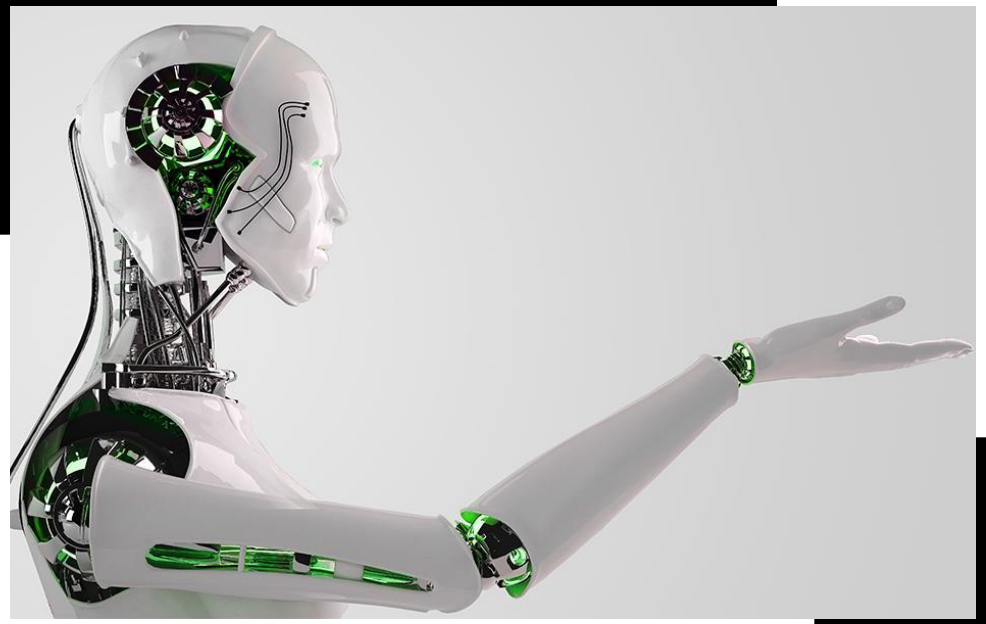
Building nlp-based ChatBot using

Deeplearning.

Our mission is to ensure that artificial general intelligence benefits all of humanity.

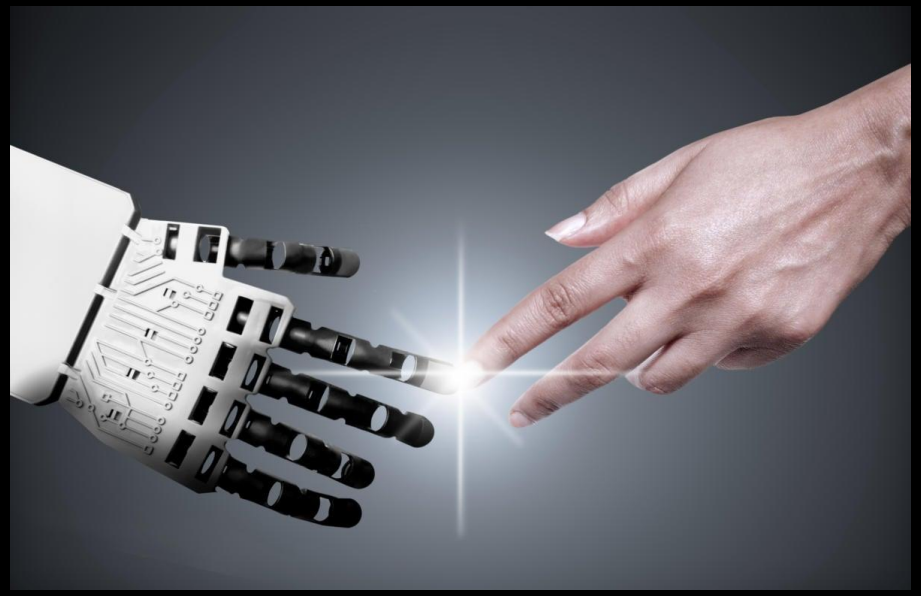


About Chatbot

A Chatbot is an automated system designed to initiate a conversation with human users or other chatbots.
Our goal is to help you build a smart chatbot.

Types of ChatBot

Before building a chatbot we should have knowledge about type of the chatbot that we are building.In this project we are using retrieval -based ChatBots.



1. Rule-Based Chatbots

rely on predefined rules and patterns to generate responses, making them suitable for simple use cases but limited in their ability to understand.

2. Retrieval-Based Chatbots

utilize predefined responses from a dataset or knowledge base, leveraging techniques such as keyword matching, TF-IDF, or word embeddings to analyze user inputs

