Problem Statement/ Bharat ChatBot

Source Code URL(Github/Openforge/etc.) :

Video Url :

Project Report (Maximum 4 PDFs - each upto 2 MB) : attached file projectreport.pdf

Idea/Concept (max 1000 characters)*

With the advent of GPUs, deep learning techniques are becoming popular for development of artificial intelligent powered intelligent chatbot which can interact with users, understand their queries, map their preferences and recommend an appropriate line of action with minimal or no human intervention. We designed deep learning based "Bharat ChatBot" App which is capable of interacting users for grievance redress task. It is designed to provide best response/relevant information to the queries made by the public on website portal. It uses NLP (Natural Language Processing) text processing for user intent detection. The app consists of python modules 1) Module 1: aichat_server.py - This module is responsible for creation of trained deep learning based model for given .json dataset. It uses python packages : nltk, tensorflow, tflearn, numpy, json, random, pickle; and 2) Module 2: aichat_server.py- This module uses python packages : flask, nltk, tensorflow, tflearn, numpy, json, random, pickle. The module uses Flask, a lightweight WSGI web application framework that runs FlaskServer hosted at local port no. 5000 (default). The application reads dataset files intents.json, data.pickle and model.tflearn to retrieve the trained model. To assess the working, GUI is provided on home.html. Whenever user visits the default portal http://127.0.0.1:5000/, home.html page will open which provides the chat interface; where user can type query in his textarea (which is focused all the time). As soon as user press enter key, the reply to the query will be provided by BharatChatBot in his texfield (which is non-editable by user). About.html gives more information about grievance redress process.

Many researchers carried out the experiments on standard datasets like "Cornell Movie Subtitle Corpus", twitter data, reddit dataset and relay chat engine data corpus; however, from grievance redress point of view these standard dataset will not be useful for training our model. Hence domain specific .json dataset file need to be created as per need of the public grievance redress services. The dataset files available on https://event.data.gov.in/ is limited and do not contain data in patterns and best possible responses format.

Project Description (max 2000 characters)

With the advent of GPUs, deep learning techniques are becoming popular for development of artificial intelligent powered intelligent chatbot which can interact with users, understand their queries, map their preferences and recommend an appropriate line of action with minimal or no human intervention.

We designed deep learning based “Bharat ChatBot” App which is capable of interacting users for grievance redress task. It is designed to provide best response/relevant information to the queries made by the public on website portal. It uses NLP (Natural Language Processing) text processing for user intent detection. Every intent triggers a different response. E.g. “Hello” → “Hi!”, “What’s your name?” → “I am Bharat”. At the same time it is capable for handling situations when same question is asked in different ways. E.g. “I am facing salary problem” and “I have not received salary yet”. It will direct user to provide additional information whenever required. The dataset format used here is .json that stores data in pattern and response pairs with tags defined for each pair as shown below:

