NOTEBOOK

Swiggy Chatbot

Hey, Where is My Order?

Learn

#1

Introduction

Swiggy is India's largest online food ordering and delivery platform, founded in 2014. It is based in Bangalore, India, and was founded in 2014 by Nandan Reddy, Sriharsha Majety and Rahul Jaimini. It is now operating in over 500 Indian cities.

Nandan Reddy and Sriharsha Majety, the two founders, created Bundl, an e-commerce website in 2013 to help with courier service and shipping within India. Bundl was put on hold and rebranded in order to enter the meal delivery business.

The meal delivery industry was in chaos at the time, with numerous major startups such as Foodpanda, TinyOwl, and Ola Cafe suffering. In 2014, Majety and Reddy approached Rahul Jaimini, a former Myntra employee, and created Swiggy and its parent firm Bundl Technologies. The company grew quickly after establishing a dedicated delivery network and focusing on logistics and securing crucial resources.

SWIGGY

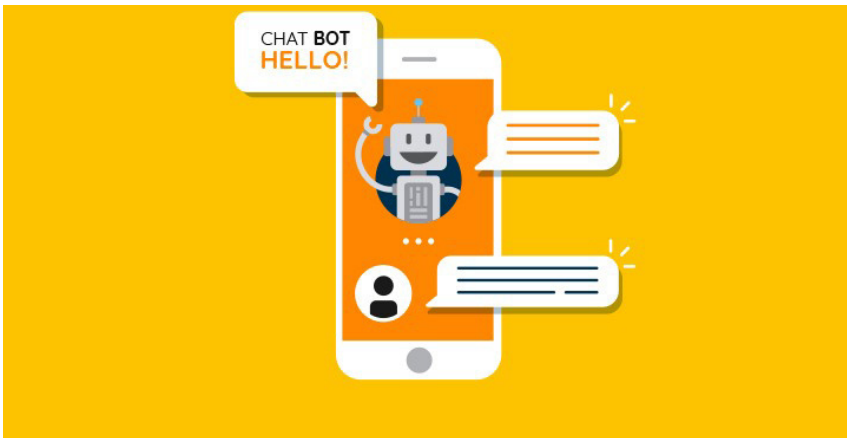
Swiggy is one of the most prominent participants in the global online food ordering and delivery sector. Its quick delivery made folks feel like they were in a bed of roses.

#2

Chatbot

A chatbot is artificial intelligence (AI) software that can imitate a natural language discussion (or chat) with a user via messaging apps, websites or mobile apps.

What is the significance of chatbots? A chatbot is frequently described as one of the most advanced and promising forms of human-machine interaction. Chatbots can automatically simulate interactions with customers based on a set of predefined conditions or events.



From a technology standpoint, however, a chatbot is simply the next step in the evolution of a Question Answering system that uses Natural Language Processing (NLP). One of the most common examples of Natural Language Processing is usage in many organizations' end-use applications in formulating responses to inquiries in natural language.

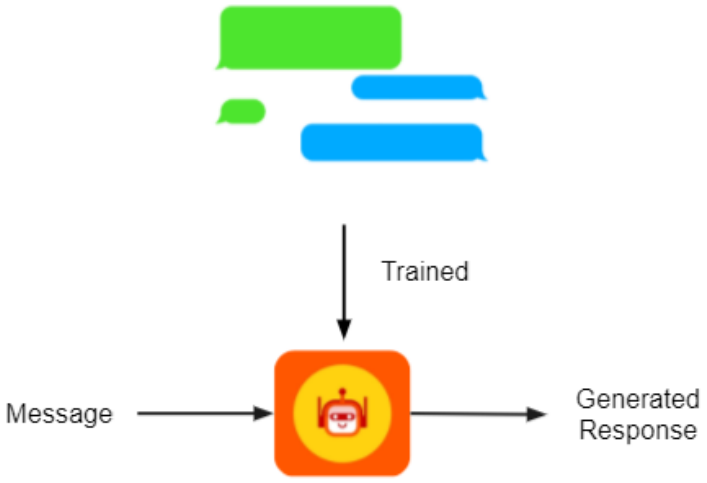
#3

Types of Chatbots

Generative Based

Generative chatbots use a combination of supervised learning, unsupervised learning & reinforcement learning. A generative chatbot is an open-domain chatbot that creates unique language combinations rather than selecting from a list of pre-defined responses. Retrieval-based systems are limited to predefined responses. Chatbots that use generative methods can generate new dialogue based on large amounts of conversational training data.

