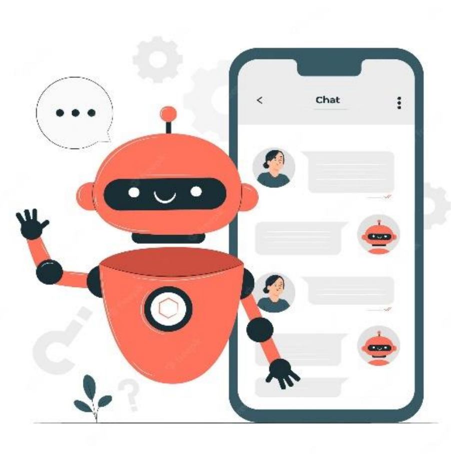
Data Science with Python Internship Project

Project Details:



Title: ChatBot

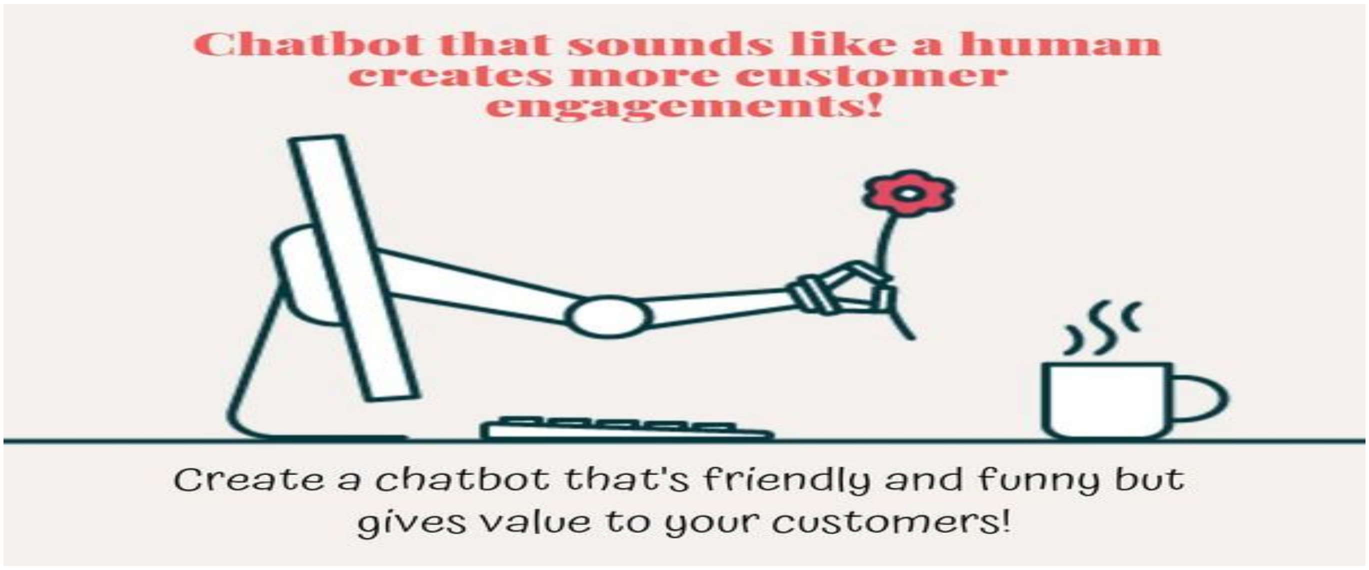
Abstract:

The chatBot can be described as software that can chat with people using artificial intelligence and Machine Learning concepts. 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.

Objective:

A chatbot can communicate with a real person behaving like a human.

- Chatbots are mainly used to provide customer support.
- It helps in catering to a huge amount of target audience at the same time 24/7.
- Can Schedule meetings, Broadcast newsletters, and auto-sequences.
- Acquire leads from Comments
- Create conversational forms and save all the data on spreadsheets
- Chatbots are very intelligent. You train them once and they will communicate with your target audience in their language.
- If you are a company that's functional all around the world, you get a hands-on chatbot asap! Because, while you are asleep, your bot can entertain your customers anywhere in the world.


Chatbot that sounds like a human creates more customer engagements!
Create a chatbot that's friendly and funny but gives value to your customers!

Introduction:

Chatbots, or conversational interfaces as they are also known, present a new way for individuals to interact with computer systems. Traditionally, getting a question answered by a software program involved using a search engine, or filling out a form. A chatbot allows a user to simply ask questions in the same manner that they would address a human. The most well-known chatbots currently are voice chatbots: Alexa and Siri. However, chatbots are currently being adopted at a high rate on computer chat platforms.

The technology at the core of the rise of the chatbot is natural language processing (“NLP”). Recent advances in machine learning have greatly improved the accuracy and effectiveness 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 effectiveness of chatbots in the years to come.

A simple chatbot can be created by loading an FAQ (frequently asked questions) into chatbot software. The functionality of the chatbot can be improved by integrating it into the organization’s enterprise software, allowing more personal questions to be answered, like “What is my balance?”, or “What is the status of my order?”.