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi", "Hello", "Good day", "Whats up"],
         "responses": ["Hello Sir!", "Good to see you Sir!", "Hi, how can I help you
Sir?", "Welcome Sir","How can i help you?"],
         "context_set": ""
        },
…..
]}
```

Many researchers carried out the experiments on standard datasets like ”Cornell Movie Subtitle Corpus”, twitter data, reddit dataset and relay chat engine data corpus; however, from grievance redress point of view these standard dataset will not be useful for training our model. Hence domain specific .json dataset file need to be created as per need of the public grievance redress services. The dataset files available on https://event.data.gov.in/ is limited and do not contain data in patterns and best possible responses format.

Algorithm Steps:

The app consists of following python modules.

Module 1: aichat_server.py

This module is responsible for creation of trained deep learning based model for given .json dataset. It uses python packages : nltk, tensorflow, tflearn, numpy, json, random, pickle.

First it reads intents.json file containing dataset containing patterns and responses with tags. It tokenizes each pattern into sequence of words and uses tags (total 11 unique tags present currently related to employee’s grievance related topics as: greeting, goodbye, problem, name, contact, pension, salary, medical, training, facilities, hours) as corresponding labels for given pattern in intent. The read text is converted to lowercase. A word stemmer based on the nltk Lancaster stemming algorithm is used to convert all unique words detected in given patterns. For e.g. if word is 'saying' will be converted to basic form i.e. word ‘say’. Both Words (total 120 unique words) and label vector are sorted alphabetically. Both the words and labels in our training data are then represented as one-hot vectors. Each word in the input is then represented as a vector of n=120 values (where n=number of unique words in .json file), with all zeros except the subscript of the word from our corpus. For example, if pension is the 10th word in the corpus, the one hot vector for pension is represented as a vector of 120 values, all zeros except 10th index in the vector as 1. We save the values of intermediate data variables in data.pickle file which will be required at runtime later.

Deep learning Module uses tensorflow package and TFlearn - a modular and transparent deep learning library built on top of Tensorflow.. First it clears the default graph stack and resets the global default graph.

Following Table are details of tensorflow model:

Table 1: model_summary()

	Variables: name (type shape) [size]
net = tflearn.input_data(shape=[None, len(training[0])],name='Input_layer')	Hidden_layer_1/W:0 (float32_ref 120x8) [960, bytes: 3840]
net = tflearn.fully_connected(net, 8,activation='relu', name='Hidden_layer_1')	Hidden_layer_1/b:0 (float32_ref 8) [8, bytes: 32]
