**Contextual - Chat Bot**

What Are Contextual Chatbots?

Contextual chatbots are kind of advanced chatbots. Their basic aim is to try to figure out what user intends i.e. in which sense or proportion the user is asking a question or doing some random stuff on the website. It then revert those sentiments and behavior according to the intentions of the user. These chatbots remember the previous things a user has already asked or performed and based on that, it presents a more thoughtful answer.

Example of contextual chatbot is when a user orders a pizza. If the user has already given his location and preferences, the bot will not ask him the same basic question again. It will simply ask for confirmation and voila! The order is placed and out for delivery.

What Makes Contextual Chatbots So Special?

Virality is what every product maker strives for; it’s the perfect customer acquisition channel. Automating workflows and finding relevant leads is the motto of every business. They want to acquire customer and grow faster. The next generation of communication will be instant and no one would want to spend much time on just buying and searching through the web. Contextual chatbots analyze the intent of the user and grabs the gist to offer what they are looking for.

**The Bits & Pieces That Makes Chatbot A Contextual One**

When it comes to chatbots, we know they have no automatic knowledge of their own, so they can’t use context like we humans do. However, it’s possible for us to provide or feed them with the right information and tools so that they can utilize the context on their own. These kinds of chatbots are called as contextual chatbots. Let’s explore more about contextual chatbots and how the context can be the most powerful tool of a chatbot :

* **Situational Context**
* **Lingual Context**
* **Persistence Context**
* **Emotional Context**

**Reference Link** : <https://medium.com/makerobos/what-are-contextual-chatbots-how-they-can-make-a-world-of-difference-in-user-experience-e7446c96664e>

**Contextualization**

We want to handle a question about renting a moped and ask if the rental is for today. That clarification question is a simple contextual response. If the user responds ‘today’ *and the context is the rental timeframe* then it’s best they call the rental company’s 1–800 #. No time to waste.

To achieve this we will add the notion of ‘state’ to our framework. This is comprised of a data-structure to maintain state and specific code to manipulate it while processing intents.

Because the state of our state-machine needs to be easily persisted, restored, copied, etc. it’s important to keep it all in a data structure such as a dictionary.

Our context state is a dictionary, it will contain state for each user. We’ll use some unique identified for each user (eg. cell #). This allows our framework and state-machine to *maintain state for multiple users simultaneously*.

Reference Link : <https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077>

YouTube : Building Contexual ChatBot using Tensor Flow : <https://youtu.be/019a30EnNGk>

Why Tflearn over Keras

1. With Tflearn python arrays can be used directly
2. Tflearn API is more closer to TensorFLow code as compared to keras.
3. Tflear provide better performance than keras

GitHub: <https://github.com/nitinkaushik01/Deep_and_Machine_Learning_Projects/blob/master/Build_ChatBot_using_Neural_Network/Chatbot.ipynb>

*#Adding some context to the conversation i.e. Contexualization for altering question and intents etc.*

*# create a data structure to hold user context*

context = {}

ERROR\_THRESHOLD = 0.25

**def** classify(sentence):

*# generate probabilities from the model*

results = model.predict([bow(sentence, words)])[0]

*# filter out predictions below a threshold*

results = [[i,r] **for** i,r **in** enumerate(results) **if** r>ERROR\_THRESHOLD]

*# sort by strength of probability*

results.sort(key=**lambda** x: x[1], reverse=**True**)

return\_list = []

**for** r **in** results:

return\_list.append((classes[r[0]], r[1]))

*# return tuple of intent and probability*

**return** return\_list

**def** response(sentence, userID='123', show\_details=**False**):

results = classify(sentence)

*# if we have a classification then find the matching intent tag*

**if** results:

*# loop as long as there are matches to process*

**while** results:

**for** i **in** intents['intents']:

*# find a tag matching the first result*

**if** i['tag'] == results[0][0]:

*# set context for this intent if necessary*

**if** 'context\_set' **in** i:

**if** show\_details: print ('context:', i['context\_set'])

context[userID] = i['context\_set']

*# check if this intent is contextual and applies to this user's conversation*

**if** **not** 'context\_filter' **in** i **or** \

(userID **in** context **and** 'context\_filter' **in** i **and** i['context\_filter'] == context[userID]):

**if** show\_details: print ('tag:', i['tag'])

*# a random response from the intent*

**return** print(random.choice(i['responses']))

results.pop(0)

Author : Piyush Jain