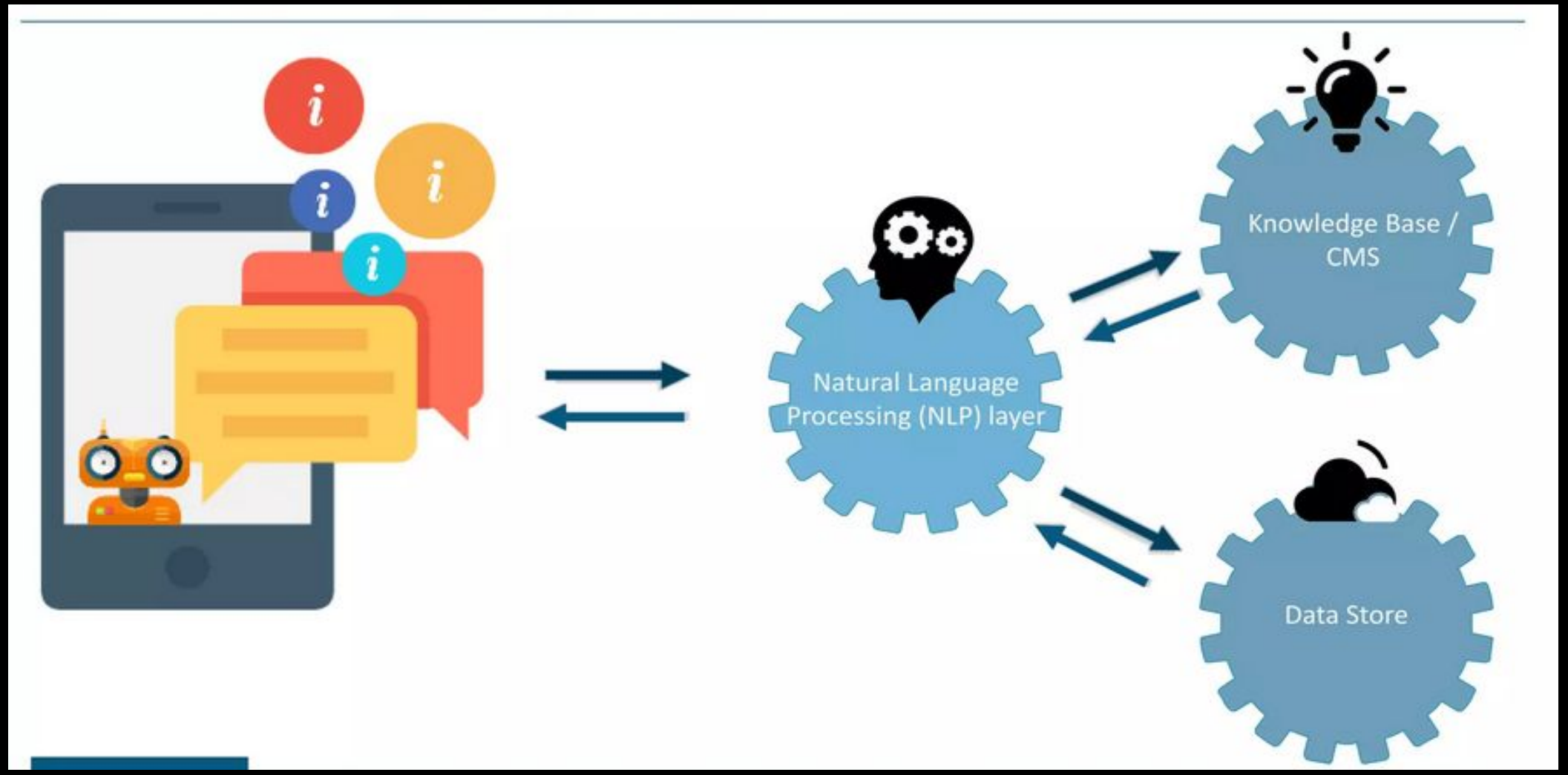
3. Generative Chatbots

utilize techniques like sequence-to-sequence models, RNNs, or transformer models to generate original and contextually relevant responses, but their training requires significant data and computational resources.