net = tflearn.fully_connected(net, 8,activation='relu', name='Hidden_layer_2')	Hidden_layer_2/W:0 (float32_ref 8x8) [64, bytes: 256]
net = tflearn.fully_connected(net, len(output[0]), activation="softmax",name='Output_layer')	Hidden_layer_2/b:0 (float32_ref 8) [8, bytes: 32]
net = tflearn.regression(net, optimizer='sgd',learning_rate=2, loss='mean_square')	Output_layer/W:0 (float32_ref 8x11) [88, bytes: 352]
model = tflearn.DNN(net)	Output_layer/b:0 (float32_ref 11) [11, bytes: 44]
	Total size of variables: 1139 Total bytes of variables: 4556

Train the model using training dataset. The number of epochs set=1000, batch_size=8. The model is evaluated against unseen test dataset. The training and testing accuracy achieved is greater than 95% and used for final deployment. Save the model in file model.tflearn for further prediction.

Module 2: aichat_server.py

This module uses python packages : flask, nltk, tensorflow, tflearn, numpy, json, random, pickle. The module uses Flask, a lightweight WSGI web application framework that runs FlaskServer hosted at local port no. 5000 (default). The application reads dataset files intents.json, data.pickle and model.tflearn to retrieve the trained model. First create virtual environment and do: pip install flask, you may need tensorflow 1.14 version. This will create api.py file. In api.py contains following routes are defined as shown in Table 2:

Table 2: app.route enrties in api.py file

@app.route('/', methods=['GET'])
def home():
  return render_template("home.html")
	Whenever user visits the default portal http://127.0.0.1:5000/, home.html page will open

@app.route('/message', methods=['POST'])	Whenever user types his query and enters on homepage, the route

def message():
  data = request.form
  messageret= chat(data['message'])
  print(data['message'])
  return render_template("home.html", messageret=messageret)
	calls chat method to get the response from trained model and sends it to user on home.html.

@app.route("/about")
def about():
  return render_template("about.html")
	Whenever user clicks on about tab on the default portal http://127.0.0.1:5000/, about.html page will open

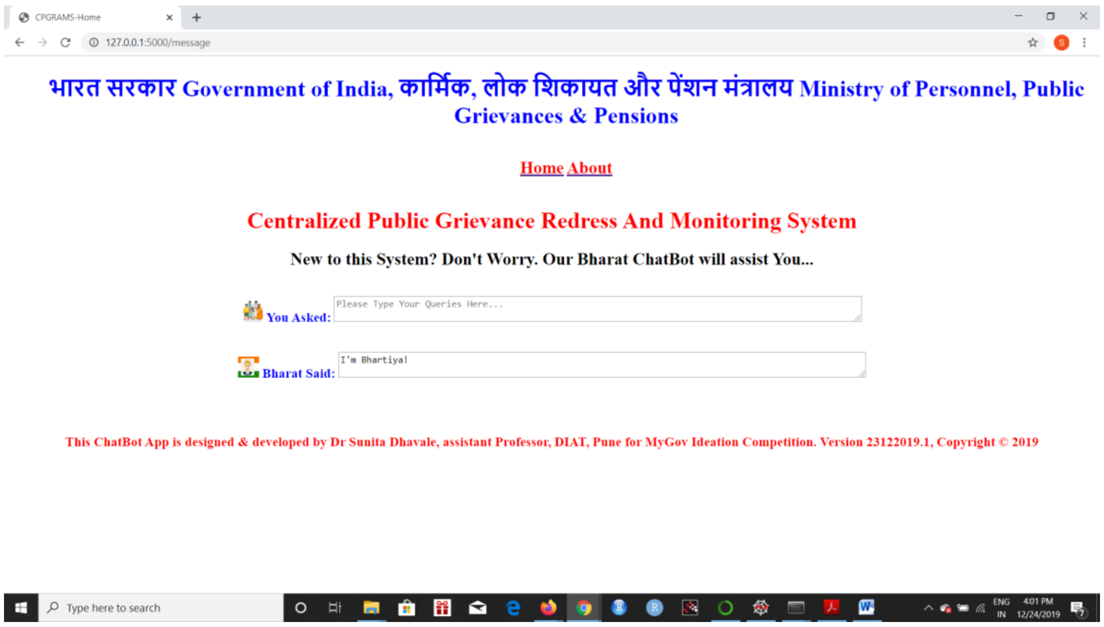
@app.errorhandler(404) def page_not_found(e): return "<h1>404</h1><p>The resource could not be found.</p>", 404	Just to display error code

The deep learning model based Bharat ChatBot App learns abstract feature of this .json dataset and later retrieves appropriate best marching response to the seen/unseen queries. The model returns a reply from the .json dataset by computing the most likely response to the current input utterance based on a scoring function. Hence the performance of system lies on the amount and accuracy of dataset on which the model is trained beside number of hidden layers/number of hyper parameters used.

