CHATBOT HACKATHON

Purpose: 1

Setup and Important instructions: 1

Setups and pre-requisites: 2

PreHackathon-Problem1: (3 Marks) 2

Important instructions: 2

Hackathon Problem statement: 2

Stage 1: Alexa chatbot for a new domain: (22 -Marks) 2

Stage 2: Sample Python code for language processing (N-gram or TF-IDF) (5 Marks) 4

Stage 3: Python code for intent recognition (5 Marks) 5

Stage 4: Exercises over end-to-end Python Chatbot program: (15 Marks) 6

Purpose:

During the course of this hackathon, we intend to build a conversational bot to interact with the user and achieve the desired outcome through the conversation. You may use the bot to either order a book online or reserve a table at a restaurant. We shall take a two-pronged approach towards this task

1. Use Amazon Alexa APIs to build a bot easily by defining the desired utterances and intents
2. See how a bot may be built from scratch and understand the underlying logic

Setup and Important instructions:

Setups and pre-requisites:

1. For all setups related to Alexa Developer and AWS account refer the PetMatch skill Alexa experiment given to you last week.
2. For server access of your teams for aiml-sandbox1.talentsprint.com environment, please note the logins were given to you last week. You could use the same credentials.
3. It is also expected that you've completed the PetMatch experiment (link) and you answer the following questions for 3 marks to your mentor (if not done last week, get the evalution done as a part of final evalutaion):

PreHackathon-Problem1: (3 Marks)

Answer the following to your mentor -

1. What is an invocation call? Create 3 possible ones?
2. For a given Intent, can the Intent names be non-unique?I.e. Can a particular intent be called by different 'Intent names'?
3. Are multiple Utterances possible for a given intent?
4. What is the difference between a "Slot Value" and a "Slot Name"
5. Lamda functions are nothing but the JSON output explaining the conversation flow? TRUE/FALSE
6. Experiment with changes in Intent, Slot, Dialog flow and see how it changes the JSON file in the JSON Editor tab.

Important instructions:

VERY IMP: Perform "source venv/bin/active" on your aiml-sandbox1.talentsprint.com server first before proceeding into any of the steps below for the hackathon!! This means, if your session disconnects for some reason, you have to ensure you have executed the above command.

Hackathon Problem statement:

Here is the problem statement and various stages of the hackathon:

Stage 1: Alexa chatbot for a new domain: (22 -Marks)

Pick “**one**” completely new Domain from the following list and set up a new Alexa skill on that domain.
NOTE: This has to be a new “Intent” in parallel with your Petmatch intent, so that we have two parallel intents to compare with for the rest of the Alexa hackathon (If for some reason you don’t have the Petmatch skill/intent ready, then you have to ensure you create two new intents, for the same comparison purpose). Also, this has to be an end-end working chatbot. (Note: To make it realistic, you could rename your ‘Pet match’ invocation as ‘Petch match and (say) Hotel booking’ skill). List below:

Buying a laptop: The user should be asked questions about Brand, RAM, Screen size, Hard Disk, Size/Type, OS and other such parameters. The bot should give a final selection
Buying a mobile phone: The user should be asked questions about Brand, Size, Accessories and
other parameters. The bot should give a final selection
Buying Vegetables/Fruits: The bot should ask which items and for each item get additional details. For example, California Grapes, Green Grapes, Shimla Apple, Washington Apple, Fuji Apple
Library: After getting information about author, title, subject and so on the bot should pick the book
Jukebox: After getting choices of Genre, Style, Artist, Album etc., the bot should pick the song
Cab Booking: Assuming that the starting point is fixed as HYD Airport, the bot should gather information about your destination, number of people, luggage quantity and pick a cab for you
Movie Choice: The bot should gather your preferences: language, Actor, Genre and the date you
want to go (today, tomorrow, coming Tuesday) and suggest the movie and offer the Movie hall choices
Baby Name Selection: The bot should find out the parameters such as Ethnic Group, Region, Religion, Gender, Starting Letter, Other Numerology constraints and offer a few choices

Doctor Appointment Booking: Gather Specialty, Hospital, Location, Time of Day, Date information and check if a Doctor is free or not

Criteria for evaluation:

1. Your chatbot should initiate an intelligent dialogue.
2. Your chatbot should be able to switch the intent between ‘Pet match’ (which you’ve already executed) and your new domain. You are expected to have at least 10 conversations turns.
3. It should ensure at least 6 different slots are ensured to be answered by the user. Slot names should allow synonyms. (You could refer definitions related to entity-resolution in the Amazon Alexa skills documentation)

4. The chatbot should have a clean exit ensuring it repeats all of the details you've given.
5. There are further a lot of possibilities of what you can do with a skill once created within AWS. These may contribute to bonus marks on mentor's discretion.

Stage 2: Sample Python code for language processing (N-gram or TF-IDF) (5 Marks)

The Python coding environment for Lamda Functions is primarily used to design your intent responses. However, since it is a Python environment, it is expected for you to use this environment to write a simple program to:

a) Perform a TF/IDF or CountVectorizer operation on all your 'Utterance' data, and provide TF/IDF vector for 2 of the sample utterances.

OR

b) Perform a simple n-gram count on your utterances.

NOTE:
i) This code has to be written over your existing lambda_function.py (for details of how to zip and upload lambda_function.py refer to PetMatch skill experiment document)
ii) You have to install necessary packages using pip install and copy into the folder containing lambda_function.py, and only then zip and upload to the S3.
For example: if you want to use NLTK you have to ensure the following:

1. Ensure you have performed 'source venv/bin/activate'
2. Run 'pip install nltk'
3. In your home directory **./venv/lib/python2.7/site-packages** you will find the folder of the package you just installed. Copy that folder into the folder where lambda_function.py resides, then zip and upload to S3 etc.

