```python
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session

import pandas as pd

import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle

import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

import numpy as np
import tensorflow as tf
from keras.models import Sequential
from keras.layers import GlobalMaxPooling1D, Dense, Activation,
Dropout, Embedding,Conv1D
import random
from keras.models import load_model
from keras.preprocessing.sequence import pad_sequences
```

```python
#loading file json
def load_json_file(filename):
    with open(filename) as f:
        file = json.load(f)
    return file

filename = '/kaggle/input/friends/intents.json'

intents = load_json_file(filename)

intents
```

```
{'intents': [{'tag': 'greeting',
   'patterns': ['Hi',
    'How are you?',
    'Is anyone there?',
    'Hello',
    'Good day',
    "What's up",
    'how are ya',
    'heyy',
    'whatsup',
    '??? ??? ??'],
   'responses': ['Hello!',
    'Good to see you again!',
    'Hi there, how can I help?'],
   'context_set': ''},
  {'tag': 'goodbye',
   'patterns': ['cya',
    'see you',
    'bye bye',
    'See you later',
    'Goodbye',
    'I am Leaving',
    'Bye',
    'Have a Good day',
    'talk to you later',
    'ttyl',
    'i got to go',
    'gtg'],
   'responses': ['Sad to see you go :(',
    'Talk to you later',
    'Goodbye!',
    'Come back soon'],
   'context_set': ''},
  {'tag': 'creator',
   'patterns': ['what is the name of your developers',
    'what is the name of your creators',
    'what is the name of the developers',
    'what is the name of the creators',
    'who created you',
    'your developers',
    'your creators',
    'who are your developers',
    'developers',
    'you are made by',
    'you are made by whom',
    'who created you',
    'who create you',
    'creators',
    'who made you',
    'who designed you'],
   'responses': ['office students'],
   'context_set': ''},
  {'tag': 'name',
   'patterns': ['name',
    'your name',
    'do you have a name',
    'what are you called',
    'what is your name',
    'what should I call you',
    'whats your name?',
    'what are you',
    'who are you',
    'who is this',
    'what am i chatting to',
    'who am i taking to',
    'what are you'],
   'responses': ['You can call me Mind Reader.',
    "I'm Mind Reader",
    'I am a Chatbot.',
    'I am your helper'],
   'context_set': ''},
  {'tag': 'hours',
   'patterns': ['timing of office',
    'what is office timing',
    'working days',
    'when are you guys open',
    'what are your hours',
    'hours of operation',
    'when is the office open',
    'office timing',
    'what about office timing',
    'is office open on saturday',
    'tell something about office timing',
    'what is the office  hours',
    'when should i come to office',
    'when should i attend office',
    'what is my office time',
    'office timing',
    'timing office'],
   'responses': ['office is open 8am-5pm Monday-Saturday!'],
   'context_set': ''},
  {'tag': 'number',
   'patterns': ['more info',
    'contact info',
    'how to contact office',
    'office telephone number',
    'office number',
    'What is your contact no',
    'Contact number?',
    'how to call you',
    'office phone no?',
    'how can i contact you',
    'Can i get your phone number',
    'how can i call you',
    'phone number',
    'phone no',
    'call'],
   'responses': ['You can contact at: NUMBER'],
   'context_set': ''},
  {'tag': 'course',
   'patterns': ['list of courses',
    'list of courses offered',
    'list of courses offered in',
    'what are the courses offered in your office?',
    'courses?',
    'courses offered',
    'courses offered in (your univrsity(UNI) name)',
    'courses you offer',
    'branches?',
    'courses available at UNI?',
    'branches available at your office?',
    'what are the courses in UNI?',
    'what are branches in UNI?',
    'what are courses in UNI?',
    'branches available in UNI?',
    'can you tell me the courses available in UNI?',
    'can you tell me the branches available in UNI?',
    'computer engineering?',
    'computer',
    'Computer engineering?',
    'it',
    'IT',
    'Information Technology',
    'AI/Ml',
    'Mechanical engineering',
    'Chemical engineering',
    'Civil engineering'],
   'responses': ['Our rganization offers Information Technology,
computer Engineering, Mechanical engineering,Chemical engineering,
Civil engineering and extc Engineering.'],
   'context_set': ''},
  {'tag': 'fees',
   'patterns': ['information about fee',
    'information on fee',
    'tell me the fee',
    'office fee',
    'fee per semester',
    'what is the fee of each semester',
    'what is the fees of each year',
    'what is fee',
    'what is the fees',
    'how much is the fees',
    'fees for first year',
    'fees',
    'about the fees',
    'tell me something about the fees',
    'What is the fees of hostel',
    'how much is the fees',
    'hostel fees',
    'fees for AC room',
    'fees for non-AC room',
    'fees for Ac room for girls',
    'fees for non-Ac room for girls',
    'fees for Ac room for boys',
    'fees for non-Ac room for boys'],
   'responses': ['For Fee detail visit <a target="_blank" href="LINK">
here</a>'],
   'context_set': ''},
  {'tag': 'location',
   'patterns': ['where is the office located',
    'office is located at',
    'where is office',
    'where is office located',
    'address of office',
    'how to reach office',
    'office location',
    'office address',
    'wheres the office',
    'how can I reach office',
    'whats is the office address',
    'what is the address of office',
    'address',
    'location'],
   'responses': ['<a target="_blank" href="ADD YOU GOOGLE MAP LINK
HERE"> here</a>'],
   'context_set': ''},
  {'tag': 'hostel',
   'patterns': ['hostel facility',
    'hostel servive',
    'hostel location',
    'hostel address',
    'hostel facilities',
    'hostel fees',
    'Does office provide hostel',
    'Is there any hostel',
    'Where is hostel',
    'do you have hostel',
    'do you guys have hostel',
    'hostel',
    'hostel capacity',
    'what is the hostel fee',
    'how to get in hostel',
    'what is the hostel address',
    'how far is hostel from office',
    'hostel office distance',
    'where is the hostel',
    'how big is the hostel',
    'distance between office and hostel',
    'distance between hostel and office'],
   'responses': ['For hostel detail visit <a target="_blank" href="ADD
YOUR HOSTEL DETAIL PDF LINK OR ANY INFORMATION LINK OR ADD YOU OWN
ANSWERS"> here</a>'],
   'context_set': ''},
  {'tag': 'event',
   'patterns': ['events organised',
    'list of events',
    'list of events organised in office',
    'list of events conducted in office',
    'What events are conducted in office',
    'Are there any event held at office',
    'Events?',
    'functions',
    'what are the events',
    'tell me about events',
    'what about events'],
   'responses': ['For event detail visit <a target="_blank" href="ADD
YOUR FUNCTIONS LINK OR YOUR OWN RESPONSE"> here</a>'],
   'context_set': ''},
  {'tag': 'document',
   'patterns': ['document to bring',
    'documents needed for admision',
    'documents needed at the time of admission',
    'documents needed during admission',
    'documents required for admision',
    'documents required at the time of admission',
