ORIGINAL RESEARCH PAPER

Volume-8 | Issue-4 | April-2019 | PRINT ISSN No 2277 - 8179

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH

COMMUNICATION BETWEEN TWO AGENTS USING EMOTIONAL COGNITIVE CONVERSATIONAL AGENT ARCHITECTURE (ECCAA)

Computer Science

Gnaneswari Gnanaguru Research Scholar, Jain University, Asst. Professor, Department of Computer Applications, New Horizon College, Bangalore

ABSTRACT

This research is an attempt to substitute an agent in place of a customer care executive in a helpdesk kind of scenario by building an emotional cognitive conversational agent. Some of the cognition that may be required to achieve human like dialog system are goal, desire, belief, intention, decision making, emotions, compassion, emotional pragmatics etc. The functionality of this Cognitive Architectures is Decision making, Prediction, Problem solving, Reasoning and Belief, action, communication between agents, Learning and reflection.
Conversational agent or Chatbot is become to rule our day to day lives. Alexa, Siri, Watson, etc have become the buzz words of today. This research paper is an attempt to implement the architecture for Conversational agents based on human cognition using Python and Machine Learning for communication between two agents.

KEYWORDS

Artificial Intelligence, Cognitive science, Chatbot, Machine Learning, Python, Conversational Agents, Communicating Agents

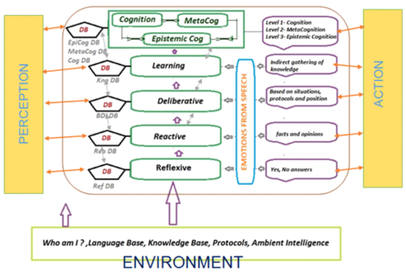
INTRODUCTION:

A simple Chatbot is programmed using Python and its machine learning packages. A dataset is created based on the domain you want the Chatbot to build a knowledge base. Train the model using the above said architecture and test for accuracy.

1. Emotional Cognitive Conversational Agent Architecture (ECCAA):

The ECCAA is designed based on the hypothesis that 'There is an internal cognitive system in a conversational agent architecture that will map a syntactic structure to a semantic one based on pragmatics'.

ECCAA



2. Reflexive Layer:

In this layer only the yes/no responses are trained to the 'GGBot'.
E.g. Are apples red fruits? Is India democratic?

Here basic stimuli and responses can be trained from the knowledge base and meta data which is the information on the agent's involve9d in conversation are stored . The dataset used to train is 'ggbotyesno'.

Knowledge base of Domain includes Company policies, Do's and Dont's, Today's Rates, Trouble shooting queries, Clarification, Functioning and protocols, etc.

Meta Data is information about the Agent which includes Name, Address, Status, designation, family details, employer details, etc.., Personality, state of mind, Status in the society, body language, facial expressions, emotions, etc.., Desire, Belief, Intentions (BDI) and Goals.

```
#Emotional Conversational Agent Architecture using Cognitive Pragmatics
#ECCAAModel.py
from chatterbot import ChatBot
bot1 = ChatBot('GGBot1',trainer='chatterbot.trainers.ListTrainer')
bot1.trainer.export_for_training('./export1.yml')
conversation=open('ggyesno.csv','r').readlines()
bot1.train(conversation)
bot1.trainer.export_for_training('./ggbotmodel1.csv')
```

