# PROJECT REPORT ON

**CHATBOT USING TENSORFLOW AI**

**DEVELOPED BY**

**SWASTICA N S**

**BCA (DATA SCIENCE)**

**INTRODUCTION**

In the era of rapid digital transformation, chatbots have emerged as essential tools for automating communication and enhancing user interaction across various platforms. This project focuses on the development of an intelligent chatbot using TensorFlow, an open-source machine learning framework developed by Google. Leveraging the capabilities of natural language processing (NLP) and deep learning, the chatbot is designed to understand user input and generate contextually relevant responses.

The goal of this project is to build a conversational agent that can simulate human-like dialogue and provide assistance or information in real time. TensorFlow’s extensive ecosystem allows for the integration of neural network models, enabling the chatbot to learn from data and improve its responses over time. The implementation demonstrates how artificial intelligence can be applied to solve real-world problems in customer service, education, healthcare, and more.

This report outlines the design, development, and evaluation of the chatbot, detailing the methodologies used, challenges encountered, and potential future enhancements. By utilizing TensorFlow for both training and inference, the project aims to highlight the power of deep learning in building scalable and intelligent chatbot solutions.

**OBJECTIVE**

The primary objective of this project is to design, develop, and implement an intelligent conversational chatbot using TensorFlow, an open-source deep learning framework developed by Google. The chatbot aims to simulate human-like conversation by understanding natural language inputs and generating relevant, context-aware responses in real time.

This project explores the application of Natural Language Processing (NLP) and deep learning techniques to build a system that can:

1. **Interpret user queries** using advanced text preprocessing, tokenization, and intent recognition.
2. **Learn from data** using neural network architectures, particularly sequence models like RNNs, LSTMs, or Transformer-based models.
3. **Generate coherent responses** either by mapping intents to predefined replies or by generating responses dynamically using sequence-to-sequence models.
4. **Deploy a functional chatbot interface**, such as a command-line interface, web interface, or messaging platform integration, to demonstrate the model’s capabilities in a user-friendly manner.

Ultimately, the chatbot developed in this project will serve as a prototype for virtual assistants that can be integrated into customer service, educational platforms, healthcare, and various other domains requiring interactive AI agents.

**PROGRAM**

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Dense, Dropout

import json

import random

import nltk

from nltk.stem import WordNetLemmatizer

import os

import pickle

try:

nltk.download('punkt')

nltk.download('wordnet')

except:

print("NLTK resources already downloaded or couldn't be downloaded")

INTENTS\_FILE = "intents.json"

MODEL\_FILE = "chatbot\_model.h5"

WORDS\_FILE = "words.pkl"

CLASSES\_FILE = "classes.pkl"

if not os.path.exists(INTENTS\_FILE):

sample\_intents = {

"intents": [

{

"tag": "greeting",

"patterns": ["Hi", "Hello", "Hey", "How are you", "What's up"],

"responses": ["Hello!", "Hey there!", "Hi! How can I help you today?"]

},

{

"tag": "goodbye",

"patterns": ["Bye", "See you later", "Goodbye", "I'm leaving"],

"responses": ["Goodbye!", "See you later!", "Have a nice day!"]

},

{

"tag": "thanks",

"patterns": ["Thanks", "Thank you", "That's helpful"],

"responses": ["You're welcome!", "Any time!", "My pleasure!"]

},

{

"tag": "about",

"patterns": ["Who are you?", "What are you?", "Tell me about yourself"],

"responses": ["I'm Zohen,your chatbot assistant!", "I'm Zohen, your friendly AI assistant."]

},

{

"tag": "help",

"patterns": ["Help", "I need help", "Can you help me?"],

"responses": ["I'll try my best to assist you!", "What do you need help with?"]

},

{

"tag": "data\_science",

"patterns": ["What is data science?", "Tell me about data science", "Define data science"],

"responses": ["Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge from data."]

},

{

"tag": "machine\_learning",

"patterns": ["What is machine learning?", "Explain machine learning", "ML definition"],

"responses": ["Machine learning is a subset of AI that allows systems to learn and improve from experience without being explicitly programmed."]

},

{

"tag": "tensorflow",

"patterns": ["What is TensorFlow?", "Tell me about TensorFlow", "TF information"],

"responses": ["TensorFlow is an open-source machine learning framework developed by Google that's used for building and training neural networks."]

},

{

"tag": "neural\_networks",

"patterns": ["What are neural networks?", "Explain neural networks", "NN definition"],

"responses": ["Neural networks are computing systems inspired by biological neural networks, consisting of layers of interconnected nodes that process information."]

