMICA')
        return True
    else:
        return False
```

"Hey AMICA, what time is it?" will be processed, but statements such as "Do you know what time it is?" will not be processed.

3.2 INTERACTING WITH SERVICES

In some cases, it is useful to have a logic adapter that can interact with an external service or api in order tocomplete its task. Here is an example that demonstrates how this could be done. For this example we will use a fictitious API endpoint that returns the currenttemperature

```
def can_process(self, statement):
    """
    Return true if the input statement
    contains "what" and "is" and
    "temperature".
    """
    words = ['what', 'is', 'temperature']
    if all(x in statement.text.split() for x in words)
        return True
    else:
    return False

def process(self, statement):
```

```
    from chatterbot.conversation import Statement
    import requests

    # Make a request to the temperature API
    response = requests.get('https://api.temperature.com/current?units=celsius')
    data = response.json()

    # Let's base the confidence value on if the request was successful
    if response.status_code == 200:
        confidence = 1
    else:
        confidence = 0

    temperature = data.get('temperature', 'unavailable')
    response_statement = Statement('The current temperature is {}'.
.format(temperature))

    return confidence, response_statement
```

3.3 TRAINING THE AMICA

The most important thing the makes AMICA different from other chatbots is ability to train the bot according to the requirements of the clients. Most of the chatbots are rigid in their approach and adding a new query is a handful task but in AMICA it just require loading the trainer and writing fixed set of lines and then inside brackets first the query is saved then in next the answer to the particular query.

```
from chatterbot.training import ListTrainer

trainer = ListTrainer(bot)

trainer.train([
    'How are you?',
    'I am good.',
    'That is good to hear.',
    'Thank you',
    'You are welcome.',
])
```

3.4 TRAINING WITH THE TWITTER API

chatterbot.trainers.**TwitterTrainer**(*storage*,***kwargs*)Allows the chat bot to be trained using data gathered fromTwitter

Parameters:

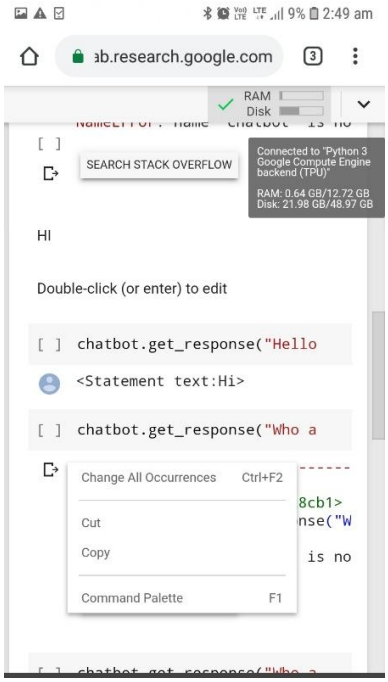
- **random_seed_word**– The seed word to be used to get random tweets from the Twitter API. This parameter is optional. By default it is the word'random'.
- **twitter_lang**– Language for results as ISO 639-1 code. This parameter is optional. Default is None (alllanguages).

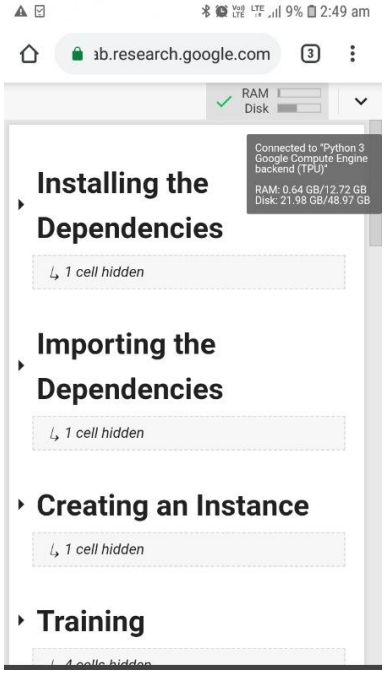
Createannewappusingyourtwitteraccount.Oncecreated,itwillprovideyouwiththefollowingcredentialsthatare required to work with the TwitterAPI.

Parameter	Description
twitter_consumer_key	Consumer key of twitter app.
twitter_consumer_secret	Consumer secret of twitter app.
twitter_access_token_key	Access token key of twitter app.
twitter_access_token_secret	Access token secret of twitter app.

3.5 TWITTER TRAINING EXAMPLE

```
from chatterbot import ChatBot
from chatterbot.trainers import TwitterTrainer
from settings import TWITTER
import logging


'''
This example demonstrates how you can train your chat bot
using data from Twitter.

