PROJECT – CHATBOT

Using python

By

ABISEK KAMTHAN R S

ABSTRACT

A chatbot enables a user to simply ask questions in the same manner that they would respond to humans. The most well-known chatbots currently are voices chatbots: SIRI and Alexa. However, chatbots have been adopted and brought into the daily application at a high rate on the computer chat platform. NLP also allows computers and algorithms to understand human interactions through various languages. Recent advances in machine learning have greatly improved the accurate and effective of natural language processing, making chatbots a viable option for many organizations. This improvement in NLP is firing a great deal of additional research which should lead to continued improvement in the effective of chatbots in the years to come.A bot is trained on and according to the training, based on some rules on which it is trained, it answers questions. It is called ruled based approach. The language by which these bots can be created is Artificial Intelligence Markup Language (AIML). It is a language based on XML which allows the developer to write the rules which the bot will follow. In this research paper, We are trying to understand these chatbots and understanding their shortcomings. question or statement submitted by a user and allow the user to control over the content to be displayed.

OBJECTIVE

Nowadays companies are trying to keep up with the upcoming trends in technology. With innovative technologies, numerous activities of the organization are getting simpler and moderate. Chatbots are one of these, embraced by practically all organizations these days.

The main purpose of chatbots is to strengthen customer relations with the company. Chatbots offer personalization and scalability in maintaining customer engagement.

Some of the benefits of chatbots are:

Customer support - Normally bots are used for answering customer queries. Bots could be trained for answering simple questions of customers on their own like "location of the company", "career email id", etc. If the query of a customer is not generic then it can also be transferred to a human agent.

Accessibility - Chatbots can answer the queries of customers 24 hours and all 365 days, while this couldn't be possible through customer service employees.

The large volume of requests - Chatbots can process a large volume of requests from customers. It can answer many customers at the same time without delay.

Maintenance - The maintenance cost of chatbots is much low. Once it's been implemented in the system, you don't need to have after-sales service for it from time to time.

Thus, Chatbots streamline interaction between people and services and enhances customer experience. It provides an opportunity to improve customer engagement.

The creation of chatbots has become much easier nowadays and it's completely hassle-free.

INTRODUCTION

The improvements in the fields of inter-networking and information technology have been intricate in executing an Artificial Intelligent systems. These systems are drawing near to human activities for example choice emotionally supportive networks, robotics, natural language processing. Indeed, even in the artificial intelligent fields there are some hybrid strategies and adaptive techniques that make increase complex techniques. That, yet these days there are additionally several Natural Language Processing and intelligent systems that could comprehend human language. AI systems learn themselves and retrieve insight by perusing required electronic articles that have been exist on the web page.

A chatbot is an AI program that copy human discussions including content and communication in natural language utilizing artificial intelligence method for example, Natural Language Processing is a picture and video processing and voice analysis. chatbot for college management system has been created utilizing AI algorithms that examine the user queries. This chatbot system is an internet application that gives an answer to the broken down queries of a user. Users simply need to choose the classification for inquiries and afterward they can ask the question to the bot that utilizes for noting it. AI has been incorporated to respond to the users inquiries then the user can procure the fitting solutions to their inquiries.

Chatbot has become the centre of focus in this current era thus the bot are being utilized to deliver information more conveniently. A chatbot is a standout amongst the most progressive and promising tools of communication among people and machines. famous chatbots like amazon Alexa, Siri, Facebook, Slack, and many more are in trend. These are very much helpful but in this era of enhancing technology day by day, technology gets updated and accordingly by the user expectations also increases. A user wants more automation in the chatbot although every system is not perfect but there is always a flaw in the system. so in the chatbot there are some problems that the user has experienced while using a chatbot. A chatbot can be described as an answering system where a system will be able to answer questions or to the statement submitted by users and allow users to control over the content to be displayed

METHODOLOGY

1. Prepare the Dependencies

The first step in creating a chatbot in Python with the pickle and keras library is to install the library in your system. It is best if you create and use a new Python virtual environment for the installation. To do so, you have to write and execute this command in your Python terminal.

Now that your setup is ready, we can move on to the next step to create chatbot using python.

2. Import Classes and train and test files

Importing classes is the second step in the Python chatbot creation process. All you need to do is import two classes – ChatBot from chatterbot and keras from tensorflow. Then open the test file and train files required for the chatbot model.