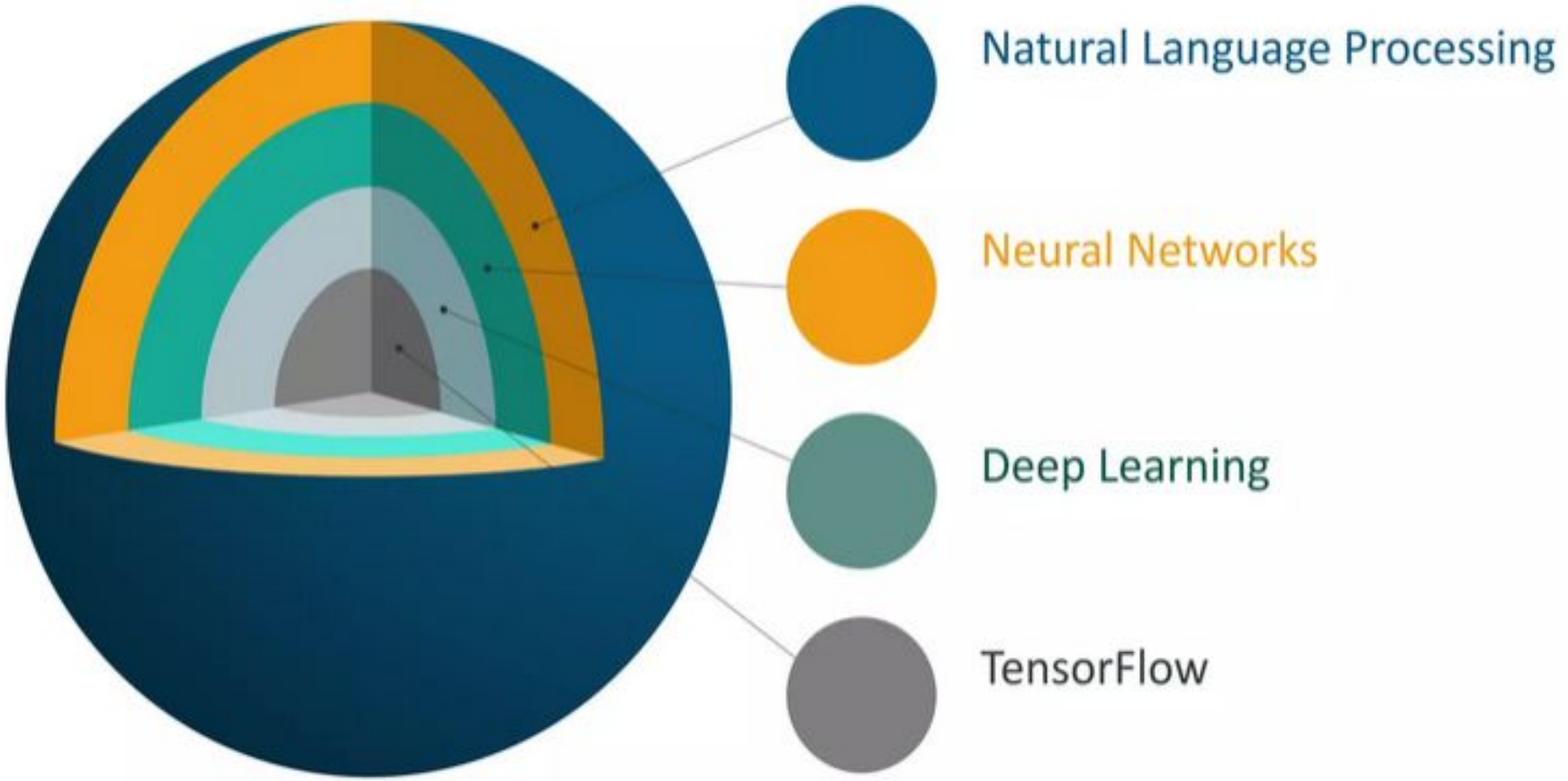
4. Hybrid Chatbots

leverage a combination of rule-based or retrieval-based approaches for specific scenarios while integrating generative models to handle user inputs, achieving a balance between response flexibility and accuracy..

HOW CHATBOT WORKS:

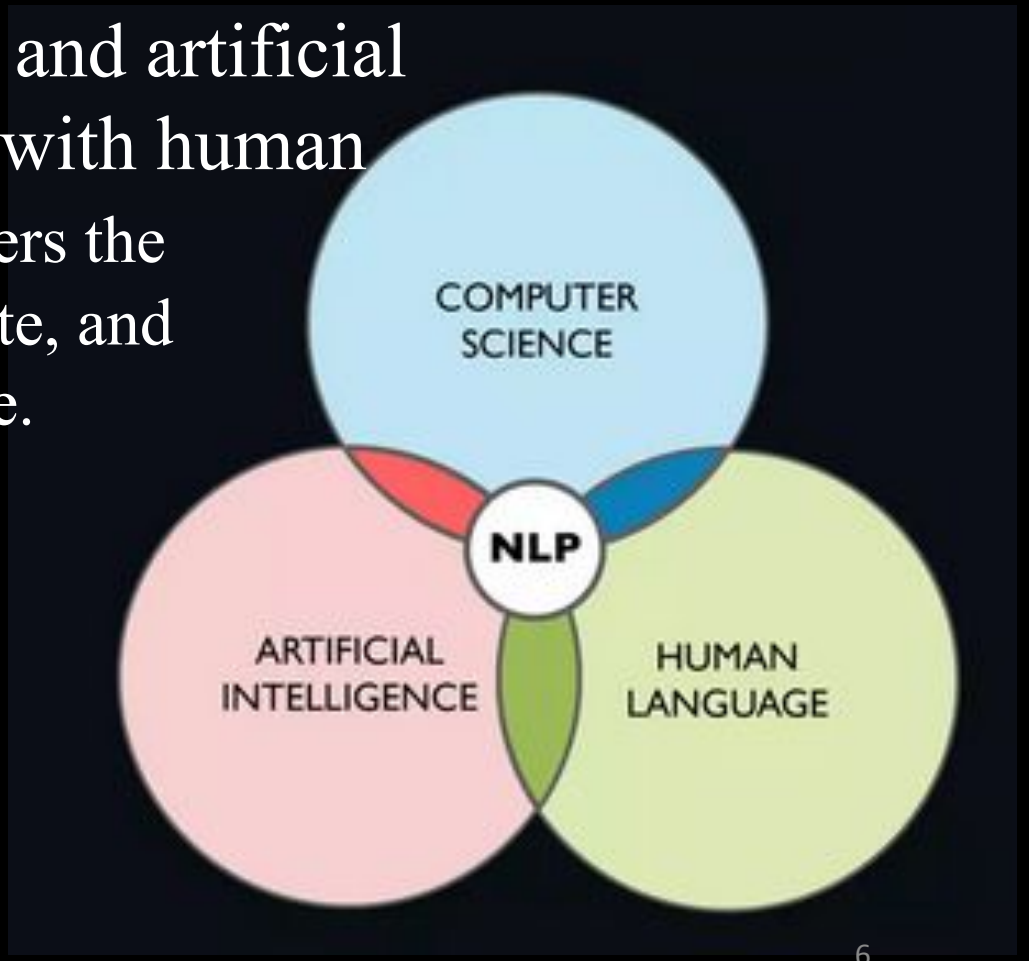


Layers of ChatBot



Natural Language Processing(NLP):

NLP: Natural Language Processing is a part of computer science and artificial intelligence which deals with human languages. It gives computers the ability to interpret, manipulate, and comprehend human language.