    'documents required during admission',
    'What document are required for admission',
    'Which document to bring for admission',
    'documents',
    'what documents do i need',
    'what documents do I need for admission',
    'documents needed'],
   'responses': ['To know more about document required visit <a
target="_blank" href="ADD LINK OF ADMISSION GUIDANCE DOCUMENT FROM
YOUR UNIVERSITY WEBSITE"> here</a>'],
   'context_set': ''},
  {'tag': 'floors',
   'patterns': ['size of campus',
    'building size',
    'How many floors does office have',
    'floors in office',
    'floors in office',
    "how tall is UNI's office of Engineering office building",
    'floors'],
   'responses': ['My office has total 2 floors '],
   'context_set': ''},
  {'tag': 'syllabus',
   'patterns': ['Syllabus for IT',
    'what is the Information Technology syllabus',
    'syllabus',
    'timetable',
    'what is IT syllabus',
    'syllabus',
    'What is next lecture'],
   'responses': ['Timetable provide direct to the students OR To know
about syllabus visit <a target="_blank" href="TIMETABLE LINK">
here</a>'],
   'context_set': ''},
  {'tag': 'library',
   'patterns': ['is there any library',
    'library facility',
    'library facilities',
    'do you have library',
    'does the office have library facility',
    'office library',
    'where can i get books',
    'book facility',
    'Where is library',
    'Library',
    'Library information',
    'Library books information',
    'Tell me about library',
    'how many libraries'],
   'responses': ['There is one huge and spacious library.timings are
8am to 6pm and for more visit <a target="blank" href="ADD LIBRARY
DETAIL LINK">here</a>'],
   'context_set': ''},
  {'tag': 'infrastructure',
   'patterns': ['how is office infrastructure',
    'infrastructure',
    'office infrastructure'],
   'responses': ['Our University has Excellent Infrastructure. Campus
is clean. Good IT Labs With Good Speed of Internet connection'],
   'context_set': ''},
  {'tag': 'canteen',
   'patterns': ['food facilities',
    'canteen facilities',
    'canteen facility',
    'is there any canteen',
    'Is there a cafetaria in office',
    'Does office have canteen',
    'Where is canteen',
    'where is cafetaria',
    'canteen',
    'Food',
    'Cafetaria'],
   'responses': ['Our organization has canteen with variety of food
available'],
   'context_set': ''},
  {'tag': 'menu',
   'patterns': ['food menu',
    'food in canteen',
    'Whats there on menu',
    'what is available in office canteen',
    'what foods can we get in office canteen',
    'food variety',
    'What is there to eat?'],
   'responses': ['we serve Franky, Locho, Alu-puri, Kachori, Khavsa,
Thaali and many more on menu'],
   'context_set': ''},
  {'tag': 'placement',
   'patterns': ['What is office placement',
    'Which companies visit in office',
    'What is average package',
    'companies visit',
    'package',
    'About placement',
    'placement',
    'recruitment',
    'companies'],
   'responses': ['To know about placement visit <a target="_blank"
href="PLACEMENT INFORMATION LINK FROM YOUR UNIVERSITY WEBSITE IF THEY
HAVE">here</a>'],
   'context_set': ''},
  {'tag': 'ithod',
   'patterns': ['Who is HOD', 'Where is HOD', 'it hod', 'name of it
hod'],
   'responses': ['All engineering departments have only one hod XYZ
who available on (Place name)'],
   'context_set': ''},
  {'tag': 'computerhod',
   'patterns': ['Who is computer HOD',
    'Where is computer HOD',
    'computer hod',
    'name of computer hod'],
   'responses': ['All engineering departments have only one hod XYZ
who available on (PLACE NAME)'],
   'context_set': ''},
  {'tag': 'extchod',
   'patterns': ['Who is extc HOD',
    'Where is  extc HOD',
    'extc hod',
    'name of extc hod'],
   'responses': ['Different school wise hod are different.So be more
clear with your school or department'],
   'context_set': ''},
  {'tag': 'principal',
   'patterns': ['what is the name of principal',
    'whatv is the principal name',
    'principal name',
    'Who is office principal',
    "Where is principal's office",
    'principal',
    'name of principal'],
   'responses': ['XYZ is office principal and if you need any help
then call your branch hod first.That is more appropriate'],
   'context_set': ''},
  {'tag': 'sem',
   'patterns': ['exam dates',
    'exam schedule',
    'When is semester exam',
    'Semester exam timetable',
    'sem',
    'semester',
    'exam',
    'when is exam',
    'exam timetable',
    'exam dates',
    'when is semester'],
   'responses': ['Here is the Academic Calendar  <a target="_blank" 
href="YOUR ACADEMIC CALENDER">website</a>'],
   'context_set': ''},
  {'tag': 'admission',
   'patterns': ['what is the process of admission',
    'what is the admission process',
    'How to take admission in your office',
    'What is the process for admission',
    'admission',
    'admission process'],
   'responses': ['Application can also be submitted online through the 
Unversity\'s  <a target="_blank" href="LINK OF ADMISSION 
DOCUMENT">website</a>'],
   'context_set': ''},
  {'tag': 'scholarship',
   'patterns': ['scholarship',
    'Is scholarship available',
    'scholarship engineering',
    'scholarship it',
    'scholarship ce',
    'scholarship mechanical',
    'scholarship civil',
    'scholarship chemical',
    'scholarship for AI/ML',
    'available scholarships',
    'scholarship for computer engineering',
    'scholarship for IT engineering',
    'scholarship for mechanical engineering',
    'scholarship for civil engineering',
    'scholarship for chemical engineering',
    'list of scholarship',
    'comps scholarship',
    'IT scholarship',
    'mechanical scholarship',
    'civil scholarship',
    'chemical scholarship',
    'automobile scholarship',
    'first year scholarship',
    'second year scholarship',
    'third year scholarship',
    'fourth year scholarship'],
   'responses': ['Many government scholarships are supported by our 
rganization. For details and updates visit <a target="_blank" 
href="(SCHOLARSHIP DETAILS LINK)">here</a>'],
   'context_set': ''},
  {'tag': 'facilities',
   'patterns': ['What facilities office provide',
    'office facility',
    'What are office facilities',
    'facilities',
    'facilities provided'],
   'responses': ["Our rganization's Engineering department provides
fully AC Lab with internet connection, smart classroom, Auditorium,
library,canteen"],
   'context_set': ''},
  {'tag': 'office intake',
   'patterns': ['max number of students',
    'number of seats per branch',
    'number of seats in each branch',
    'maximum number of seats',
    'maximum students intake',
    'What is office intake',
    'how many stundent are taken in each branch',
    'seat allotment',
    'seats'],
   'responses': ['For IT, Computer and extc 60 per branch and seat may
be differ for different department.'],
   'context_set': ''},
  {'tag': 'uniform',
   'patterns': ['office dress code',
    'office dresscode',
    'what is the uniform',
    'can we wear casuals',
    'Does office have an uniform',
    'Is there any uniform',
    'uniform',
    'what about uniform',
    'do we have to wear uniform'],
   'responses': ['ENTER YOUR OWN UNIVERSITY UNIFORM CIRCULER'],
   'context_set': ''},
  {'tag': 'committee',