Generative Based



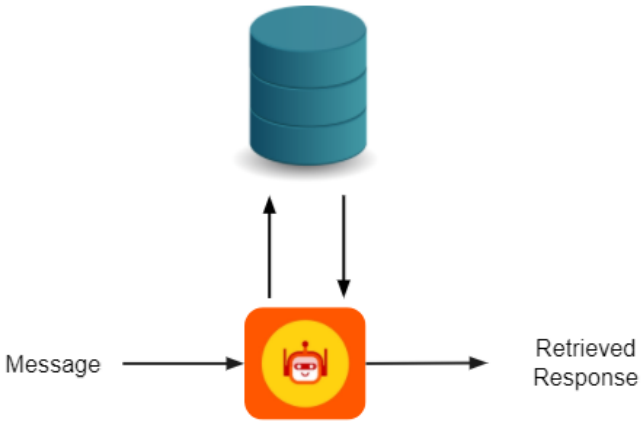
#3

Types of Chatbots

Retrieval Based

Retrieval based chatbots, employ techniques such as keyword matching, machine learning, and deep learning to find the most appropriate response. These chatbots, regardless of technology, solely deliver predefined responses and do not generate fresh output. From a database of predefined responses, the chatbot is trained to offer the best possible response. The responses are based on previously collected data.

Retrieval Based



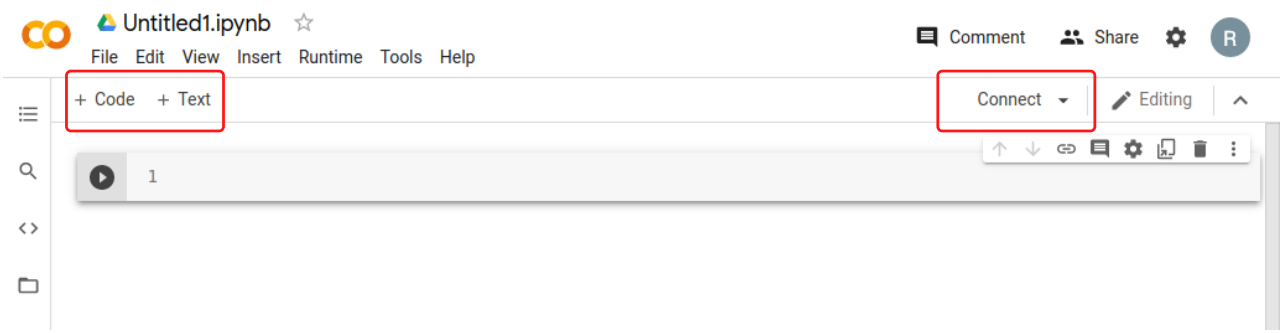
Since the Generative Based Chatbots require huge amount of data to train on and construct the appropriate response, we will be using Retrieval Based approach for our chatbot

#Extra

Introduction to Colab

Nearly all machine learning and deep learning algorithms require good hardware. What if ... you don't have good hardware? Should you drop your dream to be a data scientist? No, there's an alternative Let us introduce you to Colab.

Colab is a service provided by Google which lets you access a virtual machine hosted on google servers. These virtual machines have dual-core Xeon processors, with 12GB of RAM. You can even use GPU for your neural networks. Colab is an interactive python notebook (ipynb), which means that with writing python code, you can also write normal text, include images.
To create a new colab notebook, just go to "https://colab.research.google.com", and create a new notebook. You will get something like below



You can connect to runtime by clicking the "Connect" button in the right corner. You can add a new Code cell, or text cell using respective buttons in the toolbar.