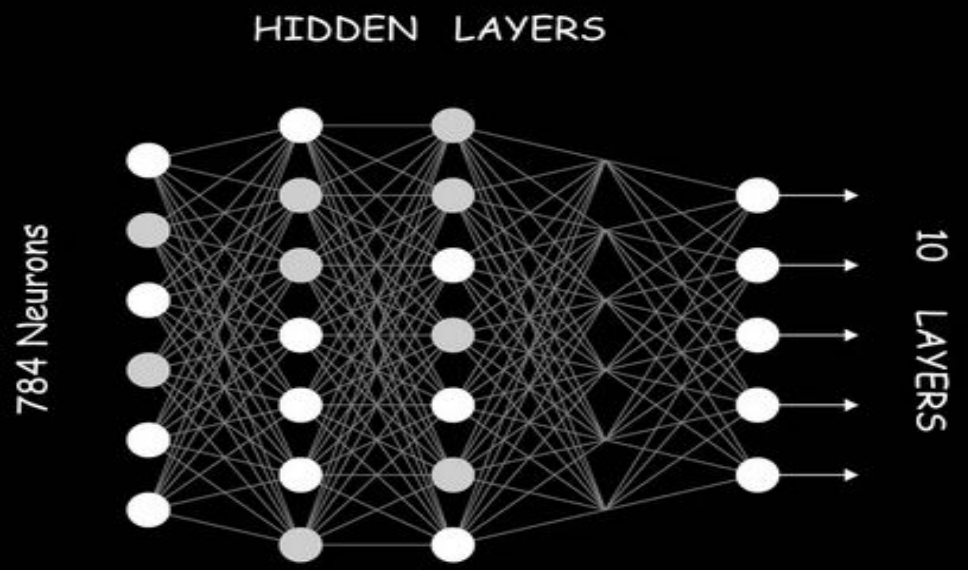


NLP

- Mapping input to useful representations
- Analyzing different aspects of the language •

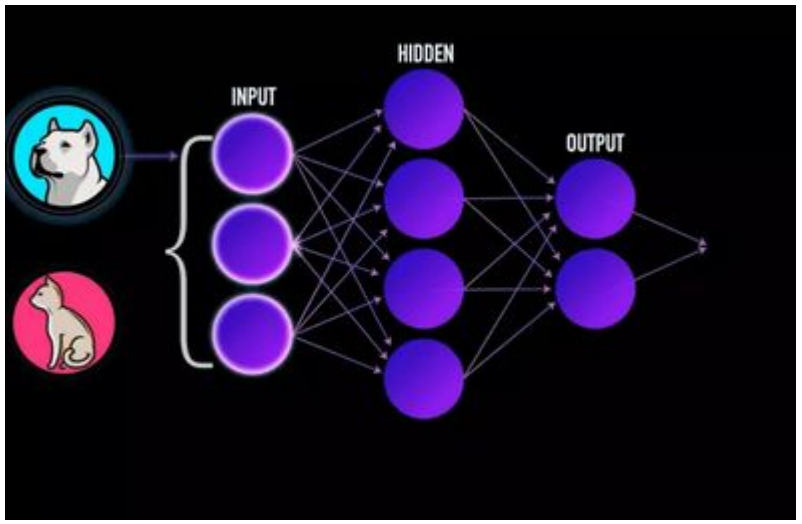
- Text Planning
- Sentence Planning
- Text Realization

2 . Neural Networks:



A neural network is a type of machine learning which models itself after the human brain.

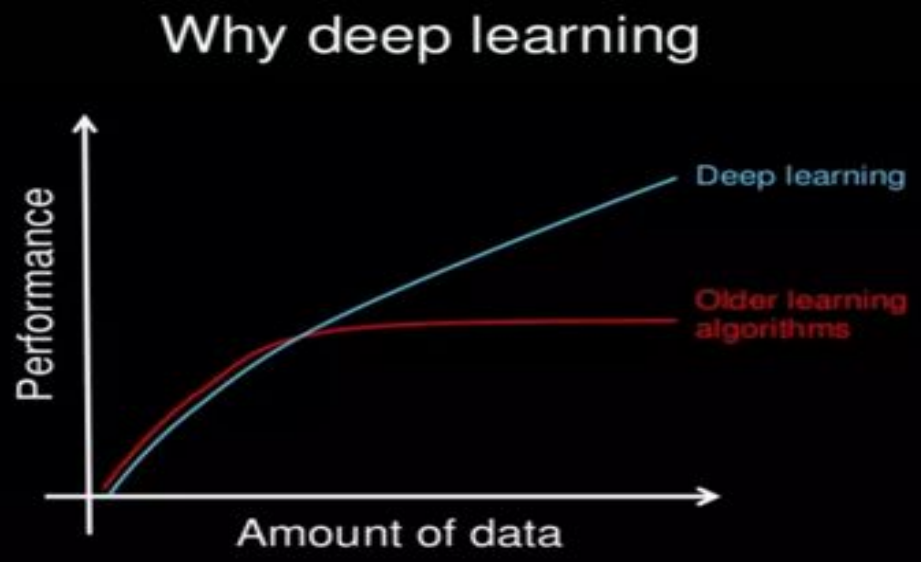
3. What is Deeplearning?