   'patterns': ['what are the different committe in office',
    'different committee in office',
    'Are there any committee in office',
    'Give me committee details',
    'committee',
    'how many committee are there in office'],
   'responses': ['For the various committe in office contact this
number: ADD NUMBER'],
   'context_set': ''},
  {'tag': 'random',
   'patterns': ['I love you', 'Will you marry me', 'Do you love me'],
   'responses': ['I am not program for this, please ask appropriate
query'],
   'context_set': ''},
  {'tag': 'swear',
   'patterns': ['fuck',
    'bitch',
    'shut up',
    'hell',
    'stupid',
    'idiot',
    'dumb ass',
    'asshole',
    'fucker'],
   'responses': ['please use appropriate language',
    'Maintaining decency would be appreciated'],
   'context_set': ''},
  {'tag': 'vacation',
   'patterns': ['holidays',
    'when will semester starts',
    'when will semester end',
    'when is the holidays',
    'list of holidays',
    'Holiday in these year',
    'holiday list',
    'about vacations',
    'about holidays',
    'When is vacation',
    'When is holidays',
    'how long will be the vacation'],
   'responses': ['Academic calender is given to you by your class-
soordinators after you join your respective classes'],
   'context_set': ''},
  {'tag': 'sports',
   'patterns': ['sports and games',
    'give sports details',
    'sports infrastructure',
    'sports facilities',
    'information about sports',
    'Sports activities',
    'please provide sports and games information'],
   'responses': ['Our rganization encourages all-round development of
students and hence provides sports facilities in the campus. For more
details visit<a target="_blank" href=/"(LINK IF HAVE)">here</a>'],
   'context_set': ''},
  {'tag': 'salutaion',
   'patterns': ['okk',
    'okie',
    'nice work',
    'well done',
    'good job',
    'thanks for the help',
    'Thank You',
    'its ok',
    'Thanks',
    'Good work',
    'k',
    'ok',
    'okay'],
   'responses': ['I am glad I helped you',
    'welcome, anything else i can assist you with?'],
   'context_set': ''},
  {'tag': 'task',
   'patterns': ['what can you do',
    'what are the thing you can do',
    'things you can do',
    'what can u do for me',
    'how u can help me',
    'why i should use you'],
   'responses': ['I can answer to low-intermediate questions regarding
office',
    'You can ask me questions regarding office, and i will try to
answer them'],
   'context_set': ''},
  {'tag': 'ragging',
   'patterns': ['ragging',
    'is ragging practice active in office',
    'does office have any antiragging facility',
    'is there any ragging cases',
    'is ragging done here',
    'ragging against',
    'antiragging facility',
    'ragging juniors',
    'ragging history',
    'ragging incidents'],
   'responses': ['We are Proud to tell you that our office provides
ragging free environment, and we have strict rules against ragging'],
   'context_set': ''},
  {'tag': 'hod',
   'patterns': ['hod', 'hod name', 'who is the hod'],
   'responses': ['HODs differ for each branch, please be more specific
like: (HOD it)'],
   'context_set': ''},
  {'tag': 'leave',
   'patterns': ['i want leave for two days',
    'i need leave',
    'give me a leave',
    'can you give me a leave',
    'will you please give permission'],
   'responses': ['How many days you need to take leave'],
   'context_set': ''},
  {'tag': 'days',
   'patterns': ['2 days',
    '3 days',
    '5 days',
    'one week',
    'two week',
    'Five days',
    'two days',
    'tommorrow',
    'day after tomorrow'],
   'responses': ['ok i will mail the  project manager'],
   'context_set': ''}]}

# Create an empty DataFrame
df = pd.DataFrame(columns=['Pattern', 'Tag'])

df

Empty DataFrame
Columns: [Pattern, Tag]
Index: []

def extract_json_info(json_file, df):
    # Iterate over each intent in the JSON file
    for intent in json_file['intents']:
        # Iterate over each pattern in the current intent
        for pattern in intent['patterns']:
            # Create a list containing the pattern and its associated 
tag
            sentence_tag = [pattern, intent['tag']]
            # Append the pattern and tag to the DataFrame
            df.loc[len(df.index)] = sentence_tag
     # Return the updated DataFrame
    return df

df = extract_json_info(intents, df)
df.head(20)

              Pattern       Tag
0                  Hi  greeting
1        How are you?  greeting
2    Is anyone there?  greeting
3               Hello  greeting
4             Good day  greeting
5            What's up  greeting
6           how are ya  greeting
7                heyy  greeting
8             whatsup  greeting
9          ??? ??? ??  greeting
10                cya   goodbye
11            see you   goodbye
12            bye bye   goodbye
13      See you later   goodbye
14            Goodbye   goodbye
```

```
15       I am Leaving   goodbye
16                Bye   goodbye
17    Have a Good day   goodbye
18  talk to you later   goodbye
19               ttyl   goodbye
```

```
#create list of unique tags from the 'Tag' colum
labels = df['Tag'].unique().tolist()
labels = [s.strip() for s in labels]

labels
```

```
['greeting',
 'goodbye',
 'creator',
 'name',
 'hours',
 'number',
 'course',
 'fees',
 'location',
 'hostel',
 'event',
 'document',
 'floors',
 'syllabus',
 'library',
 'infrastructure',
 'canteen',
 'menu',
 'placement',
 'ithod',
 'computerhod',
 'extchod',
 'principal',
 'sem',
 'admission',
 'scholarship',
 'facilities',
 'office intake',
 'uniform',
 'committee',
 'random',
 'swear',
 'vacation',
 'sports',
 'salutaion',
 'task',
 'ragging',
 'hod',
 'leave',
 'days']
```

```
df.shape
```

```
(419, 2)
```

```
len(df)
```

```
419
```

```
#info about dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 419 entries, 0 to 418
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   Pattern  419 non-null    object
 1   Tag      419 non-null    object
dtypes: object(2)
memory usage: 9.8+ KB
```

```
#count each tag
tag_counts = df['Tag'].value_counts()
print(tag_counts)
```

```
Tag
course                 27
scholarship            26
fees                   23
hostel                 22
hours                  17
creator                16
number                 15
library                14
location               14
salutaion              13
document               13
name                   13
vacation               12
goodbye                12
sem                    11
event                  11
canteen                11
ragging                10
greeting               10
swear                   9
uniform                 9
office intake           9
```

```
days                 9
placement            9
principal            7
floors               7
syllabus             7
sports               7
menu                 7
committee            6
admission            6
task                 6
facilities           5
leave                5
ithod                4
extchod              4
computerhod          4
random               3
infrastructure       3
hod                  3
Name: count, dtype: int64
```

```
len(labels)
```

```
40
```

```
# Tokenize the text patterns
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['Pattern'])

# Convert text patterns to numerical sequences
X = tokenizer.texts_to_sequences(df['Pattern'])

# Pad sequences to ensure uniform length
max_sequence_length = max(len(seq) for seq in X)