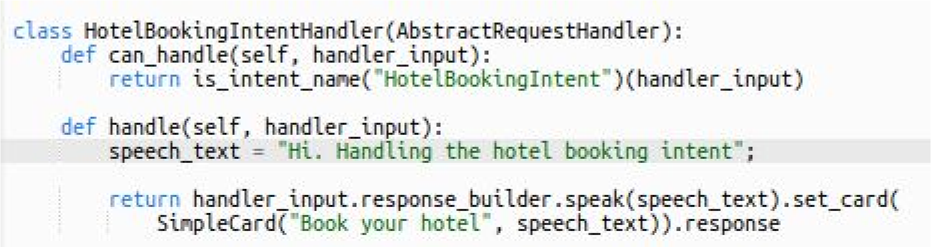
The functions to read the utterance data from lambda file is present at here HelperFunctionToReadUtterancesIntoLambda.py . You can add these to your lambda_function.py at the top after all the imports.

Criteria for evaluation:

1. You should be able to print out the N-grams (or TF/IDF vector; as per your choice above) for at least two of your sample utterances.

Stage 3: Python code for intent recognition (5 Marks)

Here too you will use the Python Lambda environment as above, this time to write a Intent recognition ML algorithm (such as n-gram match or bag-of-words), and compare this against the prediction of Alexa (which is almost always right).
For example if normal Alexa handler looks like this:
Normal Alexa handler:



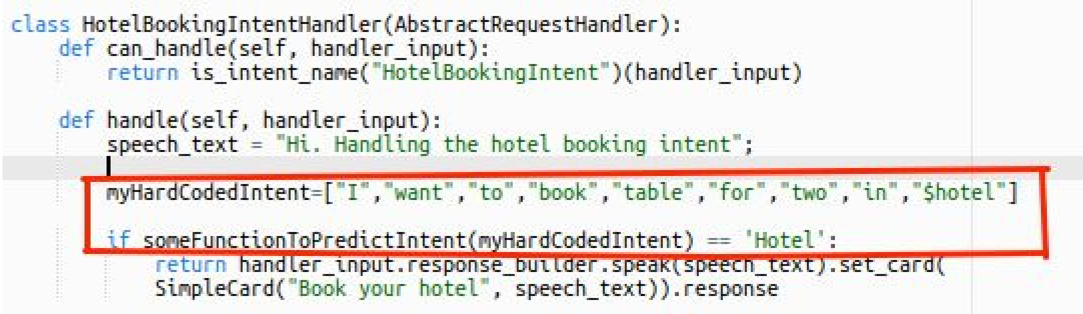
```
class HotelBookingIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return is_intent_name("HotelBookingIntent")(handler_input)

    def handle(self, handler_input):
        speech_text = "Hi. Handling the hotel booking intent";

        return handler_input.response_builder.speak(speech_text).set_card(
            SimpleCard("Book your hotel", speech_text)).response
```

You can change the code to "return" only if your intent-matching algorithm predicts the correct out of the two intents (*this is why in first stage you were asked to ensure you have 2 intents/skills within your chatbo*t) **for a given hard-coded utterance (from your list of utterances) by the "mentor"**. (For example: In the code below the string "myHardCodedIntent" will be set by your mentor). And if your algorithm inside function "someFunctionToPredictIntent" returns the right intent Alexa will return the right response. We expect to see this right response repeatably.

Modified (as per explanation above) Alexa Handler:



```
class HotelBookingIntentHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        return is_intent_name("HotelBookingIntent")(handler_input)

    def handle(self, handler_input):
        speech_text = "Hi. Handling the hotel booking intent";

        myHardCodedIntent=["I","want","to","book","table","for","two","in","$hotel"]

        if someFunctionToPredictIntent(myHardCodedIntent) == 'Hotel':
            return handler_input.response_builder.speak(speech_text).set_card(
            SimpleCard("Book your hotel", speech_text)).response
```

Criteria for evaluation:

1. At least on 4 occasions (continuous), your function should be able to classify to either of the intents. (NOTE: Mentors will try utterances from both the intents).

Stage 4: Exercises over end-to-end Python Chatbot program: (15 Marks)

Appreciate and build on a simple chatbot Python program.

To understand the architecture of this chatbot click the link (Note: the video of Chatbot Code explanation was already uploaded on LMS).

All the Intents/Entities/Params data, and also the chatbot code has been given to at the following link. **You are free add more data to the text files. However take care of ensure that you don't leave any empty space or new-line charaters at the end of your text data. (because the Python program might not handle it well)** Simply run the following commands to download the entire content at your home directory of aiml-sandbox1.talensprint.com:

1. wget https://cdn.talentsprint.com/aiml/Experiment_related_data/Chatbot_V3.zip
2. unzip Chatbot_V3.zip
3. cd Chatbot_v3
4. Run "python Chatbot.py" to start the chatbot. (If you wish to use colab, the same code is also available in the **Chatbot_V3** folder present in the **Alexa-Chatbot-Hackathon** folder of your experiment google drive.)

Note:

1) The chatbot is already quite functional with a lot of utterances. This sets the stage for your next tasks:
2) When editing all the python files such as Chatbot.py "do not use tab" to give space. Instead press space bar. Else you'll get a Python error related to indenting.)

Criteria for evaluation:

Implement all of the following.

1. • Adding word2vec for intent identification (replacing ngram match. Word2vec bin file given in the Chatbot_v3 folder i.e. after you unzip.)
2. • Spelling mistakes handling
3. • Graceful restart at any stage of the chat
4. • Range, regular expressions etc. for getting attributes
5. • Add a branch based on an incoming attribute value

.