Bharat ChatBot/conversational agent can 1) record grievance 2) route to appropriate department 3) answer user's question, 4) check status 5) give the user relevant information, 6) ask follow-up questions, and 7) continue the conversation in a realistic way.

This work is submitted as part of online Hackathon for inviting innovative solutions for the Citizen Grievance Redressal Mechanism in order to help the Citizens to resolve their common queries related to filing a Grievance in the CPGRAMS portal (https://pgportal.gov.in) and expedite smooth submission of grievances.

The model is deployed using a Flask server, and home.html page on Centralized Public Grievance Redress And Monitoring System Portal is designed to interact with it. Following Figures 1-4 show the snapshots of the same.



भारत सरकार Government of India, कार्मिक, लोक शिकायत और पेंशन मंत्रालय Ministry of Personnel, Public Grievances & Pensions

Home About

Centralized Public Grievance Redress And Monitoring System

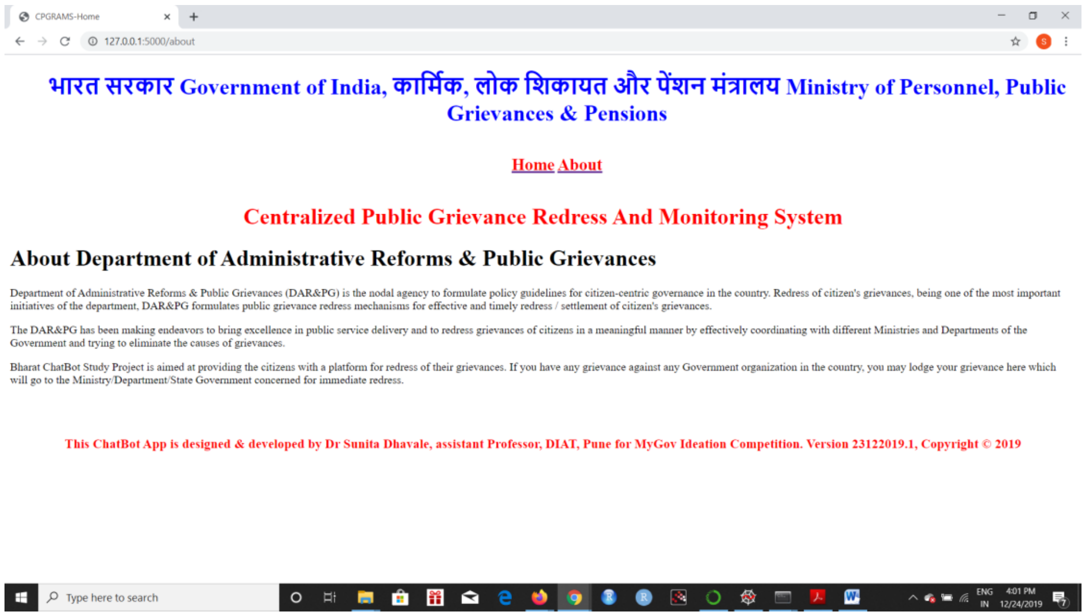
New to this System? Don't Worry. Our Bharat ChatBot will assist You...

You Asked: Please Type Your Queries Here...

Bharat Said: I'm Bhartiya!

This ChatBot App is designed & developed by Dr Sunita Dhavale, assistant Professor, DIAT, Pune for MyGov Ideation Competition. Version 23122019.1, Copyright © 2019

Figure 1: Home Page containing Bharat ChatBot Interface running at local FlaskServer hosted at local port no. 5000



भारत सरकार Government of India, कार्मिक, लोक शिकायत और पेंशन मंत्रालय Ministry of Personnel, Public Grievances & Pensions

Home About

Centralized Public Grievance Redress And Monitoring System

About Department of Administrative Reforms & Public Grievances

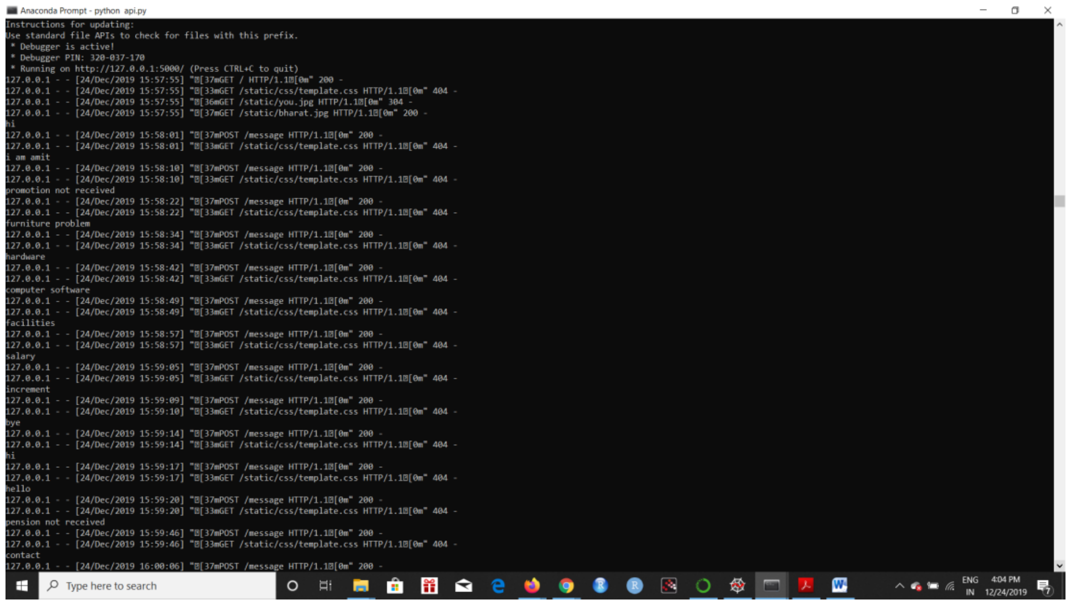
Department of Administrative Reforms & Public Grievances (DAR&PG) is the nodal agency to formulate policy guidelines for citizen-centric governance in the country. Redress of citizen's grievances, being one of the most important initiatives of the department, DAR&PG formulates public grievance redress mechanisms for effective and timely redress / settlement of citizen's grievances.

The DAR&PG has been making endeavors to bring excellence in public service delivery and to redress grievances of citizens in a meaningful manner by effectively coordinating with different Ministries and Departments of the Government and trying to eliminate the causes of grievances.

Bharat ChatBot Study Project is aimed at providing the citizens with a platform for redress of their grievances. If you have any grievance against any Government organization in the country, you may lodge your grievance here which will go to the Ministry/Department/State Government concerned for immediate redress.

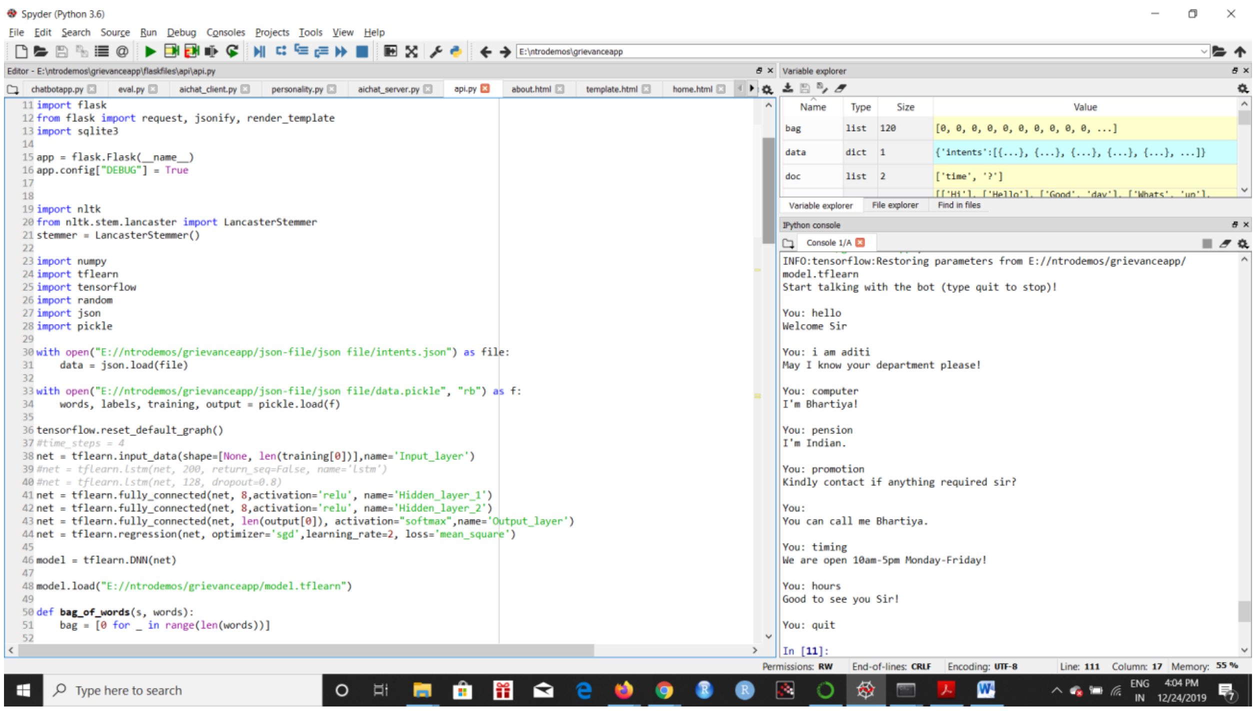
This ChatBot App is designed & developed by Dr Sunita Dhavale, assistant Professor, DIAT, Pune for MyGov Ideation Competition. Version 23122019.1, Copyright © 2019

Figure 2: About html Page



