ASSIGNMENT 02

NAME : R.D.SILVA

REG NO : EN91445

INDEX : 18/ENG/105

USER GUIDANCE

- User can easily use this application to find the shelf numbers for the user needed goods after running the application.
- User can tell each good that needed to buy from supermarket one by one (one after response from the chatbot).
- User can get the shelf numbers according to the goods that have in the shop.
  Ex: If user asks ‘Apple’ then bot will reply that ‘all the fruits are in shelf number 1’.
- Since speech recognition cannot be 100% accurate, chatbot cannot understand some words and therefore it will reply as ‘I didn’t understand that. Can you repeat it again?’.
- Once user finished asking queries about goods, he/she can close the application.
- If voice recognition didn’t work properly, user can type and send the queries.

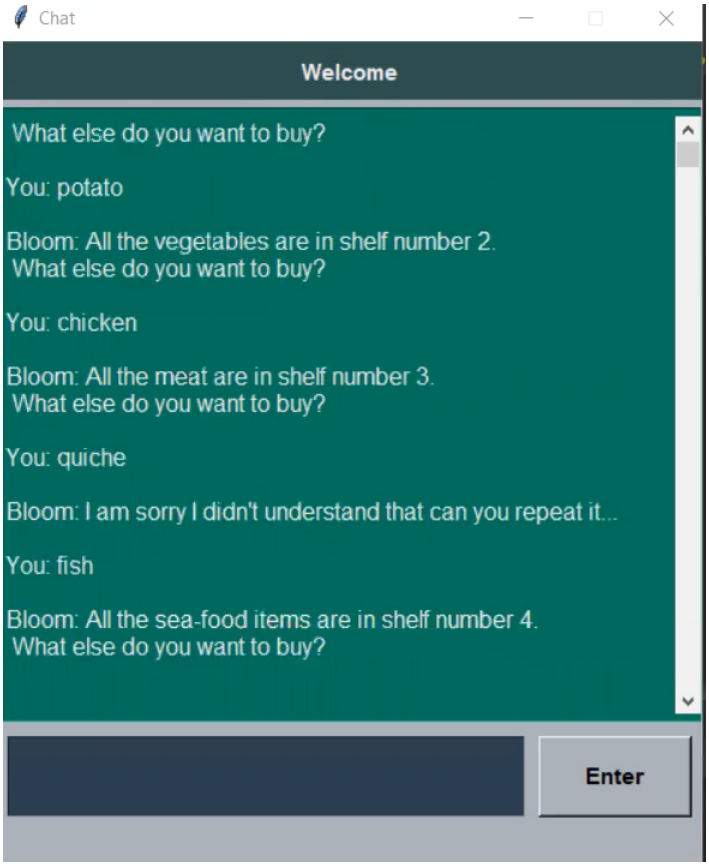

Figure 01: User interface of the application

IMPLEMENTATION

Chatbot application was implemented using PyTorch with Python 3.7 (not working for 3.8) which is a feed forward neural network with 2 hidden layers. As the first step conversational intents were transformed to Tensor flow model. Then a chatbot framework was built to process responses.

STEP 01

- The JSON file which names as 'intents' contains the tag, pattern and responses to train the chatbot. Tag is the unique name, Pattern is some user queries and from responses one will be used by chatbot. After importing the JSON file into main python file, words classification and organization was done by creating list of documents (sentences) where each sentence is a list of tokenized (split the words according to tokenization techniques) and stemmed words (generate root words) after transforming to lowercase and each document consist with a class (after tokenization and stemming all the punctuation marks are excluded from the list). Since tensorflow doesn't support that data structure the document of words were transformed to tensors of numbers using bag_of_words() function. For instance,
  - First some the words are classified into unique tags as below.
    "Hey", "Hello" → greeting
    "Thanks", "Thank you very much" → thanks
  - If the sentence is "Hello, How are you? Thank you very much", then after successful voice recognition and then stemming and tokenizing, it takes into an array as all words,
    all words = ["Hello", "How", "are", "you", "Thank", "you", "very", "much"]
  - Then for each different pattern an array was created as same size of all words array and then checked whether the words matched with all words array and if it matched then insert 1 in that index and 0 with unmatched indexes.
    "Hello" → [1, 0, 0, 0, 0, 0, 0, 0]
    "How are you?" → [0, 1, 1, 1, 0, 0, 0, 0]
    "Thanks" → [0, 0, 0, 0, 1, 0, 0, 0]
    "Thank you very much" → [0, 0, 0, 1, 0, 1, 1, 1]
  - Those patterns were named as 'x' and for each pattern tags are labeled and therefore it named as vector 'y'. (ex: greeting- 0, goodbye -1, thanks -2, etc)

- Then after the model was trained, probabilities have been taken for different classes. Below diagrams summarized all above mentioned points. (after voice converted to text)

