```
!pip install tflearn
```

```
Collecting tflearn
  Downloading tflearn-0.5.0.tar.gz (107 kB)
     |████████████████████████████████| 107 kB 7.7 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from tflearn) (1.21.6)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from tflearn) (1.15.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from tflearn) (7.1.2)
Building wheels for collected packages: tflearn
  Building wheel for tflearn (setup.py) ... done
  Created wheel for tflearn: filename=tflearn-0.5.0-py3-none-any.whl size=127299 sha256=51d9c9de279d94936026f99e088dc29b9e4de58efc7
  Stored in directory: /root/.cache/pip/wheels/5f/14/2e/1d8e28cc47a5a931a2fb82438c9e37ef9246cc6a3774520271
Successfully built tflearn
Installing collected packages: tflearn
Successfully installed tflearn-0.5.0
```

```
import random
from textblob import TextBlob
import numpy
import tflearn
import tensorflow
import json
import pickle
import nltk
import spacy
nlp = spacy.load('en_core_web_sm')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tensorflow/python/compat/v2_compat.py:107: disable_resource_variables
Instructions for updating:
non-resource variables are not supported in the long term
```

```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

```
cd /content/drive/MyDrive/shilpa/
```

```
/content/drive/MyDrive/shilpa
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
with open("bsk.json") as file:
  data = json.load(file)
```

```
words = []
labels = []
docs_x = []
docs_y = []
for intent in data["intents"]:
    for pattern in intent["patterns"]:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        docs_x.append(wrds)
        docs_y.append(intent["tag"])
    #print(f'docs_x: {docs_x}, docs_y: {docs_y}, Words: {words}')


    if intent["tag"] not in labels:
        labels.append(intent["tag"])
    #print(f'labels: {labels}')

words = [w.lower() for w in words]
words = sorted(list(set(words)))
labels = sorted(labels)
training = []
output = []
out_empty = [0 for _ in range(len(labels))]


for x, doc in enumerate(docs_x):
  bag = []
  wrds = [w.lower() for w in doc]
```

```
    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)

    output_row = out_empty[:]
    output_row[labels.index(docs_y[x])] = 1

    training.append(bag)
    output.append(output_row)

training = numpy.array(training)
output = numpy.array(output)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-0aeb1cf47677> in <module>()
      3 docs_x = []
      4 docs_y = []
----> 5 for intent in data["intents"]:
      6     for pattern in intent["patterns"]:
      7         wrds = nltk.word_tokenize(pattern)

NameError: name 'data' is not defined
```

SEARCH STACK OVERFLOW

```
tensorflow.compat.v1.reset_default_graph()

net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/tflearn/initializations.py:165: calling TruncatedNormal.__init__ (fro
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
```

```
model = tflearn.DNN(net)
model.fit(training, output, n_epoch=600, batch_size=8, show_metric=True)
model.save("model.tflearn")
```