X_padded = pad_sequences(X, maxlen=max_sequence_length,
padding='post')

# Convert tags to numerical labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['Tag'])
```

```
X_padded
```

```
array([[166,   0,   0, ...,   0,   0,   0],
       [ 12,   9,   5, ...,   0,   0,   0],
       [  1, 167,  25, ...,   0,   0,   0],
       ...,
       [115,  50,   0, ...,   0,   0,   0],
       [289,   0,   0, ...,   0,   0,   0],
       [ 92, 290, 291, ...,   0,   0,   0]], dtype=int32)
```

```
y
```

```
array([14, 14, 14, 14, 14, 14, 14, 14, 14, 14, 13, 13, 13, 13, 13, 13,
13,
       13, 13, 13, 13, 13,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,  5,
5,
        5,  5,  5,  5, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24, 24,
24,
       17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,
17,
       25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,  4,
4,
        4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,  4,
4,
        4,  4,  4,  4,  4,  4,  4,  4, 11, 11, 11, 11, 11, 11, 11, 11,
11,
       11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 22, 22,
22,
       22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 16, 16, 16, 16, 16,
16,
       16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16,
8,
        8,  8,  8,  8,  8,  8,  8,  8,  8,  8,  7,  7,  7,  7,  7,  7,
7,
        7,  7,  7,  7,  7,  7, 12, 12, 12, 12, 12, 12, 12, 36, 36, 36,
36,
       36, 36, 36, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
21,
       18, 18, 18,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1, 23, 23,
23,
       23, 23, 23, 23, 27, 27, 27, 27, 27, 27, 27, 27, 27, 19, 19, 19,
19,
        3,  3,  3,  3,  9,  9,  9,  9, 28, 28, 28, 28, 28, 28, 28, 33,
33,
       33, 33, 33, 33, 33, 33, 33, 33, 33,  0,  0,  0,  0,  0,  0, 32,
32,
       32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32,
32,
       32, 32, 32, 32, 32, 32, 32, 10, 10, 10, 10, 10, 26, 26, 26, 26,
26,
       26, 26, 26, 26, 38, 38, 38, 38, 38, 38, 38, 38, 38,  2,  2,  2,
2,
        2,  2, 30, 30, 30, 35, 35, 35, 35, 35, 35, 35, 35, 35, 39, 39,
39,
       39, 39, 39, 39, 39, 39, 39, 39, 39, 34, 34, 34, 34, 34, 34, 34,
31,
       31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 37, 37, 37, 37,
37,
       37, 29, 29, 29, 29, 29, 29, 29, 29, 29, 29, 15, 15, 15, 20, 20,
20,
       20, 20,  6,  6,  6,  6,  6,  6,  6,  6,  6])
```

```
print(X_padded.shape)
print(y.shape)
```

```
(419, 9)
(419,)
```

```
# Split the data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_padded, y,
test_size=0.2, random_state=42)

model = Sequential()
model.add(Dense(128, input_shape=(len(X_train[0]),),
activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(label_encoder.classes_), activation='softmax'))

# Compile the model
model.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, validation_data=(X_val, y_val),
epochs=200, batch_size=32)
```

```
Epoch 1/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.0204 - loss:
43.2093 - val_accuracy: 0.0000e+00 - val_loss: 15.5541
Epoch 2/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0294 - loss:
28.2782 - val_accuracy: 0.0000e+00 - val_loss: 9.0889
Epoch 3/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0426 - loss:
21.5787 - val_accuracy: 0.0238 - val_loss: 6.4169
Epoch 4/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0359 - loss:
17.8269 - val_accuracy: 0.0119 - val_loss: 5.0289
Epoch 5/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0130 - loss:
13.2501 - val_accuracy: 0.0119 - val_loss: 4.1550
Epoch 6/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0488 - loss:
11.3660 - val_accuracy: 0.0119 - val_loss: 3.8243
Epoch 7/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0142 - loss:
9.7166 - val_accuracy: 0.0119 - val_loss: 3.7861
Epoch 8/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0369 - loss:
7.8412 - val_accuracy: 0.0119 - val_loss: 3.7340
Epoch 9/200
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0555 - loss:
7.2702 - val_accuracy: 0.0238 - val_loss: 3.6778
Epoch 10/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0301 - loss:
6.1955 - val_accuracy: 0.0119 - val_loss: 3.6595
Epoch 11/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0229 - loss:
6.2829 - val_accuracy: 0.0357 - val_loss: 3.6484
Epoch 12/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0409 - loss:
5.3777 - val_accuracy: 0.0476 - val_loss: 3.6439
Epoch 13/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0556 - loss:
5.2422 - val_accuracy: 0.0476 - val_loss: 3.6456
Epoch 14/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0516 - loss:
4.5257 - val_accuracy: 0.0476 - val_loss: 3.6503
Epoch 15/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0662 - loss:
4.6945 - val_accuracy: 0.0476 - val_loss: 3.6542
Epoch 16/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0644 - loss:
4.5203 - val_accuracy: 0.0476 - val_loss: 3.6576
Epoch 17/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0562 - loss:
4.3719 - val_accuracy: 0.0476 - val_loss: 3.6603
Epoch 18/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0508 - loss:
4.1781 - val_accuracy: 0.0476 - val_loss: 3.6618
Epoch 19/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0774 - loss:
4.0786 - val_accuracy: 0.0476 - val_loss: 3.6640
Epoch 20/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0574 - loss:
4.0788 - val_accuracy: 0.0476 - val_loss: 3.6650
Epoch 21/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0548 - loss:
4.2135 - val_accuracy: 0.0476 - val_loss: 3.6651
Epoch 22/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0720 - loss:
3.9434 - val_accuracy: 0.0476 - val_loss: 3.6652
Epoch 23/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0776 - loss:
3.9867 - val_accuracy: 0.0476 - val_loss: 3.6660
Epoch 24/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0547 - loss:
4.2585 - val_accuracy: 0.0476 - val_loss: 3.6662
Epoch 25/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0593 - loss:
```

```
4.0283 - val_accuracy: 0.0476 - val_loss: 3.6654
Epoch 26/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0680 - loss:
4.1468 - val_accuracy: 0.0476 - val_loss: 3.6666
Epoch 27/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0719 - loss:
3.8629 - val_accuracy: 0.0476 - val_loss: 3.6664
Epoch 28/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step - accuracy: 0.0540 - loss:
3.7310 - val_accuracy: 0.0476 - val_loss: 3.6664
Epoch 29/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0570 - loss:
3.9835 - val_accuracy: 0.0476 - val_loss: 3.6662
Epoch 30/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0707 - loss:
3.8130 - val_accuracy: 0.0476 - val_loss: 3.6654
Epoch 31/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0732 - loss:
3.7464 - val_accuracy: 0.0476 - val_loss: 3.6648
Epoch 32/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0571 - loss:
3.7602 - val_accuracy: 0.0476 - val_loss: 3.6638
Epoch 33/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0719 - loss:
4.0411 - val_accuracy: 0.0476 - val_loss: 3.6631
Epoch 34/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0614 - loss:
3.7065 - val_accuracy: 0.0476 - val_loss: 3.6630
Epoch 35/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0743 - loss:
3.7191 - val_accuracy: 0.0476 - val_loss: 3.6614
Epoch 36/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0839 - loss:
3.6280 - val_accuracy: 0.0476 - val_loss: 3.6610
Epoch 37/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0729 - loss:
3.7434 - val_accuracy: 0.0476 - val_loss: 3.6602
Epoch 38/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0732 - loss:
3.6807 - val_accuracy: 0.0476 - val_loss: 3.6597
Epoch 39/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0834 - loss:
3.7060 - val_accuracy: 0.0476 - val_loss: 3.6581
Epoch 40/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0881 - loss:
3.6307 - val_accuracy: 0.0476 - val_loss: 3.6574
Epoch 41/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0457 - loss:
3.6926 - val_accuracy: 0.0476 - val_loss: 3.6575
```

```
Epoch 42/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0669 - loss: 
3.6714 - val_accuracy: 0.0476 - val_loss: 3.6571
Epoch 43/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0494 - loss: 
3.6832 - val_accuracy: 0.0476 - val_loss: 3.6568
Epoch 44/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0693 - loss: 
3.7405 - val_accuracy: 0.0476 - val_loss: 3.6566
Epoch 45/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0664 - loss: 
3.7375 - val_accuracy: 0.0476 - val_loss: 3.6569
Epoch 46/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0761 - loss: 
3.6344 - val_accuracy: 0.0476 - val_loss: 3.6571
Epoch 47/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0733 - loss: 
3.7041 - val_accuracy: 0.0476 - val_loss: 3.6570
Epoch 48/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0711 - loss: 
3.6707 - val_accuracy: 0.0476 - val_loss: 3.6571
Epoch 49/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0768 - loss: 
3.6830 - val_accuracy: 0.0476 - val_loss: 3.6571
Epoch 50/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0674 - loss: 
3.7303 - val_accuracy: 0.0476 - val_loss: 3.6569
Epoch 51/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0776 - loss: 
3.6299 - val_accuracy: 0.0476 - val_loss: 3.6567
Epoch 52/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0698 - loss: 
3.6459 - val_accuracy: 0.0476 - val_loss: 3.6556
Epoch 53/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0596 - loss: 
3.7594 - val_accuracy: 0.0476 - val_loss: 3.6571
Epoch 54/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0533 - loss: 
3.6646 - val_accuracy: 0.0476 - val_loss: 3.6578
Epoch 55/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0770 - loss: 
3.6953 - val_accuracy: 0.0476 - val_loss: 3.6583
Epoch 56/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0706 - loss: 
3.6414 - val_accuracy: 0.0476 - val_loss: 3.6586
Epoch 57/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0648 - loss: 
3.6726 - val_accuracy: 0.0476 - val_loss: 3.6581
Epoch 58/200
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0798 - loss:
3.6408 - val_accuracy: 0.0476 - val_loss: 3.6582
Epoch 59/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0701 - loss:
3.7441 - val_accuracy: 0.0476 - val_loss: 3.6584
Epoch 60/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0814 - loss:
3.7053 - val_accuracy: 0.0476 - val_loss: 3.6582
Epoch 61/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0695 - loss:
3.6191 - val_accuracy: 0.0476 - val_loss: 3.6579
Epoch 62/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0570 - loss:
3.6177 - val_accuracy: 0.0476 - val_loss: 3.6573
Epoch 63/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0770 - loss:
3.5917 - val_accuracy: 0.0476 - val_loss: 3.6574
Epoch 64/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0913 - loss:
3.5682 - val_accuracy: 0.0476 - val_loss: 3.6578
Epoch 65/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0711 - loss:
3.6211 - val_accuracy: 0.0476 - val_loss: 3.6579
Epoch 66/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0691 - loss:
3.5941 - val_accuracy: 0.0476 - val_loss: 3.6577
Epoch 67/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0481 - loss:
3.6311 - val_accuracy: 0.0476 - val_loss: 3.6576
Epoch 68/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0616 - loss:
3.6112 - val_accuracy: 0.0476 - val_loss: 3.6578
Epoch 69/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0596 - loss:
3.5930 - val_accuracy: 0.0476 - val_loss: 3.6579
Epoch 70/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0731 - loss:
3.5954 - val_accuracy: 0.0476 - val_loss: 3.6581
Epoch 71/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0511 - loss:
3.5665 - val_accuracy: 0.0476 - val_loss: 3.6587
Epoch 72/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0790 - loss:
3.5651 - val_accuracy: 0.0476 - val_loss: 3.6587
Epoch 73/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0663 - loss:
3.5861 - val_accuracy: 0.0476 - val_loss: 3.6576
Epoch 74/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0682 - loss:
```

```
3.6920 - val_accuracy: 0.0476 - val_loss: 3.6582
Epoch 75/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0630 - loss:
3.5875 - val_accuracy: 0.0476 - val_loss: 3.6583
Epoch 76/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0757 - loss:
3.5719 - val_accuracy: 0.0476 - val_loss: 3.6574
Epoch 77/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0997 - loss:
3.5253 - val_accuracy: 0.0476 - val_loss: 3.6572
Epoch 78/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0675 - loss:
3.5647 - val_accuracy: 0.0476 - val_loss: 3.6538
Epoch 79/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0780 - loss:
3.5817 - val_accuracy: 0.0595 - val_loss: 3.6507
Epoch 80/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0455 - loss:
3.5937 - val_accuracy: 0.0595 - val_loss: 3.6476
Epoch 81/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0681 - loss:
3.5674 - val_accuracy: 0.0595 - val_loss: 3.6470
Epoch 82/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0698 - loss:
3.5950 - val_accuracy: 0.0595 - val_loss: 3.6403
Epoch 83/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0697 - loss:
3.5729 - val_accuracy: 0.0595 - val_loss: 3.6377
Epoch 84/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0733 - loss:
3.5912 - val_accuracy: 0.0595 - val_loss: 3.6317
Epoch 85/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0566 - loss:
3.5758 - val_accuracy: 0.0595 - val_loss: 3.6262
Epoch 86/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0870 - loss:
3.5906 - val_accuracy: 0.0595 - val_loss: 3.6175
Epoch 87/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0694 - loss:
3.5743 - val_accuracy: 0.0714 - val_loss: 3.6043
Epoch 88/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0603 - loss:
3.5990 - val_accuracy: 0.0714 - val_loss: 3.6049
Epoch 89/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0650 - loss:
3.5658 - val_accuracy: 0.0714 - val_loss: 3.6077
Epoch 90/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0683 - loss:
3.6125 - val_accuracy: 0.0595 - val_loss: 3.6131
```

```
Epoch 91/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0774 - loss:
3.5781 - val_accuracy: 0.0595 - val_loss: 3.6212
Epoch 92/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0541 - loss:
3.5717 - val_accuracy: 0.0595 - val_loss: 3.6219
Epoch 93/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0788 - loss:
3.5267 - val_accuracy: 0.0595 - val_loss: 3.6207
Epoch 94/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0598 - loss:
3.5959 - val_accuracy: 0.0595 - val_loss: 3.6190
Epoch 95/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0866 - loss:
3.5228 - val_accuracy: 0.0595 - val_loss: 3.6126