},

{

"tag": "joke",

"patterns": ["Tell me a joke", "Say something funny", "Make me laugh"],

"responses": ["Why don't scientists trust atoms? Because they make up everything!",

"What did the data scientist say when asked for a ride? 'Sorry, I only fit models!'",

"Why did the statistician drown? He only knew how to work with sample data!"]

}

]

}

with open(INTENTS\_FILE, 'w') as file:

json.dump(sample\_intents, file, indent=4)

print(f"Created sample intents file at {INTENTS\_FILE}")

lemmatizer = WordNetLemmatizer()

def prepare\_training\_data():

with open(INTENTS\_FILE) as file:

intents = json.load(file)

words = []

classes = []

documents = []

ignore\_chars = ['?', '!', '.', ',']

for intent in intents['intents']:

for pattern in intent['patterns']:

word\_list = nltk.word\_tokenize(pattern)

words.extend(word\_list)

documents.append((word\_list, intent['tag']))

if intent['tag'] not in classes:

classes.append(intent['tag'])

words = [lemmatizer.lemmatize(word.lower()) for word in words if word not in ignore\_chars]

words = sorted(list(set(words)))

classes = sorted(list(set(classes)))

with open(WORDS\_FILE, 'wb') as file:

pickle.dump(words, file)

with open(CLASSES\_FILE, 'wb') as file:

pickle.dump(classes, file)

return words, classes, documents

def create\_training\_data(words, classes, documents):

training = []

output\_empty = [0] \* len(classes)

for doc in documents:

bag = []

word\_patterns = doc[0]

word\_patterns = [lemmatizer.lemmatize(word.lower()) for word in word\_patterns]

for word in words:

bag.append(1) if word in word\_patterns else bag.append(0)

output\_row = list(output\_empty)

output\_row[classes.index(doc[1])] = 1

training.append([bag, output\_row])

random.shuffle(training)

training = np.array(training, dtype=object)

train\_x = list(training[:, 0])

train\_y = list(training[:, 1])

return np.array(train\_x), np.array(train\_y)

def build\_and\_train\_model(train\_x, train\_y):

model = Sequential()

model.add(Dense(128, input\_shape=(len(train\_x[0]),), activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(len(train\_y[0]), activation='softmax'))

model.compile(loss='categorical\_crossentropy',

optimizer=tf.keras.optimizers.Adam(learning\_rate=0.01),

metrics=['accuracy'])

print("Training model...")

model.save(MODEL\_FILE)

print(f"Model saved to {MODEL\_FILE}")

return model, history

def clean\_up\_sentence(sentence):

sentence\_words = nltk.word\_tokenize(sentence)

sentence\_words = [lemmatizer.lemmatize(word.lower()) for word in sentence\_words]

return sentence\_words

def create\_bow(sentence, words):

sentence\_words = clean\_up\_sentence(sentence)

bag = [0] \* len(words)

for s in sentence\_words:

for i, word in enumerate(words):

if word == s:

bag[i] = 1

return np.array(bag)

def predict\_class(sentence, model, words, classes):

bow = create\_bow(sentence, words)

res = model.predict(np.array([bow]))[0]

ERROR\_THRESHOLD = 0.25

results = [[i, r] for i, r in enumerate(res) if r > ERROR\_THRESHOLD]

results.sort(key=lambda x: x[1], reverse=True)

return\_list = []

for r in results:

return\_list.append({

"intent": classes[r[0]],

"probability": str(r[1])

})

return return\_list

def get\_response(intents\_list, intents\_json):

if not intents\_list:

return "I'm not sure I understand. Could you rephrase that?"

tag = intents\_list[0]['intent']

list\_of\_intents = intents\_json['intents']

for i in list\_of\_intents:

if i['tag'] == tag:

result = random.choice(i['responses'])

break

else:

result = "I'm not sure how to respond to that."

return result

def run\_chatbot():

if not os.path.exists(MODEL\_FILE) or not os.path.exists(WORDS\_FILE) or not os.path.exists(CLASSES\_FILE):

print("Training new model...")

words, classes, documents = prepare\_training\_data()

train\_x, train\_y = create\_training\_data(words, classes, documents)

model, history = build\_and\_train\_model(train\_x, train\_y)

else:

print("Loading existing model...")