3. Reactive Layer:

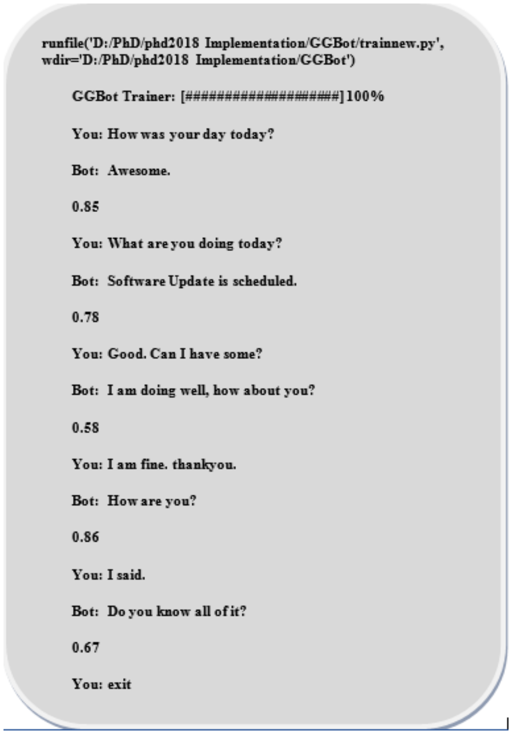
This layer is very similar to the previous layer, except that instead of on yes/no responses; responses are facts stored in the knowledge base and in the meta data. The dataset contains more stimuli-response with the responses derived from the facts stored in knowledge base together with the yes/no responses. The dataset used for this level of training is known as 'ggbotfacts'.

E.g. Who is the president of United States? What is your designation?

```
# create the dialog file
conversation=open('ggfacts.csv','r').readlines()
bot2.train(conversation)
bot2.trainer.export_for_training('./ggbotmodel2.csv')
```

4. Deliberative Layer:

A Heuristic search is performed based on Neural Networks to choose the most accurate reply. In order to train and perform heuristic search a large English corpus know as 'ChatterBotEnglishCorpus is used along with the datasets 'ggbotyesno' and 'ggbotfacts'.
The training is as follows;



```
runfile('D:/PhD/phd2018 Implementation/GGBot/trainnew.py',
wdir='D:/PhD/phd2018 Implementation/GGBot')
    GGBot Trainer: [####################] 100%
    You: How was your day today?
    Bot:  Awesome.
    0.85
    You: What are you doing today?
    Bot:  Software Update is scheduled.
    0.78
    You: Good. Can I have some?
    Bot:  I am doing well, how about you?
    0.58
    You: I am fine. thankyou.
    Bot:  How are you?
    0.86
    You: I said.
    Bot:  Do you know all of it?
    0.67
    You: exit
```

The numbers above indicate the percentage of accuracy which is also referred as confidence in the responses.

```
# Print the responses
  while True:
  request = input ('You: ')
  if request == "exit" :
  break
  i=i+1
  response1 = ggbot.get_response(request)
  threshlist1.append((response1.confidence)*100)
  ilist1.append(i)
  print('GGBot: ', response1)
  print((response1.confidence)*100)
```

5. Learning Layer:

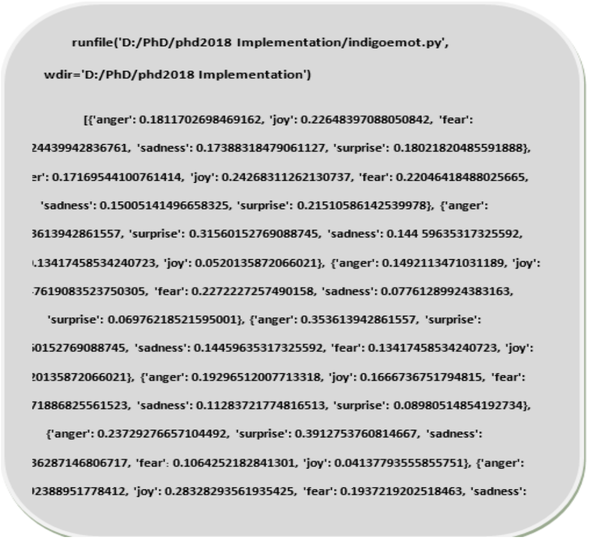
In all the above models, learning is inevitable in a ChatBot. So practically learning is not a separate layer. Whenever there is a response for a stimuli, the new response is stored in the database. The databased use is SQLite3 database.

```
        #Create and Connect to SQL database

        bot = ChatBot('GGBot',
storage_adapter='chatterbot.storage.SQLStorageAdapter',database='./datav
asc.sqlite3')

      # It happens after the learning layer
      ifget_feedback():
      bot.learn_response(response, input_statement)
      bot.storage.add_to_conversation(CONVERSATION_ID, statement,
response)
```

Only successful conversations are stored in the database. In order to filter conversations, we have a feedback system.