Epoch 96/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0799 - loss:
3.5658 - val_accuracy: 0.0714 - val_loss: 3.6095
Epoch 97/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0800 - loss:
3.5771 - val_accuracy: 0.0952 - val_loss: 3.6004
Epoch 98/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0658 - loss:
3.5449 - val_accuracy: 0.0952 - val_loss: 3.5987
Epoch 99/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0747 - loss:
3.5599 - val_accuracy: 0.0952 - val_loss: 3.5976
Epoch 100/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0663 - loss:
3.5548 - val_accuracy: 0.0833 - val_loss: 3.5980
Epoch 101/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0675 - loss:
3.5349 - val_accuracy: 0.0833 - val_loss: 3.5975
Epoch 102/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0636 - loss:
3.5627 - val_accuracy: 0.0952 - val_loss: 3.5966
Epoch 103/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0674 - loss:
3.5468 - val_accuracy: 0.0952 - val_loss: 3.5962
Epoch 104/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0747 - loss:
3.5293 - val_accuracy: 0.0952 - val_loss: 3.5947
Epoch 105/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0632 - loss:
3.5535 - val_accuracy: 0.0952 - val_loss: 3.5931
Epoch 106/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0893 - loss:
3.5492 - val_accuracy: 0.0952 - val_loss: 3.5892
Epoch 107/200
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0831 - loss:
3.5777 - val_accuracy: 0.0952 - val_loss: 3.5894
Epoch 108/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0652 - loss:
3.5489 - val_accuracy: 0.0952 - val_loss: 3.5892
Epoch 109/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0855 - loss:
3.5664 - val_accuracy: 0.0952 - val_loss: 3.5834
Epoch 110/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0784 - loss:
3.6005 - val_accuracy: 0.0952 - val_loss: 3.5809
Epoch 111/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0566 - loss:
3.5748 - val_accuracy: 0.0952 - val_loss: 3.5795
Epoch 112/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0647 - loss:
3.5542 - val_accuracy: 0.0833 - val_loss: 3.5787
Epoch 113/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0579 - loss:
3.5300 - val_accuracy: 0.0595 - val_loss: 3.5791
Epoch 114/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0939 - loss:
3.5372 - val_accuracy: 0.0595 - val_loss: 3.5846
Epoch 115/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0960 - loss:
3.5367 - val_accuracy: 0.0595 - val_loss: 3.5880
Epoch 116/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0737 - loss:
3.5316 - val_accuracy: 0.0595 - val_loss: 3.5871
Epoch 117/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0639 - loss:
3.6059 - val_accuracy: 0.0595 - val_loss: 3.5846
Epoch 118/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0799 - loss:
3.5176 - val_accuracy: 0.0595 - val_loss: 3.5826
Epoch 119/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0705 - loss:
3.5176 - val_accuracy: 0.0595 - val_loss: 3.5781
Epoch 120/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0702 - loss:
3.5347 - val_accuracy: 0.0595 - val_loss: 3.5742
Epoch 121/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0580 - loss:
3.5391 - val_accuracy: 0.0595 - val_loss: 3.5779
Epoch 122/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0754 - loss:
3.4991 - val_accuracy: 0.0595 - val_loss: 3.5775
Epoch 123/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0831 - loss:
```

```
3.5670 - val_accuracy: 0.0595 - val_loss: 3.5768
Epoch 124/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0725 - loss:
3.5285 - val_accuracy: 0.0595 - val_loss: 3.5760
Epoch 125/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0727 - loss:
3.5321 - val_accuracy: 0.0595 - val_loss: 3.5738
Epoch 126/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0782 - loss:
3.5050 - val_accuracy: 0.0595 - val_loss: 3.5743
Epoch 127/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0831 - loss:
3.5687 - val_accuracy: 0.0595 - val_loss: 3.5733
Epoch 128/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0966 - loss:
3.6457 - val_accuracy: 0.0595 - val_loss: 3.5720
Epoch 129/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0830 - loss:
3.5141 - val_accuracy: 0.0595 - val_loss: 3.5681
Epoch 130/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0830 - loss:
3.5569 - val_accuracy: 0.0595 - val_loss: 3.5672
Epoch 131/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0690 - loss:
3.5318 - val_accuracy: 0.0595 - val_loss: 3.5607
Epoch 132/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0815 - loss:
3.5073 - val_accuracy: 0.0595 - val_loss: 3.5595
Epoch 133/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0829 - loss:
3.5186 - val_accuracy: 0.0595 - val_loss: 3.5590
Epoch 134/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0856 - loss:
3.5218 - val_accuracy: 0.0595 - val_loss: 3.5589
Epoch 135/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0830 - loss:
3.4863 - val_accuracy: 0.0595 - val_loss: 3.5561
Epoch 136/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0748 - loss:
3.5750 - val_accuracy: 0.0595 - val_loss: 3.5550
Epoch 137/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0774 - loss:
3.4847 - val_accuracy: 0.0595 - val_loss: 3.5546
Epoch 138/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.1018 - loss:
3.4988 - val_accuracy: 0.0595 - val_loss: 3.5539
Epoch 139/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0726 - loss:
3.5619 - val_accuracy: 0.0595 - val_loss: 3.5538
```

```
Epoch 140/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0760 - loss:
3.4777 - val_accuracy: 0.0595 - val_loss: 3.5585
Epoch 141/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0722 - loss:
3.5527 - val_accuracy: 0.0595 - val_loss: 3.5608
Epoch 142/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0753 - loss:
3.5210 - val_accuracy: 0.0595 - val_loss: 3.5615
Epoch 143/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0652 - loss:
3.5526 - val_accuracy: 0.0595 - val_loss: 3.5604
Epoch 144/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0735 - loss:
3.5068 - val_accuracy: 0.0595 - val_loss: 3.5613
Epoch 145/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0738 - loss:
3.5235 - val_accuracy: 0.0595 - val_loss: 3.5603
Epoch 146/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0596 - loss:
3.5413 - val_accuracy: 0.0595 - val_loss: 3.5595
Epoch 147/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0747 - loss:
3.5386 - val_accuracy: 0.0595 - val_loss: 3.5593
Epoch 148/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0748 - loss:
3.4920 - val_accuracy: 0.0595 - val_loss: 3.5602
Epoch 149/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0688 - loss:
3.5556 - val_accuracy: 0.0595 - val_loss: 3.5607
Epoch 150/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0794 - loss:
3.4924 - val_accuracy: 0.0595 - val_loss: 3.5606
Epoch 151/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0690 - loss:
3.5136 - val_accuracy: 0.0595 - val_loss: 3.5610
Epoch 152/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0859 - loss:
3.4971 - val_accuracy: 0.0595 - val_loss: 3.5670
Epoch 153/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0822 - loss:
3.5188 - val_accuracy: 0.0595 - val_loss: 3.5694
Epoch 154/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0775 - loss:
3.5089 - val_accuracy: 0.0595 - val_loss: 3.5671