```
Instructions for updating:
Use standard file APIs to check for files with this prefix.
 * Debugger is active!
 * Debugger PIN: 320-037-170
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [24/Dec/2019 15:57:55] "▯[37mGET / HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:57:55] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
127.0.0.1 - - [24/Dec/2019 15:57:55] "▯[36mGET /static/you.jpg HTTP/1.1▯[0m" 304 -
127.0.0.1 - - [24/Dec/2019 15:57:55] "▯[37mGET /static/bharat.jpg HTTP/1.1▯[0m" 200 -
hi
127.0.0.1 - - [24/Dec/2019 15:58:01] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:01] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
i am amit
127.0.0.1 - - [24/Dec/2019 15:58:10] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:10] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
promotion not received
127.0.0.1 - - [24/Dec/2019 15:58:22] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:22] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
furniture problem
127.0.0.1 - - [24/Dec/2019 15:58:34] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:34] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
hardware
127.0.0.1 - - [24/Dec/2019 15:58:42] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:42] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
computer software
127.0.0.1 - - [24/Dec/2019 15:58:49] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:49] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
facilities
127.0.0.1 - - [24/Dec/2019 15:58:57] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:58:57] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
salary
127.0.0.1 - - [24/Dec/2019 15:59:05] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:59:05] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
increment
127.0.0.1 - - [24/Dec/2019 15:59:09] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:59:10] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
bye
127.0.0.1 - - [24/Dec/2019 15:59:14] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:59:14] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
hi
127.0.0.1 - - [24/Dec/2019 15:59:17] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:59:17] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
hello
127.0.0.1 - - [24/Dec/2019 15:59:20] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:59:20] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
pension not received
127.0.0.1 - - [24/Dec/2019 15:59:46] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
127.0.0.1 - - [24/Dec/2019 15:59:46] "▯[33mGET /static/css/template.css HTTP/1.1▯[0m" 404 -
contact
127.0.0.1 - - [24/Dec/2019 16:00:06] "▯[37mPOST /message HTTP/1.1▯[0m" 200 -
```

Figure 3: FlaskServer running at background. “At anaconda prompt type: python api.py”