6. Cognition Layer:

The core of the ECCAA is the cognitive layer. 'Emotion' is added to the dataset. Before training the dataset, every stimuli and response is analyzed and classified based on the emotions. This is done using an emotion detector API known as 'indico'. It is capable of detecting which is the dominant emotion in the sentence. The basic emotions considered are anger, joy, fear, sadness, surprise. For every statement, an vector value is derived for every emotion. Now training is performed as usual with all the stimulus and responses with emotions. The emotion with highest vector value is assigned to that particular sentence



```
      runfile('D:/PhD/phd2018 Implementation/indigoemot.py',
   wdir='D:/PhD/phd2018 Implementation')

         [{'anger': 0.1811702698469162, 'joy': 0.22648397088050842, 'fear':
24439942836761, 'sadness': 0.17388318479061127, 'surprise': 0.18021820485591888},
er': 0.17169544100761414, 'joy': 0.24268311262130737, 'fear': 0.22046418488025665,
 'sadness': 0.15005141496658325, 'surprise': 0.21510586142539978}, {'anger':
3613942861557, 'surprise': 0.31560152769088745, 'sadness': 0.144 59635317325592,
.13417458534240723, 'joy': 0.0520135872066021}, {'anger': 0.1492113471031189, 'joy':
7619083523750305, 'fear': 0.2272227257490158, 'sadness': 0.07761289924383163,
  'surprise': 0.06976218521595001}, {'anger': 0.353613942861557, 'surprise':
0152769088745, 'sadness': 0.14459635317325592, 'fear': 0.13417458534240723, 'joy':
20135872066021}, {'anger': 0.19296512007713318, 'joy': 0.1666736751794815, 'fear':
71886825561523, 'sadness': 0.11283721774816513, 'surprise': 0.08980514854192734},
  {'anger': 0.23729276657104492, 'surprise': 0.3912753760814667, 'sadness':
36287146806717, 'fear': 0.1064252182841301, 'joy': 0.04137793555855751}, {'anger':
2388951778412, 'joy': 0.28328293561935425, 'fear': 0.1937219202518463, 'sadness':
```

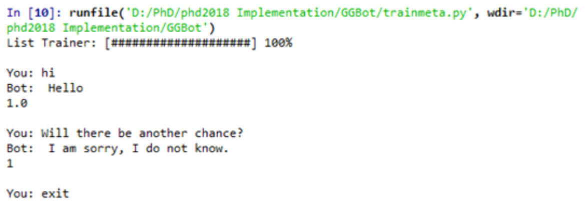
7. Emotion Agent Design in ECCAA:

Here the deliberative layer is made more effective by adding emotions to the stimuli and responses of the dataset. This enhanced dataset is executed in the Cognitive layer. The Emotion Engine is a program that is used to catch the emotion from the stimuli and responses.
Emotion Prediction of the given Text,.

```
#Prediction emotion for the given input text
import indicoio
import pandas
conversation=open('ggbot.csv','r').readlines()
print(indicoio.emotion(conversation))
#f = open('txtemotion.txt','w')
#f.write(indicoio.emotion(conversation))
#f.close()
df = pandas.DataFrame(data=indicoio.emotion(conversation))
df.to_csv("./txtemotion.csv", sep=',',index=False)
```

8. MetaCognition Layer:

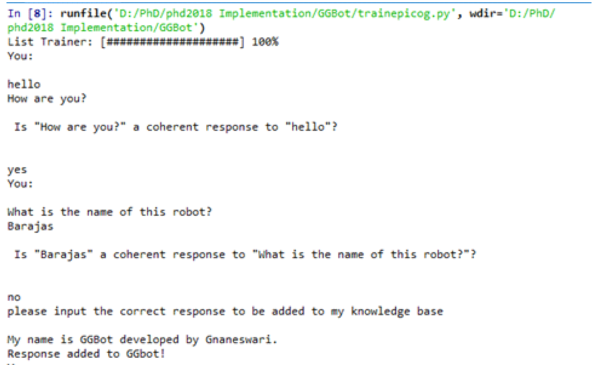
This layer is more intelligent than the above layers. This layer is self-reflective. This can be implemented in which the chatbot has to choose responses with confidence value more than the threshold assigned. If the confidence is say < 65% then, the response must be a specific response such as 'I am sorry, I don't know.'



```
In [10]: runfile('D:/PhD/phd2018 Implementation/GGBot/trainmeta.py', wdir='D:/PhD/
phd2018 Implementation/GGBot')
List Trainer: [####################] 100%

You: hi
Bot:  Hello
1.0

You: Will there be another chance?
Bot:  I am sorry, I do not know.
1

You: exit
```