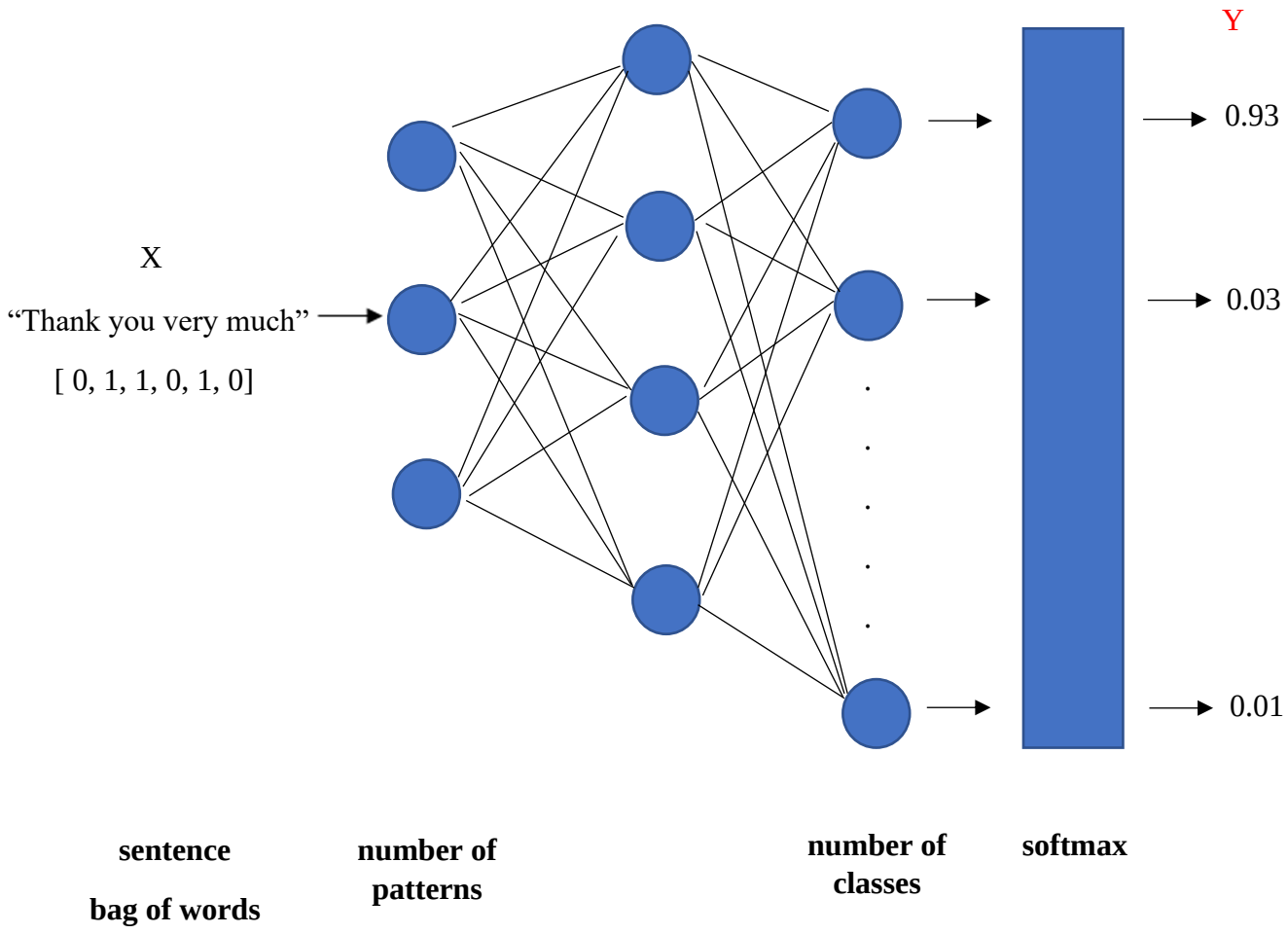

Figure 02: Feed Forward Neural Network

- ‘NeuralNet’ class was built to get probabilities as mentioned in ‘softmax’ in figure 01. (learning rate was taken as 0.001)

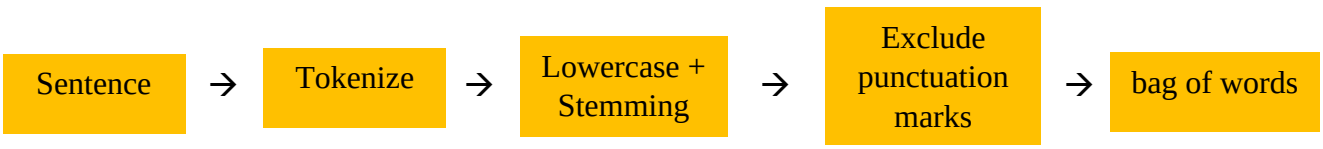

Figure 03: NLP pipeline

- Finally, the model can be built.

STEP 02

- After getting user query, the output of the bot will be model from 'NeuralNet' class and using PyTorch, tags will be predicted. Then using 'softmax' (tensorflow function) function, probabilities will be calculated after the predictions. Among the probabilities, randomly choose one predicted class which must be greater than 0.75. If it is not greater than 0.75, the chatbot will respond as it didn't understand what user said.

User asks queries through voice and at first voice is converted to string using functions consist in 'speech_recognition' library. Then the converted message is sent to the chatbot after pressing the button. After that above mentioned theories will applied to the user query and the response of the chat bot will be given as a voice using 'pyttsx3' library.

[1]

USED LIBRARIES

- nltk
- numpy
- torch (all tensorflow functions required this library)
- tkinter
- pyttsx3
- pipwin
- speech_recognition
- pyaudio

REFERENCES

[1]"Contextual Chatbots with Tensorflow", *Medium*, 2021. [Online]. Available: https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077?gi=4aa8cf9c3b2c. [Accessed: 30- Jun- 2021].