```python
import flask
from flask import request, jsonify, render_template
import sqlite3

app = flask.Flask(__name__)
app.config["DEBUG"] = True


import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import numpy
import tflearn
import tensorflow
import random
import json
import pickle

with open("E://ntrodemos/grievanceapp/json-file/json file/intents.json") as file:
    data = json.load(file)

with open("E://ntrodemos/grievanceapp/json-file/json file/data.pickle", "rb") as f:
    words, labels, training, output = pickle.load(f)

tensorflow.reset_default_graph()
#time_steps = 4
net = tflearn.input_data(shape=[None, len(training[0])],name='Input_layer')
#net = tflearn.lstm(net, 200, return_seq=False, name='lstm')
#net = tflearn.lstm(net, 128, dropout=0.8)
net = tflearn.fully_connected(net, 8,activation='relu', name='Hidden_layer_1')
net = tflearn.fully_connected(net, 8,activation='relu', name='Hidden_layer_2')
net = tflearn.fully_connected(net, len(output[0]), activation="softmax",name='Output_layer')
net = tflearn.regression(net, optimizer='sgd',learning_rate=2, loss='mean_square')

model = tflearn.DNN(net)

model.load("E://ntrodemos/grievanceapp/model.tflearn")

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

```

Name	Type	Size	Value
bag	list	120	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...]
data	dict	1	{'intents':[{...}, {...}, {...}, {...}, {...}, ...]}
doc	list	2	['time', '?']

```
INFO:tensorflow:Restoring parameters from E://ntrodemos/grievanceapp/
model.tflearn
Start talking with the bot (type quit to stop)!

You: hello
Welcome Sir

You: i am aditi
May I know your department please!

You: computer
I'm Bhartiya!

You: pension
I'm Indian.

You: promotion
Kindly contact if anything required sir?

You:
You can call me Bhartiya.

You: timing
We are open 10am-5pm Monday-Friday!

You: hours
Good to see you Sir!

You: quit

In [11]:
```

Figure 4: Spyder IDE used for development

References:

[1] I. Goodfellow, Deep learning, Cambrigde, MA: The MIT Press, 2016.

[2] D. Britz, "Deep Learning for Chatbots, Part 1 –Introduction," April 2016. [Online]. Available: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.

[3] H. N. Io and C. B. Lee, "Chatbots and conversational agents: A bibliometric analysis," 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 2017, pp. 215-219. doi: 10.1109/IEEM.2017.8289883

Acknowledgements

We would like to thank NVIDIA Corporation for granting us TITAN V GPU for carrying out deep learning based research work.