```
Training Step: 2999  | total loss: 0.05230 | time: 0.015s
| Adam | epoch: 600 | loss: 0.05230 - acc: 1.0000 -- iter: 32/33
Training Step: 3000  | total loss: 0.05162 | time: 0.022s
| Adam | epoch: 600 | loss: 0.05162 - acc: 1.0000 -- iter: 33/33
--
INFO:tensorflow:/content/drive/MyDrive/NLP/datasets/model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

```
model = tflearn.DNN(net)
model.load("model.tflearn")
```

```
INFO:tensorflow:Restoring parameters from /content/drive/MyDrive/NLP/datasets/model.tflearn
```

```
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [word.lower() for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)


def recommend_food(pref, flag=True):
    results = model.predict([bag_of_words(pref, words)])
    results_index = numpy.argmax(results)
    tag = labels[results_index]
    if results[0][results_index]> 0.7:
      for tg in data["intents"]:
        if tg['tag'] == tag:
```

```
            responses = tg['responses']
        food = random.choice(responses)
        if flag == True:
          print(f"I would recommend you to try {food}")
        return food
      else :
        if flag == True:
          print("Sorry I didn't get it")
        return ""



place_name = "BCK"


list_of_work = ['Order', 'Order status']
customer_preferences = ["feel like heaven", "spicy", "green and healthy"]


print('Please enter your name : ')
ans = input()
print(f'Hello {ans} Welcome to {place_name}, How can I help you today? ')
print(f'For best results select from below ')

print(f'1. I want to {list_of_work[0]}')
print(f'2. I want to know my {list_of_work[1]}')
ans = input()
if '1' == ans :
  print(f"Please select the option best suited ")
  print(f'1. Full Menu')
  print(f'2. Customized Menu')
  ans = input()
  if '1' == ans:
    print(f"Display full Menu")
  elif '2' == ans:
    print("Express your requirement of food")
    print(f'For best results select from below ')
    print(f'1. Food should {customer_preferences[0]}')
    print(f'2. Food should be {customer_preferences[1]}')
    print(f'3. Food should be {customer_preferences[2]}')
    ans = input()
    if '1' == ans:
      ans = customer_preferences[0]
    elif '2' == ans:
      ans = customer_preferences[1]
    elif '3' == ans:
      ans = customer_preferences[2]
    f = recommend_food(ans)
    if f == "":
      print("Please try again using the options mentioned above")
    else:
      print('Shall I confirm your order !!!')
      print('1. Yes')
      print('2. No')
      ans = input()
      if '1' == ans:
        print(f'Your order for {f} is confirmed')
        print("...")
        print("We'll get your food ready in 20 minutes")
      elif '2' == ans:
        print("I suggest you should checkout our full menu ")

      print(f"Kindly express your thoughts about {f}")
      ans = input()
      rev = recommend_food(ans,flag=False)
      if rev==="":
        rev = TextBlob(ans)
        if rev.polarity >0.2:
          print(f"I am glad that you enjoyed our food")
        else:
          print(f"Sorry to hear that we'll try to get better next time")
      else:
        print(f"{rev}")

elif '2' == ans:
  print(f"Can I have your order number ")
  ans = input()
  ## Here we can connect to database or any system to check estimated order time
  # and return that, for now lets return 10 minutes
  print(f"....")
  print(f"Order {ans} should be ready in 10 minutes")
  ## Conversation ends
```

```
else:
  doc = nlp(ans.lower())
  # Check if object is order
  # Assuming that user wants to order
  subtree =[]
  for token in doc:
    if ("dobj" in token.dep_):
      subtree = list(token.subtree)
  if "order" in str(subtree):
    print(f"Here is our menu ")
  else:
    print(f"Sorry, I do not understand please try again")
```

```
Please enter your name :
Manasvi
Hello Manasvi Welcome to BCK, How can I help you today?
For best results select from below
1. I want to Order
2. I want to know my Order status
1
Please select the option best suited
1. Full Menu
2. Customized Menu
2
Express your requirement of food
For best results select from below
1. Food should feel like heaven
2. Food should be spicy
3. Food should be green and healthy
1
I would recommend you to try Chicken Triple Rice With Gravy
Shall I confirm your order !!!
1. Yes
2. No
1
Your order for Chicken Triple Rice With Gravy is confirmed
...
We'll get your food ready in 20 minutes
Kindly express your thoughts about Chicken Triple Rice With Gravy
I enjoyed it very much
I am glad that you enjoyed our food
```

```
print("Express your requirement of food")
ans = input()
f = recommend_food(ans)
if f == "":
  print("Please try again using the options mentioned above")
else:
  print('Shall I confirm your order !!!')
  print('1. Yes')
  print('2. No')
```

```
Express your requirement of food
low price
I would recommend you to try Veg Hakka Noodles
Shall I confirm your order !!!
1. Yes
2. No
```

```
sent = "need my order"
doc = nlp(sent.lower
```

```
for token in doc:
  if ("dobj" in token.dep_):
      subtree = list(token.subtree)

if "order" in str(subtree):
    print(f"Here is our menu ")
else:
    print(f"Sorry, I do not understand please try again")
```

```
Here is our menu
```

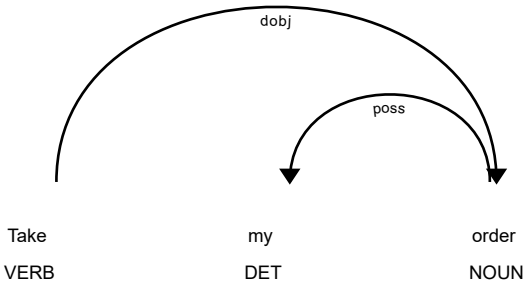
```
for token in doc:
  print(f" {token.text} {token.pos_} {token.dep_}")
```

```
 need VERB ROOT
 my DET poss
 order NOUN dobj
```

```
for np in doc.noun_chunks:
    print(np.root.text)
    print(np.text)
```

```
order
my order
```

```
from spacy import displacy
displacy.render(doc, style='dep', jupyter=True)
```




```
from spacy.symbols import VERB, dobj
for np in doc.noun_chunks:
    if np.root.dep == dobj:
        print(np.root.text)
```

```
order
```

```
print(doc.noun_chunks)
```

```
<generator object at 0x7f4b1ddf4410>
```

```
def get_object_phrase(doc):
  for token in doc:
    if ("ROOT" in token.dep_):
      subtree = list(token.subtree)
      print(f"Tree {subtree}")

      start = subtree[0].i
      print(start)
      end = subtree[-1].i + 1
      return doc[start:end]
```

```
get_object_phrase(doc)
```

```
Tree [take, my, order]
0
take my order
```

```
print(li)
```

```
my order
```

```
TextBlob("Extremely hungry").sentiment
```

```
Sentiment(polarity=-0.125, subjectivity=1.0)
```