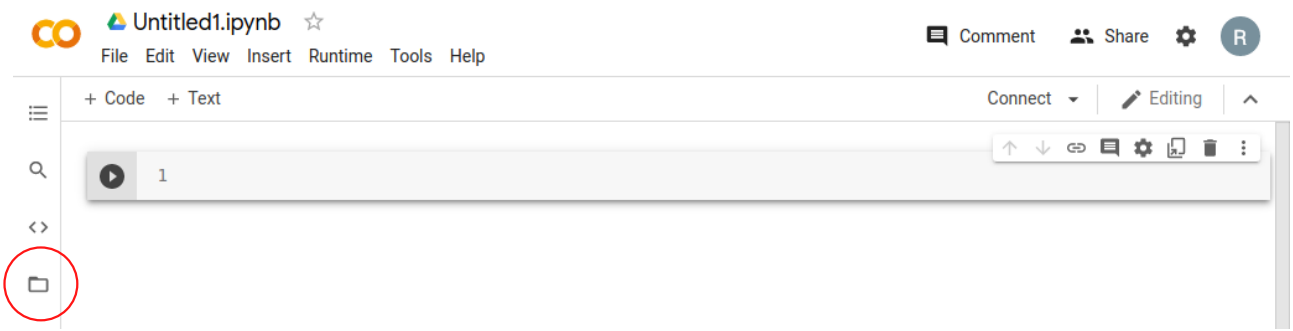
#Extra

Introduction to Colab

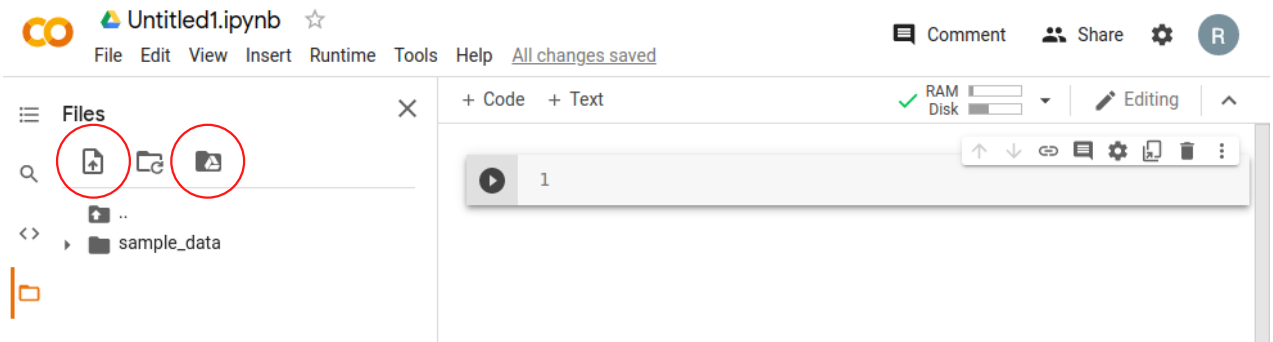
Uploading files

Sometimes you need to use a file from your PC. For that, you can upload the required files to google colab. Colab provides 100 GB in a colab session. If you want to access some files from your drive, you can even do so by connecting the drive to colab.

To upload something, open file pane from left toolbar.



Now, just upload the file using first button, and you can also mount google drive using last button



#3

Natural Language Processing

Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

We will be using a three-step process to transform our "content" into a form that could be understood by a computer algorithm and with which it can extract meaningful insights.

Importing Libraries & Modules

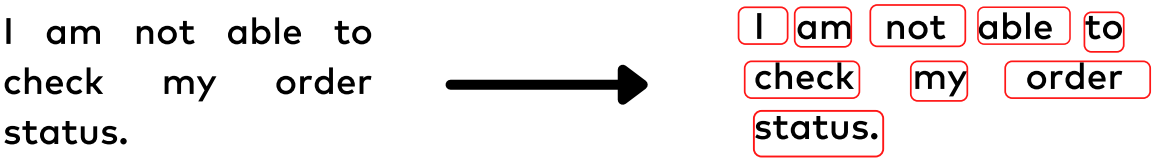
Import nltk and numpy libraries for the text preprocessing.

```
import nltk
import numpy as np
```

#3.1

Tokenization

Tokenization is the process of breaking down sentence or paragraphs into smaller chunks of words called tokens



Downloading 'punkt' model to tokenize message and importing 'word_tokenize' for word tokenization.

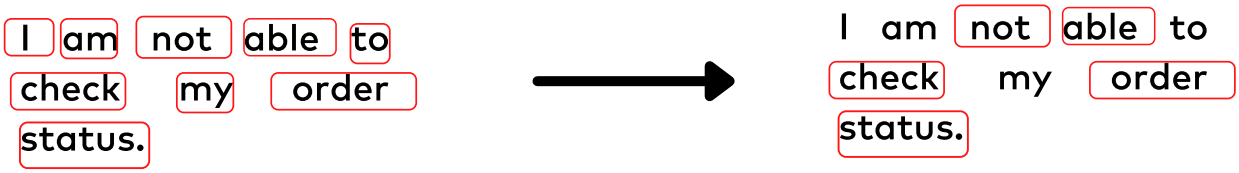
```
nltk.download('punkt')     # downloading model to tokenize message
from nltk.tokenize import word_tokenize
```

#3.2

Stop Words Removal

On removal of some words, the meaning of the sentence doesn't change, like and, am. Those words are called stopwords and should be removed before feeding to any algorithm.

