**Report**

**Steps:**

First of all, In order to create a chatbot the process started by the following:

Taking our dataset after it was translated from english to arabic and trying to remove unnecessary columns that won’t make difference in the training and leaving only :

* The context
* translated \_Prompt
* Translated\_Utterance

This step’s main purpose was to reduce the size of the data.

The second step was to clean this dataset by removing any rows that contain null values, sentences that contain (< , >,..etc) which does not make any sense. This step was mainly done to have a clean and meaningful data set for the machine learning model to work on, in order to have the required model.

Then,after searching for the two top feelings having the most meaningful sentences, we choose to train the model on two contexts which are angry and proud. And then we went to the second phase.

The second phase started by tokenizing each sentence in our data set according to the feeling and having a bag of every word that appeared in these sentences in order to give them to the machine learning model, these steps were as follows. After having a clean and a ready data set, we looped over all the questions and tokenized them using a predefined function called nltk.word\_tokenize(),after tokenization we removed unnecessary stop words such as[',','.','.','"','!','(',')','?','؟'] . Then we appended each array of tokenization with its context and we added the context we chose in the array of classes. As a result we had something called documents that hold each sentence and it’s feeling in one place to be easily accessed by the machine learning model.

The next step was to build our model using 3 dense layers with dropout= 0.5 and SGD optimizer and we gave it the train dataset that we cleaned. At this step our model was able to predict the context (Feeling) from the input sentence or from the context of the conversation that was given as an input.

The last step was to build a GUI for our chat bot so that the user can easily interact with the chatbot without accessing the code.

**The pipeline of the chatbot goes as following :**

* The question is given to the chatbot\_response function. This function first sends the sentence to the model and predicts its context.
* Sending this context along with the sentence to get\_response function which does one of the three followings
* Return the same answer if the question already exists in the trained dataset.
* If the entered text is a greeting text the chatbot will automatically response with (“مرحبا بك”) and if it's a goodbye sentence the chat bot will respond with (“الى اللقاء”).
* Otherwise save both questions and answers of the same predicted context in two separate arrays to choose one answer for the question that is the most similar to the input question.
* Pass the given sentence to clean\_up\_sentence which clean the sentence and tokenize it to be easily accessed.
* After that it checks if the returned value from get\_response found an answer which means that the question already was in the dataset or if the returned array contains more than one answer and if that is the case we calculate the tf (term frequency ) for it.
* Then we return the answer with the higher term frequency. However under two conditions
  + The first condition is that the length of the incoming question must have more than third the length value of the found question in order to have its answer.
  + The second condition is that the number of similar words found between the two sentences must be greater than third the length of the input question.

In the uploaded video demo we first ran the GUI and inside the GUI when the user hits the send button, the text inside the message box is given as input to chatbot\_reponse function and the result is displayed on the screen as shown in the demo