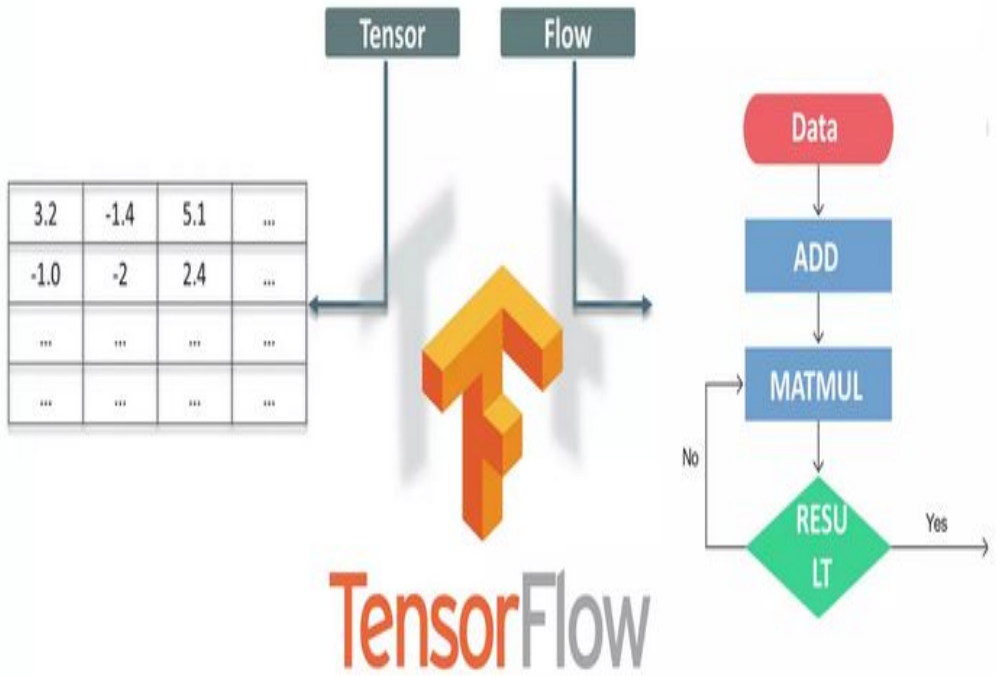


Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data

WHY DEEPLEARNING?

Built With DeepLearning.





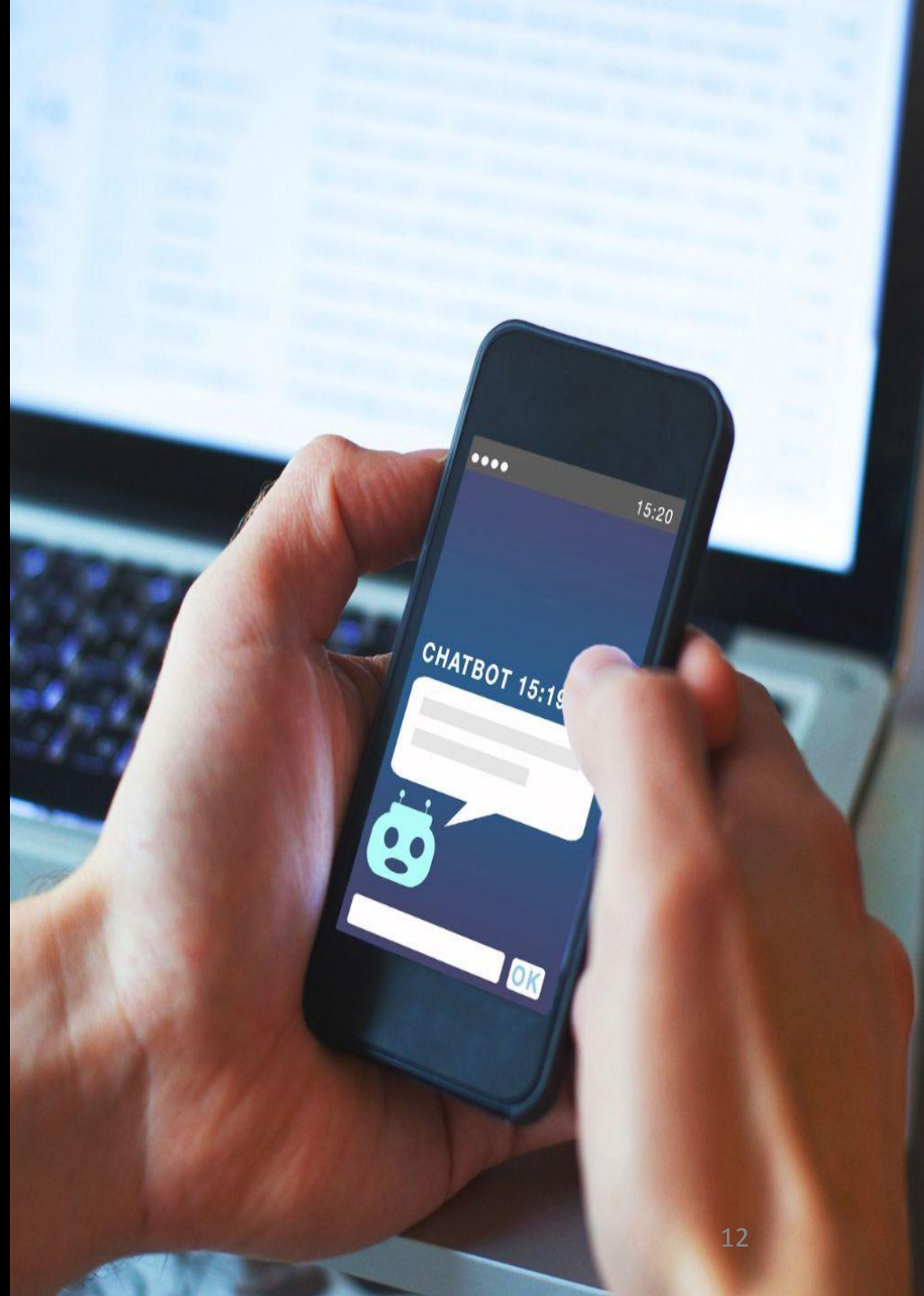
4. Tensorflow

- **Tensors** are the standard way of representing data in deep learning and **Tensorflow** computes data as a dataflow graph

- **TensorFlow** can train and run deep neural networks for handwritten digit classification and image recognition