To use this example, create a new file called settings.py.
In settings.py define the following:

TWITTER = {
    "CONSUMER_KEY": "my-twitter-consumer-key",
    "CONSUMER_SECRET": "my-twitter-consumer-secret",
    "ACCESS_TOKEN": "my-access-token",
    "ACCESS_TOKEN_SECRET": "my-access-token-secret"
}
'''

# Comment out the following line to disable verbose logging
logging.basicConfig(level=logging.INFO)

chatbot = ChatBot(
    "TwitterBot",
    logic_adapters=[
        "chatterbot.logic.BestMatch"
    ],
    input_adapter="chatterbot.input.TerminalAdapter",
    output_adapter="chatterbot.output.TerminalAdapter",
    database_uri="./twitter-database.db",
    twitter_consumer_key=TWITTER["CONSUMER_KEY"],
    twitter_consumer_secret=TWITTER["CONSUMER_SECRET"],
    twitter_access_token_key=TWITTER["ACCESS_TOKEN"],
    twitter_access_token_secret=TWITTER["ACCESS_TOKEN_SECRET"]
)

trainer = TwitterTrainer(chatbot)

trainer.train()

chatbot.logger.info('Trained database generated successfully!')
```





(2) SCREENSHOTS OF CURRENT WORKING PROJECT SAVED ON COLAB

IV. CONCLUSION

AMICA is one of the first general chatbots which can be trained and revised according to the needs of the users and it can cater to any types of businesses as it is not made for any specific company and it uses twitter dataset which makes it's vocabulary even more user friendly and let it incorporate most of the slang words and acronym used by today's kids .

Although it solves a lot of current problem but still there is always room for improvement as more corpus can be included in the AMICA to make it reach more users and data set of the that twitter can be incorporated for the corpus and we can also provide more self learning capabilities to the AMICA .So it can be considered as good start on the new path of better and ever evolving chatbots and it's technology .

REFERENCES

[1] SameeraAAbdul-Kaderand John Woods.2015. Survey on chatbot design techniques in speech conversation systems. *Int.J.Adv.Comput.Sci.Appl.*(2015).

[2] H.Al-Zubaideand A.Issa.2011. Ont Bot: Ontology based chatbot. In *International Symposiumon Innovations in Information and Communications Technology*.

[3] A.Augello,G.Pilato, G.Vassallo, and S.Gaglio.2009. A Semantic Layeron Semi-Structured Data Sources for Intuitive Chatbots. In *2009 International Conferenceon Complex, Intelligent and Software Intensive Systems.*760–765.

[4] Andreas Both, Dennis Diefenbach, Kuldeep Singh, Saeedeh Shekarpour, Didier Cherix, and Christoph Lange.2016.Qanary-A Methodology for Vocabulary-Driven Open Question Answering Systems.In*ESWC*.625–641.

[5] TimofeyErmilov,DiegoMoussallem,RicardoUsbeck,andAxel-Cyrille Ngonga Ngomo.2017. GENESIS: a generic RDF data access interface. In *Proceeding sof the International Conference on WebIntelligence*.125–131.

[6] JensLehmann,RobertIsele,MaxJakob,AnjaJentzsch,DimitrisKontokostas,PabloNMendes,SebastianHellmann,MohamedMorsey,PatrickVanKleef,SörenAuer,etal.2015.DBpedia–alarge-scale, multilingual knowledge base extracted fromWikipedia. *SemanticWeb*6,2(2015),167–195.

[7] Michael Mc Tear, Zoraida Callejas, and DavidGriol.2016.Conversational Inter-faces: Pastand Present. In *The Conversational Interface*.Springer,51–72.

[8] PabloN.Mendes,MaxJakob,AndrésGarcía-Silva, and Christian Bizer.2011. DB pediaspot light: shedding light on the web of documents. In *Proceedings the 7th International Conference on Semantic Systems, I-SEMANTICS*.1–8