Create a **Chatbot**

INDEX:-

Abstract –

A chatbot is a computer software program that conducts a conversation via auditory or textual methods. This software is used to perform tasks such as quickly responding to users, informing them, helping to purchase products and providing better service to customers. Chatbots are programs that work on Artificial Intelligence (AI) & Machine Learning & Deep learning Platform. Chatbot has become more popular in business groups right now as it can reduce customer service costs and handles multiple users at a time. But yet to accomplish many tasks there is a need to make chatbots as efficient as possible. In this project, we provide the design of a chatbot, which provides a genuine and accurate answer for any query using Artificial Intelligence Markup Language (AIML) and Latent Semantic Analysis (LSA) with python platform.

Objective:-

A chatbot can communicate with a real person behaving like a human. You can create chatbots for any business the same as you recruit a person for any department of your company. Whether you are a:

- Wedding Planner
- Insurance Assistant
- Education Consultant
- Legal Assistant
- A real estate business
- Recruiter
- Travel Agency
- Hospital or a Beautician!

INTRODUCTION :-

A chatbot is an automated software program that interacts with humans. A chatbot is merely a computer program that fundamentally simulates human conversations. A chatbot that functions through AI and machine learning has an artificial neural network inspired by the neural nodes of the human brain. Chatbots are programs that can do talk like human conversations very easily. For example, Facebook has a machine learning chatbot that creates a platform for companies to interact with their consumers through the Facebook Messenger application. In 2016, chatbots became too popular on Messenger. By the consequences is noted that 2016 was the entire year of chatbots. The software industry is mainly oriented on chatbots. Thousands of chatbots are invented on startups and used by the businesses to improve their customer service, keeping them hanging by a kind communication. According to research, nowadays chatbots are used to solve a number of business tasks across many industries like E-Commerce, Insurance, Banking, Healthcare, Finance, Legal, Telecom, Logistics, Retail, Auto, Leisure, Travel, Sports, Entertainment, Media and many others. Thus that was the moment to look at the chatbots as a new technology in the communication field. Nowadays various companies are using chatbots to answer quickly and efficiently some frequented asking questions from their own customers.

METHODOLOGY:-

We used two datasets in this project provides by in course train. Dataset size is 10000 and test dataset size is 1000,

We used deep learning libraries Keras and TensorFlow, Python libraries Numpy and Math.lib, and NLP methods like tokenization and sequences.

Standard Input,Activation,Dense,Permute,Dropout,add,dot,concatenate,LSTM.

CODE:-

Import Libraries

```
import numpy as np
import tensorflow as tf
import pickle
from tensorflow.keras import layers , activations , models , preprocessing
```

Loading data :-

```
from tensorflow.keras import preprocessing , utils
import os
import yaml

dir_path = 'chatbot_nlp/data'
files_list = os.listdir(dir_path + os.sep)

questions = list()
answers = list()

for filepath in files_list:
    stream = open( dir_path + os.sep + filepath , 'rb')
    docs = yaml.safe_load(stream)
    conversations = docs['conversations']
    for con in conversations:
        if len( con ) > 2 :
```

```
            questions.append(con[0])
            replies = con[ 1 : ]
            ans = ''
            for rep in replies:
                ans += ' ' + rep
            answers.append( ans )
        elif len( con )> 1:
            questions.append(con[0])
            answers.append(con[1])

answers_with_tags = list()
for i in range( len( answers ) ):
    if type( answers[i] ) == str:
        answers_with_tags.append( answers[i] )
    else:
        questions.pop( i )

answers = list()
for i in range( len( answers_with_tags ) ) :
    answers.append( '<START> ' + answers_with_tags[i] + ' <END>' )
tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts( questions + answers )
VOCAB_SIZE = len( tokenizer.word_index )+1
print( 'VOCAB SIZE : {}'.format( VOCAB_SIZE ))
```