9. Epistemic Cognition Layer:

In Epistemic cognition level, the agent reflects on the limits of its knowledge and overcome such limitations. This is the highest level in ECCAA. It is not only 'self-reflective' but also learn by itself. This is implemented by adding responses directly to store in the database that is specified in the 'Storage Adapter'. This added response is added to the trained dataset.



```
In [8]: runfile('D:/PhD/phd2018 Implementation/GGBot/trainepicog.py', wdir='D:/PhD/
phd2018 Implementation/GGBot')
List Trainer: [####################] 100%
You:

hello
How are you?

 Is "How are you?" a coherent response to "hello"?


yes
You:

What is the name of this robot?
Barajas

 Is "Barajas" a coherent response to "What is the name of this robot?"?


no
please input the correct response to be added to my knowledge base

My name is GGBot developed by Gnaneswari.
Response added to GGbot!
```

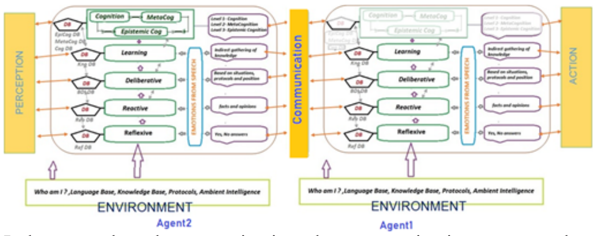
10. Experimentation Results:

A questionnaire was prepared commonly for all the models inorder to measure their performances. A graph was plotted with responses in the x axis and the confidence value in the y axis.

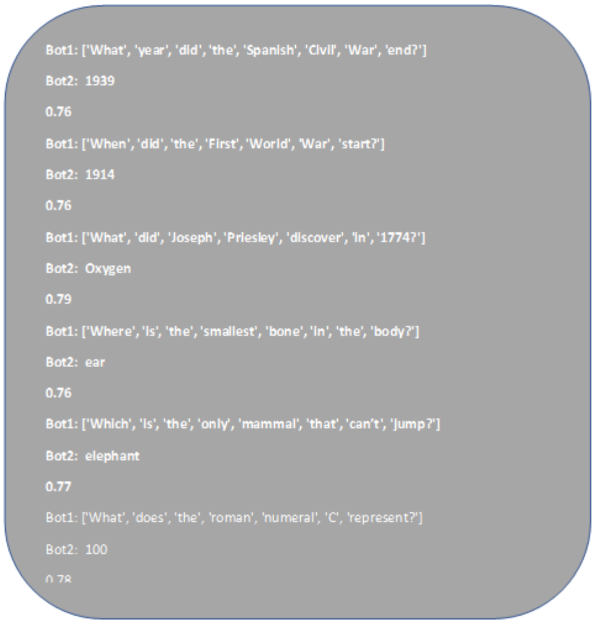
```
import matplotlib.pyplot as plt
plt.plot(ilist1,threshlist1, color='green')
plt.xlabel('Responses')
plt.ylabel('Confidence Level')
plt.title('Confidence Level in responses in Model 1')
plt.show()
```

11. Design of ECCAA for communication between two agents

Communication depends on situation, belief or goal which decides the direction the conversation agent choses. A decision tree is used to chooses that one answer based on the goal. The BDI approach is one of the successful approaches in this scenario.