3. Create and Train the Chatbot

This is the third step on creating chatbot in python. The chatbot you are creating will be an instance of the class "ChatBot." Then train the chatbot using the train set .

4. Communicate with the Python Chatbot

Then try to interact with the chatbot using the commands and then try to get the desired value and print down the accuracy results too.

Evaluate the results and modify and train accordingly. Then try some types of examples and communicate with the chatbot and test whether the model produces right results.

CHATBOT CODE

In [1]:

```
import pickle
import numpy as np
```

In [2]:

```
with open ('train_qa.txt','rb') as fp:
    train_data = pickle.load(fp)
```

In [3]:

```
train_data
```

Out[3]:

```
[(['Mary',
   'moved',
   'to',
   'the',
   'bathroom',
   '.',
   'Sandra',
   'journeyed',
   'to',
   'the',
   'bedroom',
   '.'],
  ['Is', 'Sandra', 'in', 'the', 'hallway', '?'],
  'no'),
 (['Mary',
   'moved',
   'to',
   'the'.
```

In [4]:

```
with open ('test_qa.txt','rb') as fo:
    test_data = pickle.load(fo)
```

In [5]:

```
test_data
```

Out[5]:

```
[(['Mary',
   'got',
   'the',
   'milk',
   'there',
   '.',
   'John',
   'moved',
   'to',
   'the',
   'bedroom',
   '.'],
  ['Is', 'John', 'in', 'the', 'kitchen', '?'],
  'no'),
 (['Mary',
   'got',
   'the',
   'milk'.
```

In [6]:

```
len(train_data)
```

Out[6]:

```
10000
```

In [7]:

```
len(test_data)
```

Out[7]:

```
1000
```

In [8]:

```
vocab= set()
```

In [9]:

```
all_data = test_data + train_data
```

In [10]:

```
for story,question,answer in all_data:
    vocab = vocab.union(set(story))
    vocab = vocab.union(set(question))
```

In [11]:

```
vocab.add('yes')
```

In [12]:

```
vocab.add('no')
```

In [13]:

```
max_story_len =  max([len(data[0])for data in all_data])
max_ques_len =  max([len(data[1])for data in all_data])
```

In [14]:

```
max_story_len
```

Out[14]:

```
156
```

In [15]:

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
```

In [16]:

```
tokenizer = Tokenizer(filters = [])
```

In [17]:

```
tokenizer.fit_on_texts(vocab)
```

In [18]:

```
tokenizer.word_index
```

Out[18]:

```
{'in': 1,
 'apple': 2,
 'garden': 3,
 'bathroom': 4,
 'milk': 5,
 'sandra': 6,
 'there': 7,
 'took': 8,
 'yes': 9,
 'down': 10,
 'bedroom': 11,
 'up': 12,
 'put': 13,
 'picked': 14,
 'left': 15,
 'moved': 16,
 'mary': 17,
 'kitchen': 18,
 'hallway': 19,
 'dropped': 20,
 'travelled': 21,
 'to': 22,
 'daniel': 23,
 'is': 24,
 'football': 25,
 '?': 26,
 'the': 27,
 'john': 28,
 'went': 29,
 '.': 30,
 'no': 31,
 'got': 32,
 'grabbed': 33,
 'journeyed': 34,
 'discarded': 35,
 'back': 36,
 'office': 37}
```

In [19]:

```
train_story_text =[]
train_question_text =[]
train_answers_text =[]

for story,question,answer in train_data:
    train_story_text.append(story)
    train_question_text.append(question)
```

In [20]:

```
train_story_seq = tokenizer.texts_to_sequences(train_story_text)
train_story_seq
```

Out[20]:

```
[[17, 16, 22, 27, 4, 30, 6, 34, 22, 27, 11, 30],
 [17,
  16,
  22,
  27,
  4,
  30,
  6,
  34,
  22,
  27,
  11,
  30,
  17,
  29,
  36,
  22,
  27,
```

In [21]:

```
def vectorize_stories(data , word_index=tokenizer.word_index , max_story_len = max_story_len , max_ques_len = max_ques_len ):
    X = []
    Xq = []
    Y = []
    for story,ques,ans in data :
        x = [word_index[word.lower()] for word in story]
        fg = [word_index[word.lower()] for word in ques]
        y = np.zeros(len(word_index)+1)
        y[word_index[ans]] = 1