Preparing data for Seq2Seq model:-

```
from gensim.models import Word2Vec
import re

vocab = []
for word in tokenizer.word_index:
    vocab.append( word )

def tokenize( sentences ):
    tokens_list = []
    vocabulary = []
    for sentence in sentences:
        sentence = sentence.lower()
        sentence = re.sub( '[^a-zA-Z]', ' ', sentence )
        tokens = sentence.split()
        vocabulary += tokens
        tokens_list.append( tokens )
    return tokens_list , vocabulary

#p = tokenize( questions + answers )
#model = Word2Vec( p[ 0 ] )

#embedding_matrix = np.zeros( ( VOCAB_SIZE , 100 ) )
#for i in range( len( tokenizer.word_index ) ):
    #embedding_matrix[ i ] = model[ vocab[i] ]
```

```
# encoder_input_data

tokenized_questions = tokenizer.texts_to_sequences( questions )

maxlen_questions = max( [ len(x) for x in tokenized_questions ] )

padded_questions = preprocessing.sequence.pad_sequences( tokenized_questions ,
 maxlen=maxlen_questions , padding='post' )

encoder_input_data = np.array( padded_questions )

print( encoder_input_data.shape , maxlen_questions )


# decoder_input_data

tokenized_answers = tokenizer.texts_to_sequences( answers )

maxlen_answers = max( [ len(x) for x in tokenized_answers ] )

padded_answers = preprocessing.sequence.pad_sequences( tokenized_answers , m
axlen=maxlen_answers , padding='post' )

decoder_input_data = np.array( padded_answers )

print( decoder_input_data.shape , maxlen_answers )

# decoder_output_data

tokenized_answers = tokenizer.texts_to_sequences( answers )

for i in range(len(tokenized_answers)) :

    tokenized_answers[i] = tokenized_answers[i][1:]

padded_answers = preprocessing.sequence.pad_sequences( tokenized_answers , m
axlen=maxlen_answers , padding='post' )

onehot_answers = utils.to_categorical( padded_answers , VOCAB_SIZE )

decoder_output_data = np.array( onehot_answers )

print( decoder_output_data.shape )
```

Defining the Encoder-Decoder model:-

```
encoder_inputs = tf.keras.layers.Input(shape=( maxlen_questions , ))

encoder_embedding = tf.keras.layers.Embedding( VOCAB_SIZE, 200 , mask_zero =True ) (encoder_inputs)

encoder_outputs , state_h , state_c = tf.keras.layers.LSTM( 200 , return_state=True )( encoder_embedding )

encoder_states = [ state_h , state_c ]


decoder_inputs = tf.keras.layers.Input(shape=( maxlen_answers ,  ))

decoder_embedding = tf.keras.layers.Embedding( VOCAB_SIZE, 200 , mask_zero =True) (decoder_inputs)

decoder_lstm = tf.keras.layers.LSTM( 200 , return_state=True , return_sequences= True )

decoder_outputs , _ , _ = decoder_lstm ( decoder_embedding , initial_state=encoder_states )

decoder_dense = tf.keras.layers.Dense( VOCAB_SIZE , activation=tf.keras.activations.softmax )

output = decoder_dense ( decoder_outputs )


model = tf.keras.models.Model([encoder_inputs, decoder_inputs], output )

model.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='categorical_crossentropy')


model.summary()
```

output----

```
Model: "model"
_____
_____
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 22)]	0	[]
input_2 (InputLayer)	[(None, 74)]	0	[]
embedding (Embedding)	(None, 22, 200)	378800	['input_1[0][0]']
embedding_1 (Embedding)	(None, 74, 200)	378800	['input_2[0][0]']
lstm (LSTM)	[(None, 200), (None, 200), (None, 200)]	320800	['embedding[0][0]']
lstm_1 (LSTM)	[(None, 74, 200), (None, 200), (None, 200)]	320800	['embedding_1[0][0]', 'lstm[0][1]', 'lstm[0][2]']
dense (Dense)	(None, 74, 1894)	380694	['lstm_1[0][0]']

Total params: 1,779,894

Trainable params: 1,779,894

Non-trainable params: 0

_____

Training the model:-

model.fit([encoder_input_data , decoder_input_data], decoder_output_data, batch_size=50, epochs=20)

model.save('model.h5')

output:-

Epoch 1/20

12/12 [==============================] - 8s 622ms/step - loss: 1.1047

Epoch 2/20

12/12 [==============================] - 8s 667ms/step - loss: 1.0845

Epoch 3/20

12/12 [==============================] - 8s 667ms/step - loss: 1.0645

Epoch 4/20

12/12 [==============================] - 8s 688ms/step - loss: 1.0457

Epoch 5/20

12/12 [==============================] - 8s 634ms/step - loss: 1.0305

Epoch 6/20

12/12 [==============================] - 8s 673ms/step - loss: 1.0164

Epoch 7/20

12/12 [==============================] - 9s 761ms/step - loss: 1.0047

Epoch 8/20

12/12 [==============================] - 8s 649ms/step - loss: 0.9918

Epoch 9/20

12/12 [==============================] - 8s 657ms/step - loss: 0.9785

Epoch 10/20

12/12 [==============================] - 8s 664ms/step - loss: 0.9639

Epoch 11/20

12/12 [==============================] - 8s 665ms/step - loss: 0.9493

Epoch 12/20

12/12 [==============================] - 8s 651ms/step - loss: 0.9360

Epoch 13/20

12/12 [==============================] - 8s 659ms/step - loss: 0.9215

Epoch 14/20

12/12 [==============================] - 10s 811ms/step - loss: 0.9070

Epoch 15/20

12/12 [==============================] - 8s 656ms/step - loss: 0.8933

Epoch 16/20

12/12 [==============================] - 8s 668ms/step - loss: 0.8799

Epoch 17/20

12/12 [==============================] - 8s 663ms/step - loss: 0.8652

Epoch 18/20

12/12 [==============================] - 8s 644ms/step - loss: 0.8521

Epoch 19/20

12/12 [==============================] - 8s 649ms/step - loss: 0.8382

Epoch 20/20

12/12 [==============================] - 8s 624ms/step - loss: 0.8252

CodeText

Defining inference models:-