In datasets, some non-stop words repeat very frequently. Those words too should be removed to get an unbiased result from the algorithm.



Loading the complete list of all the common stopwords that needs to be discarded

```
nltk.download('stopwords') # downloading stopwords
from nltk.corpus import stopwords
```

#3.3

Lemmatization

Lemmatization is the process of converting a word to its base form. It considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

For example, lemmatization would correctly identify the base form of 'moving' to 'move'.

'Moving' -> Lemmatization -> 'Move'

Downloading all the lemmas present in English Language & WordNetLemmatizer for lemmetaization.

```
nltk.download('wordnet') # downloading all lemmas of english language
from nltk.stem import WordNetLemmatizer
```

#3.4

Vectorization

After tokenization, and stop words removal, our "content" are still in string format. We need to convert those strings to numbers based on their importance (features). We use TF-IDF vectorization to convert those text to vector of importance.

With TF-IDF we can extract important words in our data. It assign rarely occuring words a high number, and frequently occuring words a very low number.

You can learn more about it from:
https://en.wikipedia.org/wiki/Tf-idf

We can define our own function for stopwords removal, tokenization & lemmatization. It takes the corpus of words as an argument.

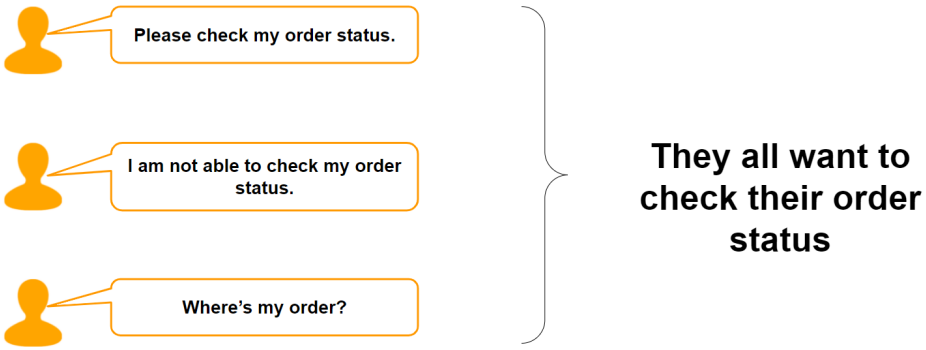
```
def clean_corpus(corpus):
  # lowering every word in text
  corpus = [ doc.lower() for doc in corpus]
  cleaned_corpus = []
  stop_words = stopwords.words('english')
  wordnet_lemmatizer = WordNetLemmatizer()

  # iterating over every text
  for doc in corpus:
    # tokenizing text
    tokens = word_tokenize(doc)
    cleaned_sentence = []
    for token in tokens:
      # removing stopwords, and punctuation
      if token not in stop_words and token.isalpha():
        # applying lemmatization
        cleaned_sentence.append(wordnet_lemmatizer.lemmatize(token))
    cleaned_corpus.append(' '.join(cleaned_sentence))
  return cleaned_corpus
```

#4

Intent Classification

Intent recognition is a form of natural language processing (NLP), a subfield of artificial intelligence. Natural language processing (NLP) and analysis is concerned with computers processing and analyzing natural language, that is, any language that has evolved naturally rather than artificially, such as computer coding languages.

Intent classification or intent recognition is the task of taking a written or spoken input, and classifying it based on what the user wants to achieve. Intent recognition forms an essential component of chatbots and finds use in sales conversions, customer support, and many other areas.



In the above example, three different users are inquiring about their order status using different sentences but the underlying intent is to check what's the current status of their order.