        X.append(x)
        Xq.append(fg)
        Y.append(y)
    return(pad_sequences(X,max_story_len),
           pad_sequences(Xq,max_ques_len),

           np.array(Y))
```

In [22]:

```
inputs_train,queries_train,answers_train = vectorize_stories(train_data)
```

In [23]:

```
vocab_len = len(vocab)+1
```

In [24]:

```
inputs_test,queries_test,answers_test = vectorize_stories(test_data)
```

In [25]:

```
from tensorflow.keras.models import Sequential ,Model
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import Input,Activation,Dense,Permute,Dropout,add,dot,concatenate,LSTM
```

In [26]:

```
input_sequence = Input((max_story_len,))
question = Input((max_ques_len,))
```

In [27]:

```
input_encoder_m = Sequential()
input_encoder_m.add(Embedding(input_dim = vocab_len, output_dim = 64))
input_encoder_m.add(Dropout(0.3))
```

In [28]:

```
input_encoder_c = Sequential()
input_encoder_c.add(Embedding(input_dim = vocab_len, output_dim = max_ques_len))
input_encoder_c.add(Dropout(0.3))
```

In [29]:

```
ques_encoder = Sequential()
ques_encoder.add(Embedding(input_dim = vocab_len, output_dim = 64,input_length = max_ques_len))
ques_encoder.add(Dropout(0.3))
```

In [30]:

```
input_encoded_m = input_encoder_m(input_sequence)
input_encoded_c = input_encoder_c(input_sequence)
ques_encoded = ques_encoder(question)
```

In [31]:

```
match = dot([input_encoded_m,ques_encoded],axes = (2,2))
match = Activation('softmax')(match)
```

In [32]:

```
response =  add([match,input_encoded_c])
response = Permute((2,1))(response)
```

In [33]:

```
answer = concatenate([response,ques_encoded])
```

In [34]:

```
answer = LSTM(32)(answer)
```

In [35]:

```
answer = Dropout(0.5)(answer)

answer = Dense(vocab_len)(answer)
```

In [36]:

```
answer = Activation('softmax')(answer)
```

In [37]:

```
model = Model([input_sequence,question],answer)
model.compile(optimizer = 'rmsprop', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

In [38]:

```
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 156)]        0           []

 input_2 (InputLayer)           [(None, 6)]          0           []

 sequential (Sequential)        (None, None, 64)     2432        ['input_1[0][0]']

 sequential_2 (Sequential)      (None, 6, 64)        2432        ['input_2[0][0]']

 dot (Dot)                      (None, 156, 6)       0           ['sequential[0][0]',
                                                                  'sequential_2[0][0]']

 activation (Activation)        (None, 156, 6)       0           ['dot[0][0]']

 sequential_1 (Sequential)      (None, None, 6)      228         ['input_1[0][0]']

 add (Add)                      (None, 156, 6)       0           ['activation[0][0]',
                                                                  'sequential_1[0][0]']

 permute (Permute)              (None, 6, 156)       0           ['add[0][0]']

 concatenate (Concatenate)      (None, 6, 220)       0           ['permute[0][0]',
                                                                  'sequential_2[0][0]']

 lstm (LSTM)                    (None, 32)           32384       ['concatenate[0][0]']

 dropout_3 (Dropout)            (None, 32)           0           ['lstm[0][0]']

 dense (Dense)                  (None, 38)           1254        ['dropout_3[0][0]']

 activation_1 (Activation)      (None, 38)           0           ['dense[0][0]']

==================================================================================================
Total params: 38,730
Trainable params: 38,730
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [39]:

```
history = model.fit([inputs_train,queries_train],answers_train,
                    batch_size = 30, epochs = 22,
                    validation_data=([inputs_test,queries_test],answers_test)
                    )
```