Epoch 155/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0721 - loss:
3.5064 - val_accuracy: 0.0714 - val_loss: 3.5470
Epoch 156/200
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0756 - loss:
3.5077 - val_accuracy: 0.0952 - val_loss: 3.5431
Epoch 157/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0649 - loss:
3.5195 - val_accuracy: 0.0714 - val_loss: 3.5461
Epoch 158/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0992 - loss:
3.5157 - val_accuracy: 0.0595 - val_loss: 3.5516
Epoch 159/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0824 - loss:
3.5144 - val_accuracy: 0.0595 - val_loss: 3.5545
Epoch 160/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0644 - loss:
3.5044 - val_accuracy: 0.0595 - val_loss: 3.5554
Epoch 161/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0731 - loss:
3.5022 - val_accuracy: 0.0595 - val_loss: 3.5569
Epoch 162/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0545 - loss:
3.5203 - val_accuracy: 0.0595 - val_loss: 3.5579
Epoch 163/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0581 - loss:
3.5244 - val_accuracy: 0.0595 - val_loss: 3.5623
Epoch 164/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0588 - loss:
3.5571 - val_accuracy: 0.0595 - val_loss: 3.5659
Epoch 165/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0600 - loss:
3.6901 - val_accuracy: 0.0595 - val_loss: 3.5670
Epoch 166/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0781 - loss:
3.5128 - val_accuracy: 0.0595 - val_loss: 3.5675
Epoch 167/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0865 - loss:
3.4838 - val_accuracy: 0.0595 - val_loss: 3.5690
Epoch 168/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0968 - loss:
3.4696 - val_accuracy: 0.0595 - val_loss: 3.5705
Epoch 169/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0658 - loss:
3.4850 - val_accuracy: 0.0595 - val_loss: 3.5659
Epoch 170/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0750 - loss:
3.4763 - val_accuracy: 0.0595 - val_loss: 3.5625
Epoch 171/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0828 - loss:
3.5204 - val_accuracy: 0.0595 - val_loss: 3.5595
Epoch 172/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0587 - loss:
3.5534 - val_accuracy: 0.0595 - val_loss: 3.5558
```

```
Epoch 173/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0726 - loss:
3.4917 - val_accuracy: 0.0595 - val_loss: 3.5531
Epoch 174/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0839 - loss:
3.5116 - val_accuracy: 0.0595 - val_loss: 3.5542
Epoch 175/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0790 - loss:
3.4620 - val_accuracy: 0.0595 - val_loss: 3.5553
Epoch 176/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0537 - loss:
3.5353 - val_accuracy: 0.0595 - val_loss: 3.5559
Epoch 177/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0894 - loss:
3.4947 - val_accuracy: 0.0714 - val_loss: 3.5564
Epoch 178/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0685 - loss:
3.5337 - val_accuracy: 0.0714 - val_loss: 3.5572
Epoch 179/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0865 - loss:
3.4904 - val_accuracy: 0.0714 - val_loss: 3.5570
Epoch 180/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0958 - loss:
3.5145 - val_accuracy: 0.0595 - val_loss: 3.5532
Epoch 181/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0629 - loss:
3.5245 - val_accuracy: 0.0714 - val_loss: 3.5496
Epoch 182/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0537 - loss:
3.5557 - val_accuracy: 0.0952 - val_loss: 3.5450
Epoch 183/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0776 - loss:
3.4917 - val_accuracy: 0.0952 - val_loss: 3.5427
Epoch 184/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0764 - loss:
3.5287 - val_accuracy: 0.0952 - val_loss: 3.5448
Epoch 185/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0648 - loss:
3.5342 - val_accuracy: 0.0952 - val_loss: 3.5484
Epoch 186/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0799 - loss:
3.4997 - val_accuracy: 0.0595 - val_loss: 3.5528
Epoch 187/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0956 - loss:
3.4649 - val_accuracy: 0.0595 - val_loss: 3.5544
Epoch 188/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0744 - loss:
3.4897 - val_accuracy: 0.0595 - val_loss: 3.5526
Epoch 189/200
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0653 - loss: 
3.5227 - val_accuracy: 0.0714 - val_loss: 3.5486
Epoch 190/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0774 - loss: 
3.4824 - val_accuracy: 0.0595 - val_loss: 3.5467
Epoch 191/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0961 - loss: 
3.4766 - val_accuracy: 0.0833 - val_loss: 3.5454
Epoch 192/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0799 - loss: 
3.4794 - val_accuracy: 0.0714 - val_loss: 3.5458
Epoch 193/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0557 - loss: 
3.5481 - val_accuracy: 0.0714 - val_loss: 3.5477
Epoch 194/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0841 - loss: 
3.5103 - val_accuracy: 0.0476 - val_loss: 3.5514
Epoch 195/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0785 - loss: 
3.5373 - val_accuracy: 0.0476 - val_loss: 3.5550
Epoch 196/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0887 - loss: 
3.5377 - val_accuracy: 0.0595 - val_loss: 3.5602
Epoch 197/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.0732 - loss: 
3.4995 - val_accuracy: 0.0595 - val_loss: 3.5668
Epoch 198/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.0914 - loss: 
3.4873 - val_accuracy: 0.0595 - val_loss: 3.5674
Epoch 199/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0885 - loss: 
3.4936 - val_accuracy: 0.0595 - val_loss: 3.5583
Epoch 200/200
11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.0833 - loss: 
3.4604 - val_accuracy: 0.0595 - val_loss: 3.5494

# Evaluate the model on the validation set
val_loss, val_accuracy = model.evaluate(X_val, y_val)
print(f'Validation Accuracy: {val_accuracy:.4f}')

3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.0805 - loss: 3.5813

Validation Accuracy: 0.0595

#saving model
model.save('chatbot.h5')

# Load the saved model
loaded_model = load_model('chatbot.h5')
```

```python
# Function to preprocess input sentence
def preprocess_input_sentence(sentence, tokenizer,
max_sequence_length):
    # Tokenize input sentence
    input_sequence = tokenizer.texts_to_sequences([sentence])
    # Pad sequences
    padded_sequence = pad_sequences(input_sequence,
maxlen=max_sequence_length, padding='post')
    return padded_sequence

# Function to generate response
def generate_response(sentence, loaded_model, tokenizer,
max_sequence_length, label_encoder):
    # Preprocess input sentence
    preprocessed_input = preprocess_input_sentence(sentence,
tokenizer, max_sequence_length)
    # Predict probabilities for all classes
    predicted_probabilities = loaded_model.predict(preprocessed_input)
    # Get the index of the class with the highest probability
    predicted_label_index = np.argmax(predicted_probabilities, axis=-
1)
    # Get the predicted tag based on the index
    predicted_tag =
label_encoder.inverse_transform(predicted_label_index)
    return predicted_tag

#Add sentence and get their predicted Tag
# Example of conversation loop
print("Chatbot: Hello! How can I assist you today? (Type 'exit' to
end)")
while True:
    user_input = input("You: ")
    if user_input.lower() == 'exit':
        print("Chatbot: Goodbye! Have a great day!")