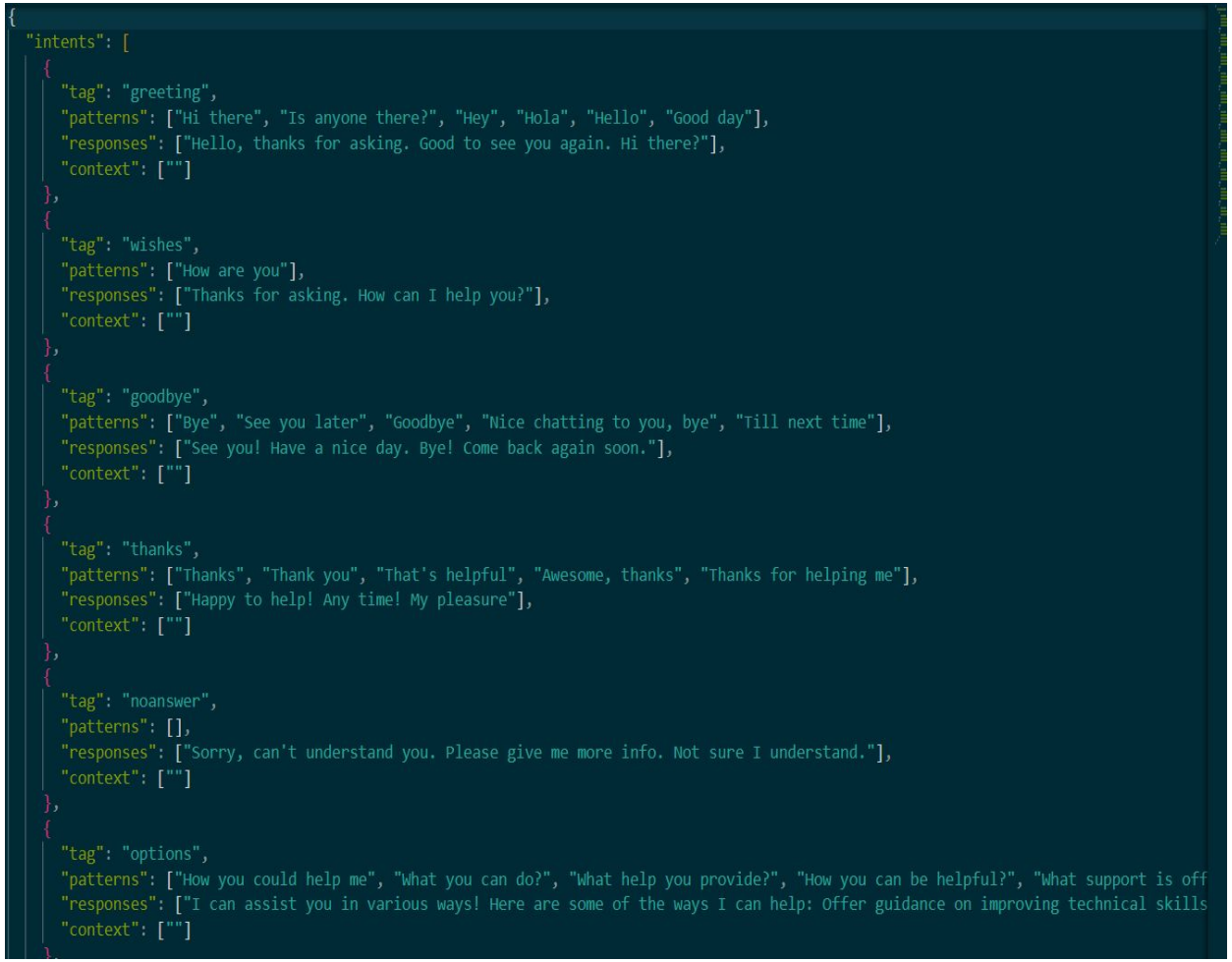
STEPS TO BE FOLLOWED:

→ Installing packages
→ Defining intents
→ Creating the training model
→Load the trained model and create flask app
→ Creating chatbot web interface —>HTML
—>CSS