Most commercial chatbots are dependent on platforms created by the technology giants for their natural language processing. These include Amazon Lex, Microsoft Cognitive Services, Google Cloud Natural Language API, Facebook Deep Text, and IBM Watson. Platforms, where chatbots are deployed, include Facebook Messenger, Skype, and Slack, among many others.

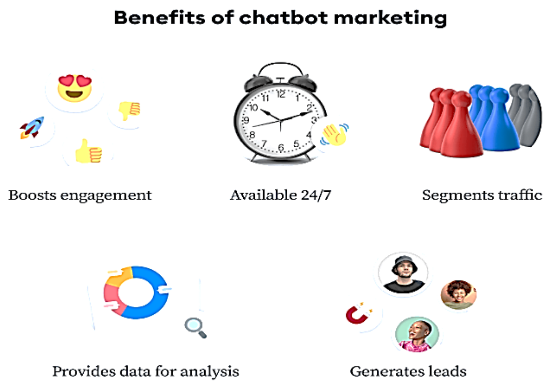
Applications:

A chatbot can be used anywhere a human is interacting with a computer system. These are the areas where the fastest adoption is occurring:

- **Customer Service** — A chatbot can be used as an "assistant" to a live agent, increasing the agent's efficiency. When trained, they can also provide service when the call centre is closed or even act as an independent agent, if desired.
- **Sales/Marketing/Branding** — Chatbots can be used for sales qualification, eCommerce, promotional campaigns, or as a branding vehicle.
- **Human Resources** — An HR chatbot can help with frequently asked questions ("how many vacation days do I have left?") and can act as an onboarding assistant.

Benefits:

- Economically offer 24/7 Service
- Improve Customer Satisfaction
- Reach a Younger Demographic
- Reduce Costs
- Increase Revenue



Methodology:

1. Download the Data from LMS Dashboard.
   - There are train data and test data both lengths are 1000.
2. Defining Objective, Application, and Benefits.
3. Creating ChatBot Model By using Python.
   - Import the required libraries.
   - Define the variables.
   - Create a model.
4. Conclusion
5. Preparation of Project Report.

Code:

# import libraries

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

# open the file

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

# length of the train_data

```
len(train_data)
```

# open test data file

```
with open("test_qa_new.txt",'rb')as fp:
   test_data=pickle.load(fp)
test_data
```

# len of test data

```
len(test_data)
```

# create empty set

```
vocab=set()
```

# combine the data

```
all_data=train_data+test_data
all_data
```

# looping

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

# adding 'yes' and 'no'

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

```
vocab
```

# length of vocab

```
len(vocab)
```

# indexing start from 0 therefore, exact length of vocab

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

# maximum story length

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

# maximum question length

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

# import from keras

```
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
tokenizer=Tokenizer(filters =[])
tokenizer.fit_on_texts(vocab)
```

# display the index number

```
tokenizer.word_index
train_story_text=[]
train_question_text=[]
# train_answer = []
```

```
for story,question,answer in train_data:
   train_story_text.append(story)
   train_question_text.append(question)
train_story_seq=tokenizer.texts_to_sequences(train_story_text)
train_story_seq
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,question,answer in data:
      x.append([word_index[word.lower()] for word in story])
      xq.append([word_index[word.lower()] for word in question])
      y1= np.zeros(len(word_index)+1)
      y1[word_index[answer]] =1
      y.append(y1)



 return (pad_sequences(x,maxlen=max_story_len),
        pad_sequences(xq, maxlen=max_ques_len),
        np.array(y))
```

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

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

```
from keras.models import Sequential,Model
from keras.layers.embeddings import Embedding #input
from keras.layers import Input,Activation,Dense, Permute, Dropout, add, dot,concatenate, LSTM
input_sequence= Input((max_story_len,))
question= Input((max_ques_len,))
```

```
input_sequence
```

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

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

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

```
input_encoded_m= input_encoder_m(input_sequence)
input_encoded_c=input_encoder_c(input_sequence)
question_encoded=question_encoder(question)
```