```
Epoch 1/22
334/334 [==============================] - 5s 8ms/step - loss: 0.8741 - accuracy: 0.4963 - val_loss: 0.6990 - val_a
ccuracy: 0.5030
Epoch 2/22
334/334 [==============================] - 2s 7ms/step - loss: 0.7049 - accuracy: 0.5075 - val_loss: 0.6936 - val_a
ccuracy: 0.5030
Epoch 3/22
334/334 [==============================] - 2s 6ms/step - loss: 0.6976 - accuracy: 0.5000 - val_loss: 0.6951 - val_a
ccuracy: 0.5030
Epoch 4/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6967 - accuracy: 0.4985 - val_loss: 0.6932 - val_a
ccuracy: 0.5030
Epoch 5/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6963 - accuracy: 0.4941 - val_loss: 0.6948 - val_a
ccuracy: 0.4970
Epoch 6/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6954 - accuracy: 0.5038 - val_loss: 0.6941 - val_a
ccuracy: 0.5030
Epoch 7/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6957 - accuracy: 0.4981 - val_loss: 0.6932 - val_a
ccuracy: 0.4970
Epoch 8/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6951 - accuracy: 0.5049 - val_loss: 0.6951 - val_a
ccuracy: 0.4970
Epoch 9/22
334/334 [==============================] - 3s 8ms/step - loss: 0.6954 - accuracy: 0.4943 - val_loss: 0.6951 - val_a
ccuracy: 0.4970
Epoch 10/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6956 - accuracy: 0.4947 - val_loss: 0.6956 - val_a
ccuracy: 0.4970
Epoch 11/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6950 - accuracy: 0.5039 - val_loss: 0.6934 - val_a
ccuracy: 0.4970
Epoch 12/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6952 - accuracy: 0.5000 - val_loss: 0.6933 - val_a
ccuracy: 0.4970
Epoch 13/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6952 - accuracy: 0.4959 - val_loss: 0.6932 - val_a
ccuracy: 0.5030
Epoch 14/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6951 - accuracy: 0.4916 - val_loss: 0.6946 - val_a
ccuracy: 0.5030
Epoch 15/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6944 - accuracy: 0.5075 - val_loss: 0.6936 - val_a
ccuracy: 0.4970
Epoch 16/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6947 - accuracy: 0.5046 - val_loss: 0.6937 - val_a
ccuracy: 0.4970
Epoch 17/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6951 - accuracy: 0.4969 - val_loss: 0.6941 - val_a
ccuracy: 0.5030
Epoch 18/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6957 - accuracy: 0.4893 - val_loss: 0.6934 - val_a
ccuracy: 0.5030
Epoch 19/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6951 - accuracy: 0.4978 - val_loss: 0.6960 - val_a
ccuracy: 0.4970
Epoch 20/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6947 - accuracy: 0.5030 - val_loss: 0.6933 - val_a
ccuracy: 0.4970
Epoch 21/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6949 - accuracy: 0.4990 - val_loss: 0.6938 - val_a
ccuracy: 0.4970
Epoch 22/22
334/334 [==============================] - 2s 7ms/step - loss: 0.6952 - accuracy: 0.4943 - val_loss: 0.6942 - val_a
ccuracy: 0.5030
```

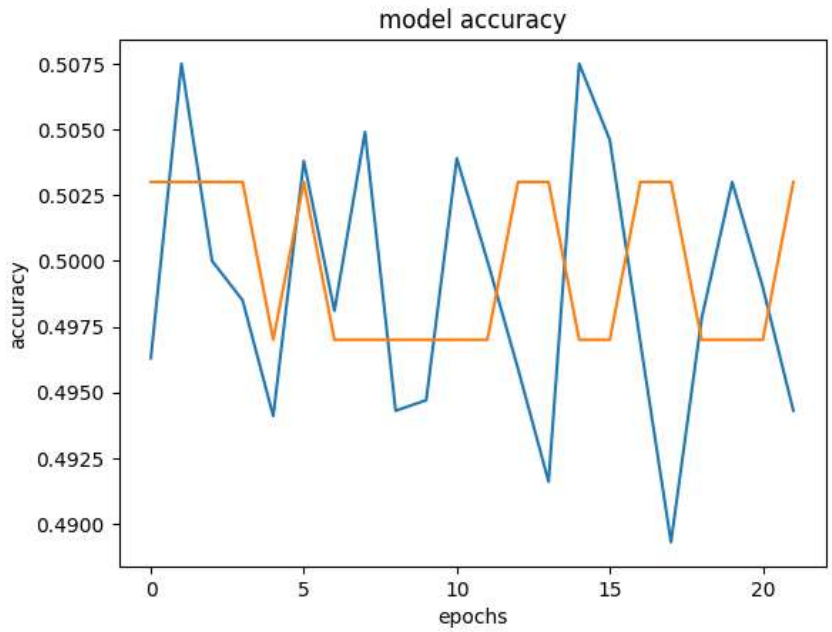
In [40]:

```
import matplotlib.pyplot as plt
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("model accuracy")
plt.ylabel("accuracy")
plt.xlabel("epochs")
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Out[40]:

```
Text(0.5, 0, 'epochs')
```



In [41]:

```
model.save("chatbot_model")
```

```
WARNING:absl:Found untraced functions such as _update_step_xla, lstm_cell_layer_call_fn, lstm_cell_layer_call_and_r
eturn_conditional_losses while saving (showing 3 of 3). These functions will not be directly callable after loadin
g.

INFO:tensorflow:Assets written to: chatbot_model\assets

INFO:tensorflow:Assets written to: chatbot_model\assets
```

In [42]:

```
model.load_weights("chatbot_model")
```

Out[42]:

```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x23745bcb340>
```

In [43]:

```
pred_results = model.predict(([inputs_test,queries_test]))
```

```
32/32 [==============================] - 0s 2ms/step
```

In [44]:

```
pred_results
```

Out[44]:

```
array([[1.26673370e-07, 1.13923697e-07, 1.14301322e-07, ...,
        1.31722217e-07, 1.25852637e-07, 1.04012805e-07],
       [1.27621149e-07, 1.14430506e-07, 1.14886049e-07, ...,
        1.32750330e-07, 1.27064226e-07, 1.04608723e-07],
       [1.26728764e-07, 1.13920812e-07, 1.14090966e-07, ...,
        1.31674554e-07, 1.25894331e-07, 1.03898245e-07],
       ...,
       [1.24142545e-07, 1.11731545e-07, 1.11686163e-07, ...,
        1.28401496e-07, 1.23176463e-07, 1.01533651e-07],
       [1.22207211e-07, 1.09820938e-07, 1.09807225e-07, ...,
        1.25575880e-07, 1.21593743e-07, 9.97840957e-08],
       [1.21639886e-07, 1.09451527e-07, 1.09668647e-07, ...,
        1.25089386e-07, 1.20395526e-07, 9.92313076e-08]], dtype=float32)
```

In [52]:

```
test_data[90][0]
```

Out[52]:

```
['Mary',
 'picked',
 'up',
 'the',
 'apple',
 'there',
 '.',
 'Mary',
 'went',
 'to',
 'the',
 'bathroom',
 '.']
```

In [53]:

```
story = ' '.join(word for word in test_data[90][0])
```

In [54]:

```
story
```

Out[54]:

```
'Mary picked up the apple there . Mary went to the bathroom .'
```

In [55]:

```
queries = ' '.join(word for word in test_data[90][1])
```

In [56]:

```
queries
```

Out[56]:

```
'Is Mary in the bathroom ?'
```

In [57]:

```
test_data[90][2]
```

Out[57]:

```
'yes'
```

In [58]:

```
val_max=np.argmax(pred_results[37])
for key,val in tokenizer.word_index.items():
    if val==val_max:
        k=key
print("predicted answer is ",k)
print("probability of certainity",pred_results[37][val_max])
```

```
predicted answer is  no
probability of certainity 0.52589846
```

CONCLUSION

In this project, we made a college-specific chatbot system that can be custom and fits in an education domain chatbot the addition of this chatbot system in the college website will make the webpage more user interactive as it responds to the user queries very accurately as it is a domain-specific chatbot system, and furthermore we had investigated our college chatbot system design stages. a few different techniques by which the precision of the chatbot system can be made better. gathering feedback from the potential user can be helpful in developing the college Chatbot system ultimately servicing the user queries in conclusion we have made a chatbot in python that can understand user queries and reply accordingly. In the intent file of our chatbot on we can add more patterns and improve patterns which will be helpful when replying to the users and improve the accuracy of our chatbot DL enabled chatbots are becoming more and more popular because of their applications and they can tackle all the problem. it can also be very helpful in teaching and has a lot of applications in teaching the visually impaired.