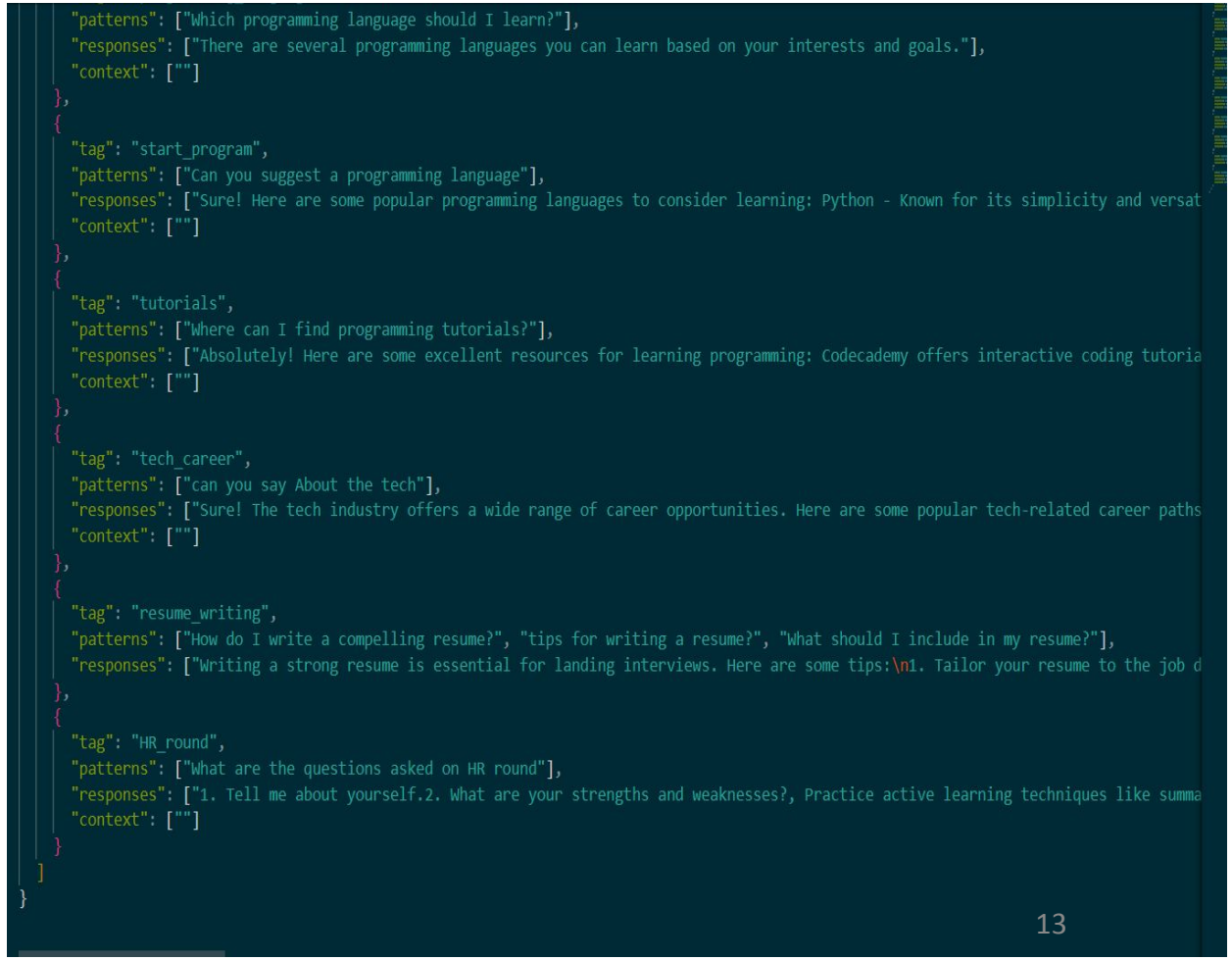


Defining Intents:

To define intents, messages, and corresponding responses, we have created a JSON file named "data.json" with the following structure as shown in figure..



```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi there", "Is anyone there?", "Hey", "Hola", "Hello", "Good day"],
      "responses": ["Hello, thanks for asking. Good to see you again. Hi there?"],
      "context": [""]
    },
    {
      "tag": "wishes",
      "patterns": ["How are you"],
      "responses": ["Thanks for asking. How can I help you?"],
      "context": [""]
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
      "responses": ["See you! Have a nice day. Bye! Come back again soon."],
      "context": [""]
    },
    {
      "tag": "thanks",
      "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
      "responses": ["Happy to help! Any time! My pleasure"],
      "context": [""]
    },
    {
      "tag": "noanswer",
      "patterns": [],
      "responses": ["Sorry, can't understand you. Please give me more info. Not sure I understand."],
      "context": [""]
    },
    {
      "tag": "options",
      "patterns": ["How you could help me", "What you can do?", "What help you provide?", "How you can be helpful?", "What support is off
      "responses": ["I can assist you in various ways! Here are some of the ways I can help: Offer guidance on improving technical skills
      "context": [""]
```



```
      "patterns": ["Which programming language should I learn?"],
      "responses": ["There are several programming languages you can learn based on your interests and goals."],
      "context": [""]
    },
    {
      "tag": "start_program",
      "patterns": ["Can you suggest a programming language"],
      "responses": ["Sure! Here are some popular programming languages to consider learning: Python - Known for its simplicity and versat
      "context": [""]
    },
    {
      "tag": "tutorials",
      "patterns": ["Where can I find programming tutorials?"],
      "responses": ["Absolutely! Here are some excellent resources for learning programming: Codecademy offers interactive coding tutoria
      "context": [""]
    },
    {
      "tag": "tech_career",
      "patterns": ["can you say About the tech"],
      "responses": ["Sure! The tech industry offers a wide range of career opportunities. Here are some popular tech-related career paths
      "context": [""]
    },
    {
      "tag": "resume_writing",
      "patterns": ["How do I write a compelling resume?", "tips for writing a resume?", "What should I include in my resume?"],
      "responses": ["Writing a strong resume is essential for landing interviews. Here are some tips:\n1. Tailor your resume to the job d
    },
    {
      "tag": "HR_round",
      "patterns": ["What are the questions asked on HR round"],
      "responses": ["1. Tell me about yourself.2. What are your strengths and weaknesses?, Practice active learning techniques like summa
      "context": [""]
    }
  ]
}
```

Training Model

In this step, we import the necessary packages required for building the chatbot. The packages include nltk, WordNetLemmatizer from nltk.stem, json, pickle, numpy, Sequential and various layers from Dense, Activation, Dropout from keras.models, and SGD from keras.optimizers. These packages are essential for performing NLP tasks and building the neural network model.

Loading and preprocessing the training data

- We collect all the unique words and intents, and finally, we create the documents by combining patterns and intents by tokenize the words.

- After preprocessing the data

- we created the training data by converting the documents into a bag-of-words representation. We iterate through each document, create a bag-of-words array with 1 if a word is present in the pattern.

Building the Neural Network Model

```
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'))
```

- The model consists of three layers: the input layer with 128 neurons, a dropout layer to prevent overfitting, a hidden layer with 64 neurons, another dropout layer, and an output layer with a number of neurons equal to the number of intents.
- The activation functions used are 'relu' for hidden layers and 'softmax' for the output layer.