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

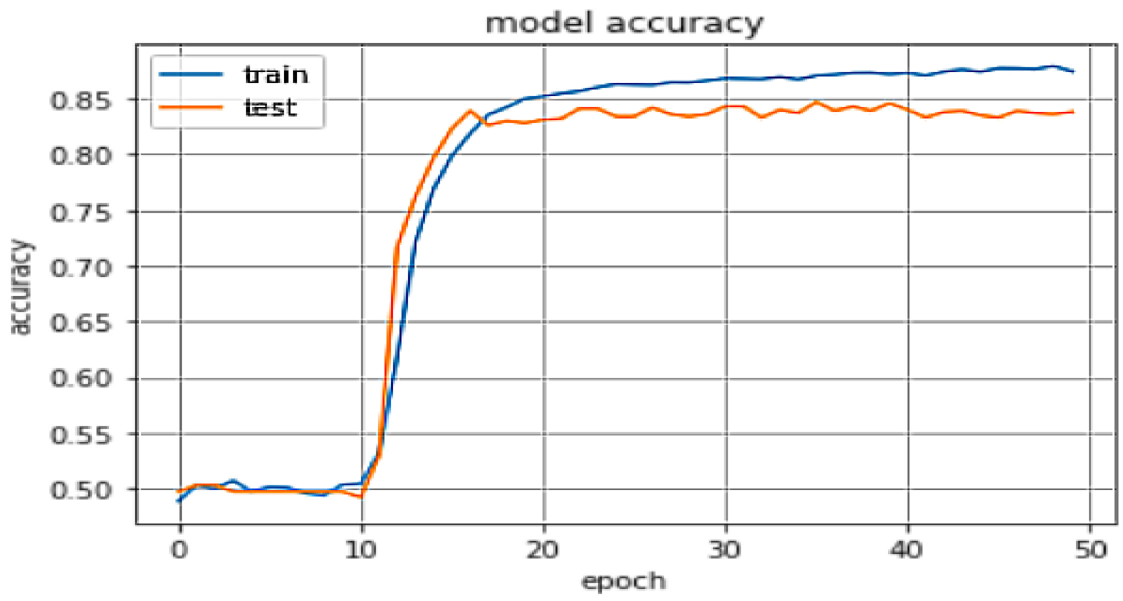
```
response= add([match,input_encoded_c])
response= Permute((2,1))(response)
answer= concatenate([response,question_encoded])
answer= LSTM(32)(answer)
answer= Dropout(0.5)(answer)
```

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

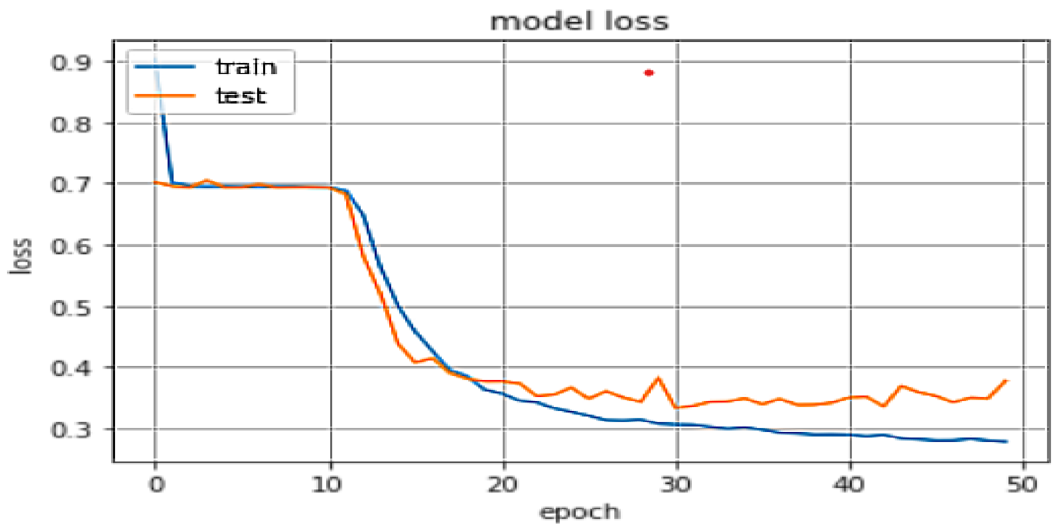
```
model=Model([input_sequence,question],answer)
model.compile(optimizer='rmsprop',loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
model_out=model.fit([inputs_train,queries_train],answer_train,
          batch_size= 30, epochs=25,
          validation_data=([inputs_test,queries_test],answer_test))
```

```
import matplotlib.pyplot as plt
print(model_out.history.keys())
plt.plot(model_out.history['accuracy'])
plt.plot(model_out.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.grid()
plt.legend(['train','test'],loc='upper left')
plt.show()plt.plot(model_out.history['loss'])
```



```
plt.plot(model_out.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.grid()
plt.legend(['train','test'],loc='upper left')
plt.show()
```



```
# save
model.save('chatbot_model')
pred_results=model.predict(([inputs_test, queries_test]))
pred_results
```

```
story=' '.join(word for word in test_data[0][0])
story
querie=' '.join(word for word in test_data[0][1])
querie
answer=''.join(word for word in test_data[0][2])
answer
story=' '.join(word for word in test_data[22][0])
story
querie=' '.join(word for word in test_data[22][1])
querie
answer=''.join(word for word in test_data[22][2])
answer
```

```
val_max=np.argmax(pred_results[22])

for key,val in tokenizer.word_index.items():
   if val==val_max:
      k= key

print('predicted answer is',{k})
print('probability of certainty:', pred_results[22][val_max])
```

```
predicted answer is {'yes'}
probability of certainty: 0.5432208
```

The conclusion from the coding part:

- from the graph plot we conclude that the Accuracy of both train and test data increases by the Epoch.
- The model predicts the answer correctly and the probability of an event will occur is approximately 50% to 57%.

Conclusion:

A chatbot is one of the simple ways to transport data from a computer without having to think of proper keywords to look up in a search or browse several web pages to collect information; users can easily type their query in natural language and retrieve information.

Report by: Maseera Sayyed Riyaz.
(Data Science – Microsoft –B5)
Email ID: maseera2345@gmail.com