        break
    # Get response from the chatbot
    response = generate_response(user_input, loaded_model, tokenizer,
max_sequence_length, label_encoder)
    print("Chatbot:", response[0])
```

```
Chatbot: Hello! How can I assist you today? (Type 'exit' to end)

You:  hello

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Chatbot: course

You:  how are you

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Chatbot: course
```

```
You:  exit

Chatbot: Goodbye! Have a great day!

#getting all patterns of the predicted Tag
# Function to preprocess input sentence
def preprocess_input_sentence(sentence, tokenizer,
max_sequence_length):
    # Tokenize input sentence
    input_sequence = tokenizer.texts_to_sequences([sentence])
    # Pad sequences
    padded_sequence = pad_sequences(input_sequence,
maxlen=max_sequence_length, padding='post')
    return padded_sequence
# Function to get response
def get_response(sentence, loaded_model, tokenizer,
max_sequence_length, intents, df):
    # Preprocess input sentence
    preprocessed_input = preprocess_input_sentence(sentence,
tokenizer, max_sequence_length)
    # Predict label
    predicted_label =
loaded_model.predict(preprocessed_input).argmax(axis=-1)
    # Convert label to tag
    predicted_tag = label_encoder.inverse_transform(predicted_label)
    # Get the response based on the predicted tag
    if predicted_tag[0] in df['Tag'].values:
        response_row = df[df['Tag'] == predicted_tag[0]].iloc[0]
        response = response_row['Tag'] + ": " + df['Pattern']
    else:
        response = intents.get(predicted_tag[0], "I'm sorry, I didn't
understand that.")
    return response
# Example usage
sentence = "Hi"
response = get_response(sentence, loaded_model, tokenizer,
max_sequence_length, intents, df)
print("Chatbot:", response)

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Chatbot: 0                        course: Hi
1            course: How are you?
2        course: Is anyone there?
3                   course: Hello
4                course: Good day
                  ...
414              course: two week
415             course: Five days
416              course: two days
417             course: tommorrow
```

```
418    course: day after tomorrow
Name: Pattern, Length: 419, dtype: object
```

```
#Now i can chat with my chatbot
# Function to preprocess input sentence
def preprocess_input_sentence(sentence, tokenizer,
max_sequence_length):
    # Tokenize input sentence
    input_sequence = tokenizer.texts_to_sequences([sentence])
    # Pad sequences
    padded_sequence = pad_sequences(input_sequence,
maxlen=max_sequence_length, padding='post')
    return padded_sequence
# Function to get response
def get_response(sentence, loaded_model, tokenizer,
max_sequence_length, intents):
    # Preprocess input sentence
    preprocessed_input = preprocess_input_sentence(sentence,
tokenizer, max_sequence_length)
    # Predict label
    predicted_label =
loaded_model.predict(preprocessed_input).argmax(axis=-1)
    # Convert label to tag
    predicted_tag = label_encoder.inverse_transform(predicted_label)

    # Iterate through the intents to find the matching tag
    for intent in intents["intents"]:
        if intent["tag"] == predicted_tag[0]:
            # Select a random response from the list of responses
            response = random.choice(intent["responses"])
            return response

    # If no matching intent is found, return a default message
    return "I'm sorry, I didn't understand that."
print("Chatbot: Hello! How can I assist you today? (Type 'exit' to
end)")
while True:
    # Get user input
    sentence = input("You: ")
    # Check if the user wants to exit
    if sentence.lower() == 'exit':
        print("Chatbot: Goodbye! Have a great day!")
        break
    # Get response from chatbot
    response = get_response(sentence, loaded_model, tokenizer,
max_sequence_length, intents)
    print("Chatbot:", response)
```

```
Chatbot: Hello! How can I assist you today? (Type 'exit' to end)
```

```
You:  hello

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Chatbot: Our rganization offers Information Technology, computer
Engineering, Mechanical engineering,Chemical engineering, Civil
engineering and extc Engineering.

You:  i need leave

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Chatbot: Our rganization offers Information Technology, computer
Engineering, Mechanical engineering,Chemical engineering, Civil
engineering and extc Engineering.

You:  exit

Chatbot: Goodbye! Have a great day!
```

```python
def preprocess_input_sentence(sentence, tokenizer,
max_sequence_length):
    # Tokenize input sentence
    input_sequence = tokenizer.texts_to_sequences([sentence])
    # Pad sequences
    padded_sequence = pad_sequences(input_sequence,
maxlen=max_sequence_length, padding='post')
    return padded_sequence

def get_response(sentence, loaded_model, tokenizer,
max_sequence_length, intents):
    # Preprocess input sentence
    preprocessed_input = preprocess_input_sentence(sentence,
tokenizer, max_sequence_length)
    # Predict probabilities
    predicted_probabilities = loaded_model.predict(preprocessed_input)
[0]
    # Get index with highest probability
    predicted_label_index = predicted_probabilities.argmax()
    # Convert index to tag
    predicted_tag =
label_encoder.inverse_transform([predicted_label_index])[0]

    # Iterate through the intents to find the matching tag
    for intent in intents["intents"]:
        if intent["tag"] == predicted_tag:
            # Select a random response from the list of responses
            response = random.choice(intent["responses"])
            return response

    # If no matching intent is found, return a default message
    return "I'm sorry, I didn't understand that."

print("Chatbot: Hello! How can I assist you today? (Type 'exit' to
end)")
while True:
    # Get user input
    sentence = input("You: ")
    # Check if the user wants to exit
    if sentence.lower() == 'exit':
        print("Chatbot: Goodbye! Have a great day!")
        break
    # Get response from chatbot
    response = get_response(sentence, loaded_model, tokenizer,
max_sequence_length, intents)
    print("Chatbot:", response)
```