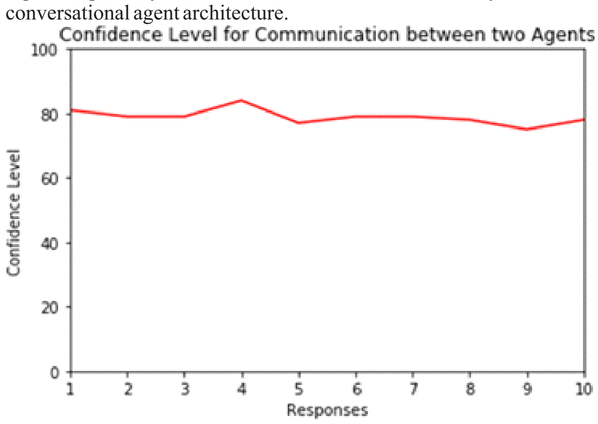


In language-based communication, the communication process takes place by mapping the internal representation of Agent1 to the communication language. Thus, successfully transmitting the internal representation of Agent1 to Agent2 with the language used for communication. Then such conversational agent is known as collaborative agent.



```
Bot1: ['What', 'year', 'did', 'the', 'Spanish', 'Civil', 'War', 'end?']
Bot2: 1939
0.76
Bot1: ['When', 'did', 'the', 'First', 'World', 'War', 'start?']
Bot2: 1914
0.76
Bot1: ['What', 'did', 'Joseph', 'Priesley', 'discover', 'in', '1774?']
Bot2: Oxygen
0.79
Bot1: ['Where', 'is', 'the', 'smallest', 'bone', 'in', 'the', 'body?']
Bot2: ear
0.76
Bot1: ['Which', 'is', 'the', 'only', 'mammal', 'that', 'can't', 'jump?']
Bot2: elephant
0.77
Bot1: ['What', 'does', 'the', 'roman', 'numeral', 'C', 'represent?']
Bot2: 100
0.78
```

A communication design becomes complete only when the knowledge between two agents is transferrable. This can be achieved between two agents that are trained with ECCAA architecture. Agent1 requires some information from Agent2, So, let us assume that Agent2 is well trained with ECCAA architecture and is fully intelligent whereas, Agent1 is partially trained with ECCAA architecture or any other basic conversational agent architecture.



CONCLUSION:

Thus, it is very essential that the agents communicate with each other to gain knowledge. The ECCAA architecture can be used to train each agent and can also be programmed to communicate with each other. The above training using ECCAA can achieve upto 85% of accuracy depending on the dataset.

REFERENCES:

[1] Pat Langley, John E. Laird, Seth Rogers, Cognitive architectures: Research issues and challenges, Cognitive Syst ems Research, ISSN 1389-0417, Volume 10, Issue 2, 2009, Pages 141-160
[2] Venkatamuni, Vijayakumar Maragal. A Society of Mind Approach to Cognition and Metacognition in a Cognitive Architecture, Dissertation of Doctor of Philosophy in Computer Science and Engineering, University of Hull, London, 2008
[3] Gnanaguru Gnaneswari, Venkatamuni, Vijayakumar Maragal , Building a Conversational Agent based on the principles of Cognitive Pragmatics using Cognitive Architecture, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 6 Issue 02, February-2017.
[4] AbuShawar, Bayan & Atwell, Eric. (2015). ALICE chatbot: Trials and outputs, Computación y Sistemas, Vol. 19, No. 4, 2015, pp. 625–63
[5] Arend Hintze, Understanding the four types of AI, from reactive robots to self-aware beings, Michigan State University, November 14, 2016, https://theconversation.com/understanding-the-four-types-of-ai-from-reactive-robots-to-self-aware-beings-67616, accessed on 15th Nov 2017.
[6] P Singh, EMONE: An Architecture for Reflective Commonsense Thinking, Dissertation of Doctor of Philosophy in Computer Science and Engineering. Cambridge, MA: Massachusetts Institute of Technology, 2005.
[7] Weitzenfeld, A., Arbib, M., Alexander, A.: NSL—Neural Simulation Language: A System for Brain Modeling, MIT Press, Cambridge, MA (2002) .
[8] N, Davis D., Computational Architectures for Intelligence and Motivation. International Symposium on Intelligent Control, Vancouver, Canada :17th IEEE, 2002.