#4

Code

Intent Classification

Loading the intents from 'intents.json' file.

```
import json
with open('/content/intents.json') as file:
    intents = json.load(file)
```

This is how your json file looks. It needs to be cleaned and stored in the form of the vectors.

```
{
        "tag": "greeting",
        "patterns": ["Hi there", "Is anyone there?","Hey","Hola", "Hello", "Good day"],
        "responses": ["Hello, how can I help?", "Good to see you again", "Hi there, how can I help?"]
},
{
        "tag": "goodbye",
        "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time", "No, that's it"],
        "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."]
},
{
        "tag": "thanks",
        "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
        "responses": ["Happy to help!", "Any time!", "My pleasure"]
},
```

```
corpus = []
tags = []
for intent in intents['intents']:
    # taking all patterns in intents to train a neural network
    for pattern in intent['patterns']:
        corpus.append(pattern)
        tags.append(intent['tag'])

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(cleaned_corpus)
```

#4

Intent Classification

Reshaping the vectors for our neural network

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
y = encoder.fit_transform(np.array(tags).reshape(-1,1))
```

We will be developing our neural network for intent classification using the sequential class from tensorflow API

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential([
                    Dense(128, input_shape=(X.shape[1],),
activation='relu'),
                    Dropout(0.2),
                    Dense(64, activation='relu'),
                    Dropout(0.2),
                    Dense(y.shape[1], activation='softmax')
])

model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model.fit(X.toarray(), y.toarray(), epochs=20,
batch_size=1)
```

#4

Intent Classification

Defining the function to predict intent tag of a particular message.

```
# if prediction for every tag is low, then we want to classify that
message as noanswer
INTENT_NOT_FOUND_THRESHOLD = 0.40

def predict_intent_tag(message):
  message = clean_corpus([message])
  X_test = vectorizer.transform(message)
  y = model.predict(X_test.toarray())
  # if probability of all intent is low, classify it as noanswer
  if y.max() < INTENT_NOT_FOUND_THRESHOLD:
    return 'noanswer'

  prediction = np.zeros_like(y[0])
  prediction[y.argmax()] = 1
  tag = encoder.inverse_transform([prediction])[0][0]
  return tag

print(predict_intent_tag('How you could help me?'))
print(predict_intent_tag('swiggy chat bot'))
print(predict_intent_tag('Where\'s my order'))
```

Define function to fetch the tag of the intent

```
import random
import time

def get_intent(tag):
  # to return complete intent from intent tag
  for intent in intents['intents']:
    if intent['tag'] == tag:
      return intent
```

#4

Intent Classification

Till now we have fetched the intent of the message received from the user. Now let's define a function to perform a certain action on the basis of the intent classified

```
def perform_action(action_code, intent):
  # funition to perform an action which is required by intent

  if action_code == 'CHECK_ORDER_STATUS':
    print('\n Checking database \n')
    time.sleep(2)
    order_status = ['in kitchen', 'with delivery executive']
    delivery_time = []
    return {'intent-tag':intent['next-intent-tag'][0],
            'order_status': random.choice(order_status),
            'delivery_time': random.randint(10, 30)}

  elif action_code == 'ORDER_CANCEL_CONFIRMATION':
    ch = input('BOT: Do you want to continue (Y/n) ?')
    if ch == 'y' or ch == 'Y':
      choice = 0
    else:
      choice = 1
    return {'intent-tag':intent['next-intent-tag'][choice]}

  elif action_code == 'ADD_DELIVERY_INSTRUCTIONS':
    instructions = input('Your Instructions: ')
    return {'intent-tag':intent['next-intent-tag'][0]}
```

#5

Complete Chatbot

Using all the functions which we have defined to classify intent and perform action accordingly are integrated together to develop our final chatbot.

```
while True:
  # get message from user
  message = input('You: ')
  # predict intent tag using trained neural network
  tag = predict_intent_tag(message)
  # get complete intent from intent tag
  intent = get_intent(tag)
  # generate random response from intent
  response = random.choice(intent['responses'])
  print('Bot: ', response)

  # check if there's a need to perform some action
  if 'action' in intent.keys():
    action_code = intent['action']
    # perform action
    data = perform_action(action_code, intent)
    # get follow up intent after performing action
    followup_intent = get_intent(data['intent-tag'])
    # generate random response from follow up intent
    response = random.choice(followup_intent['responses'])

    # print randomly selected response
    if len(data.keys()) > 1:
      print('Bot: ', response.format(**data))
    else:
      print('Bot: ', response)

  # break loop if intent was goodbye
  if tag == 'goodbye':
    break
```

Learn

Knowledge Discovery through Brand Stories

Visit Us at :
techlearn.live

Email : support@techlearn.live
Phone : +91-9154796743