model = load\_model(MODEL\_FILE)

with open(WORDS\_FILE, 'rb') as file:

words = pickle.load(file)

with open(CLASSES\_FILE, 'rb') as file:

classes = pickle.load(file)

with open(INTENTS\_FILE) as file:

intents = json.load(file)

print("\n----- Simple TensorFlow Chatbot -----")

print("Type 'quit' to exit")

while True:

message = input("\nYou: ")

if message.lower() in ['quit', 'exit', 'bye']:

print("Chatbot: Goodbye!")

break

intents\_list = predict\_class(message, model, words, classes)

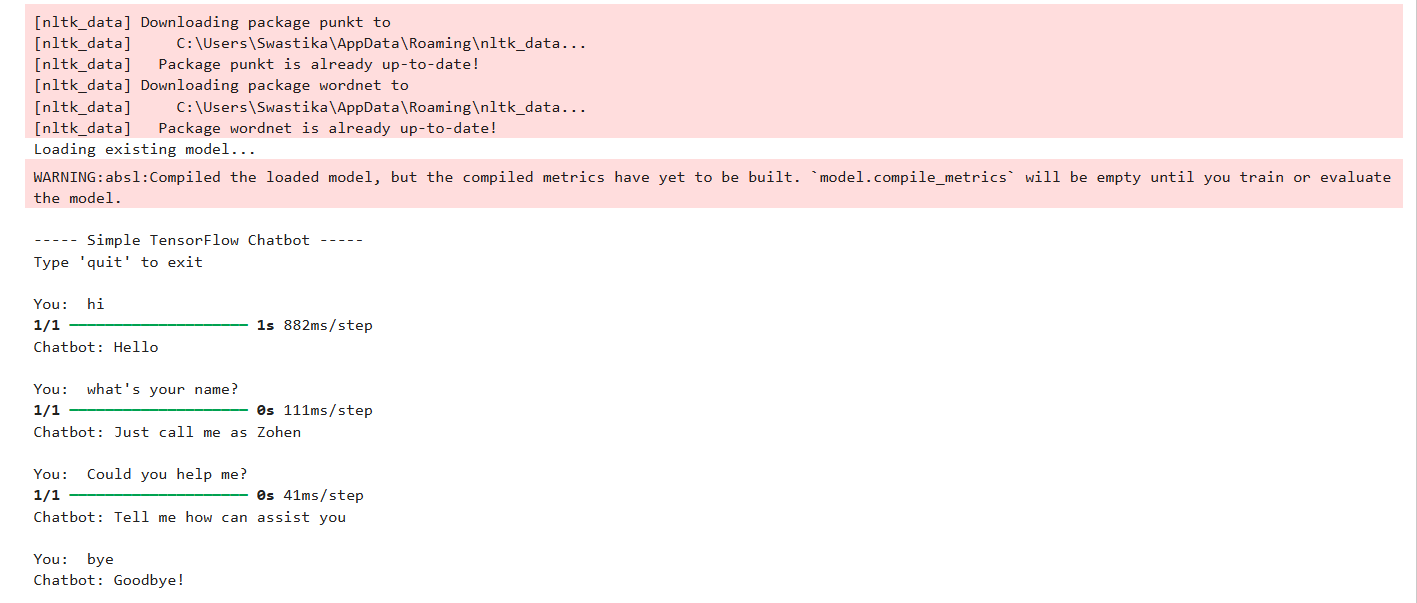
response = get\_response(intents\_list, intents)

print(f"Chatbot: {response}")

if \_\_name\_\_ == "\_\_main\_\_":

run\_chatbot()

**OUTPUT**

****

**REFERENCE**

* W3schools - <https://www.w3schools.com/ai/>
* GeeksforGeeks - <https://www.w3schools.com/ai/>
* Deeplearning.ai - <https://www.deeplearning.ai/resources/natural-language-processing/>
* Tutorialspoint - <https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_natural_language_processing.htm>

**SUMMARY**

This project presents the development of an AI-powered chatbot using TensorFlow, an open-source machine learning framework. The chatbot is designed to simulate human-like conversations, provide relevant responses, and assist users in real time.

The system leverages natural language processing (NLP) techniques to understand user queries and respond appropriately. Key components include intent classification using neural networks, tokenization, lemmatization, and training the model on a predefined dataset of intents and responses.

TensorFlow is used to build and train the deep learning model, allowing the chatbot to learn patterns and improve its accuracy over time. The chatbot is implemented using Python and integrates tools such as NLTK for NLP preprocessing.

The end result is a lightweight, responsive chatbot capable of basic conversation, FAQs, and task assistance. This project showcases how AI and machine learning can be used to enhance user interaction and automate communication processes.