```
def make_inference_models():

    encoder_model = tf.keras.models.Model(encoder_inputs, encoder_states)

    decoder_state_input_h = tf.keras.layers.Input(shape=( 200 ,))
    decoder_state_input_c = tf.keras.layers.Input(shape=( 200 ,))
  decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
    decoder_outputs, state_h, state_c = decoder_lstm(
        decoder_embedding , initial_state=decoder_states_inputs)
    decoder_states = [state_h, state_c]
    decoder_outputs = decoder_dense(decoder_outputs)
    decoder_model = tf.keras.models.Model(
        [decoder_inputs] + decoder_states_inputs,
        [decoder_outputs] + decoder_states)

    return encoder_model , decoder_model
```

Talking with our Chatbot:-

```
def str_to_tokens( sentence : str ):
    words = sentence.lower().split()
```

```
    tokens_list = list()
    for word in words:
        tokens_list.append( tokenizer.word_index[ word ] )
    return preprocessing.sequence.pad_sequences( [tokens_list] , maxlen=maxlen_q
uestions , padding='post')


enc_model , dec_model = make_inference_models()


for _ in range(10):
    states_values = enc_model.predict( str_to_tokens( input( 'Enter question : ' ) ) )
    empty_target_seq = np.zeros( ( 1 , 1 ) )
    empty_target_seq[0, 0] = tokenizer.word_index['start']
    stop_condition = False
    decoded_translation = ''
    while not stop_condition :
        dec_outputs , h , c = dec_model.predict([ empty_target_seq ] + states_values )
        sampled_word_index = np.argmax( dec_outputs[0, -1, :] )
        sampled_word = None
        for word , index in tokenizer.word_index.items() :
            if sampled_word_index == index :
                decoded_translation += ' {}'.format( word )
                sampled_word = word


        if sampled_word == 'end' or len(decoded_translation.split()) > maxlen_answe
rs:
            stop_condition = True
```

```
        empty_target_seq = np.zeros( ( 1 , 1 ) )
        empty_target_seq[ 0 , 0 ] = sampled_word_index
        states_values = [ h , c ]

    print( decoded_translation )
```

OUTPUT:-

```
Enter question : start hello
1/1 [==============================] - 1s 1s/step
WARNING:tensorflow:Model was constructed with shape (None, 74) for input KerasTensor(type_spec=TensorSpec(shape=(None, 74), dtype=tf.float32, name='input_2'), name='input_2', description="created by layer 'input_2'"), but it was called on an input with incompatible shape (None, 1).
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
 i am a lot of the computer end
Enter question : what can you do
1/1 [==============================] - 0s 25ms/step
```

1/1 [==============================] - 0s 23ms/step

1/1 [==============================] - 0s 19ms/step

1/1 [==============================] - 0s 22ms/step

1/1 [==============================] - 0s 22ms/step

1/1 [==============================] - 0s 25ms/step

1/1 [==============================] - 0s 24ms/step

1/1 [==============================] - 0s 20ms/step

i am i am i am end

Enter question : end

1/1 [==============================] - 0s 21ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 23ms/step

1/1 [==============================] - 0s 26ms/step

1/1 [==============================] - 0s 21ms/step

1/1 [==============================] - 0s 20ms/step

1/1 [==============================] - 0s 21ms/step

computer is a human end

Enter question : <end>

CONCLUSION:-

In this project, we have introduced a chatbot that is able to interact with users. This chatbot can answer queries in the textual user input. For this purpose, AIML with program-o has been used. The chatbot can answer only those questions which he has the answer in its AIML dataset. So, to increase the knowledge of the chatbot, we can add the APIs of Wikipedia, Weather Forecasting Department, Sports, News, Government and a lot more. In such cases, the user will be able to talk and interact with the chatbot in any kind of domain. Using APIs like Weather, Sports, News and Government Services, the chatbot will be able to answer the questions outside of its dataset and which are currently happening in the real world.