Load the trained a model

The predict_class function enables the model to predict the most likely intent for a given input sentence and provides the associated probability. This allows the chatbot to understand the user's intention and generate appropriate responses based on the predicted intent.

```python
def predict_class(sentence, model):

    p = bow(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[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
```

Create the flask app

This code sets up a Flask web application with routes for the home page and receiving user input. It integrates the chatbot functionality by calling the chatbot_response function to generate responses based on user messages.

```python
from flask import Flask, render_template, request

app = Flask(__name__)
app.static_folder = 'static'

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/get")
def get_bot_response():
    userText = request.args.get('msg')
    return chatbot_response(userText)


if __name__ == "__main__":
    app.run()
```

Designing the Chatbot Web Interface

The HTML code creates a chatbot interface with a header, message display area, input field, and send button. It utilizes JavaScript to handle user interactions communicate with the server to generate bot responses dynamically. The appearance and behavior of the interface can be further customized using CSS.

When we run the code of training we get the output as model is created .

Output for flask the server is created.

```
Epoch 186/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0264
Epoch 187/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 989us/step - accuracy: 0.9657 - loss: 0.0930
Epoch 188/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9926 - loss: 0.0345
Epoch 189/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0057
Epoch 190/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 1.0000 - loss: 0.0347
Epoch 191/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0255
Epoch 192/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9835 - loss: 0.0542
Epoch 193/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 1.0000 - loss: 0.0054
Epoch 194/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9885 - loss: 0.0189
Epoch 195/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0326
Epoch 196/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0192
Epoch 197/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0189
Epoch 198/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0036
Epoch 199/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 1.0000 - loss: 0.0660
Epoch 200/200
7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0033
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recomme
ing instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
Model created
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                  python

[nltk_data]     |    C:\Users\Home\AppData\Roaming\nltk_data...
[nltk_data]     |  Package snowball_data is already up-to-date!
[nltk_data]     | Downloading package averaged_perceptron_tagger to
[nltk_data]     |    C:\Users\Home\AppData\Roaming\nltk_data...
[nltk_data]     |  Package averaged_perceptron_tagger is already up-
[nltk_data]     |      to-date!
[nltk_data]     |
[nltk_data]   Done downloading collection popular
2024-03-14 12:35:01.960504: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due t
nt round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-03-14 12:35:02.834604: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due t
nt round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2024-03-14 12:35:04.483875: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions
-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluat
 * Serving Flask app 'app'
 * Debug mode: off
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
INFO:werkzeug:Press CTRL+C to quit
```

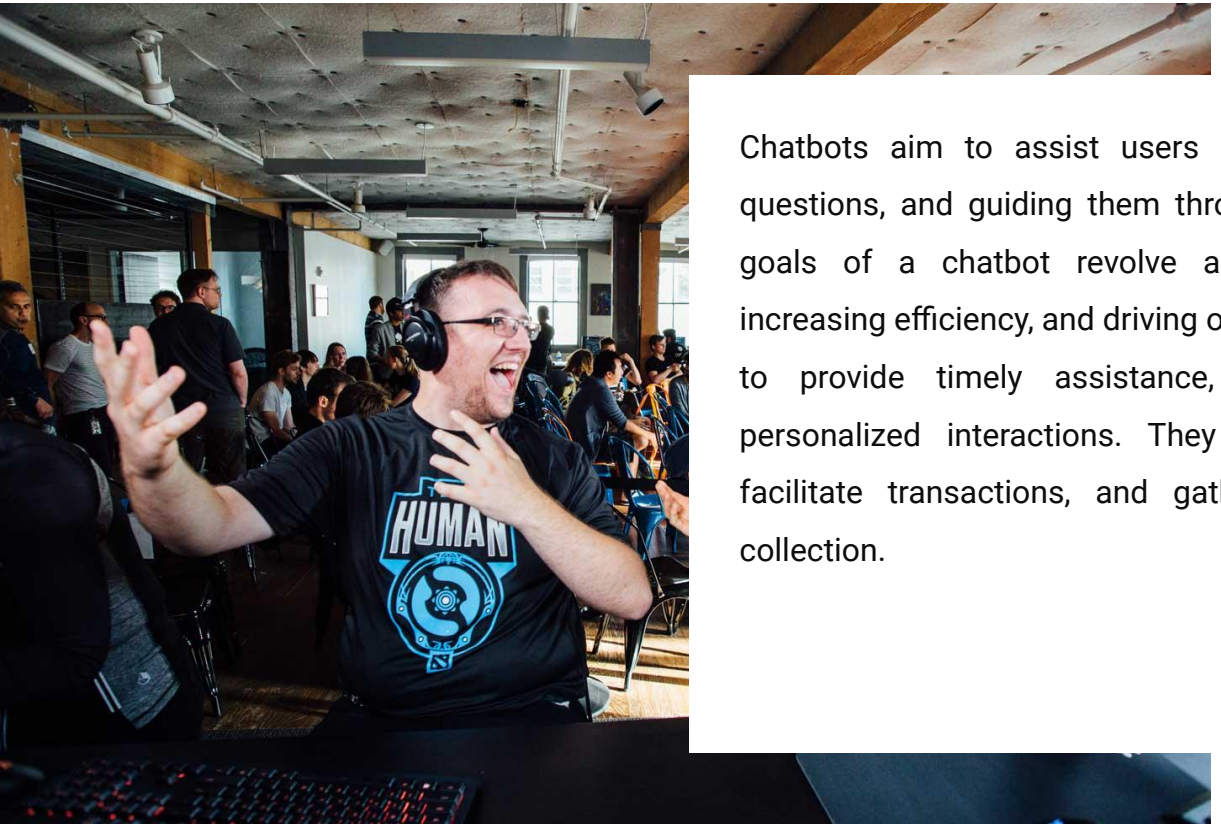
https://drive.google.com/file/d/16fxjtYDNDmPoFbS9Vz86ZX_IkIYlOpyL/view?usp=sharing

For the output

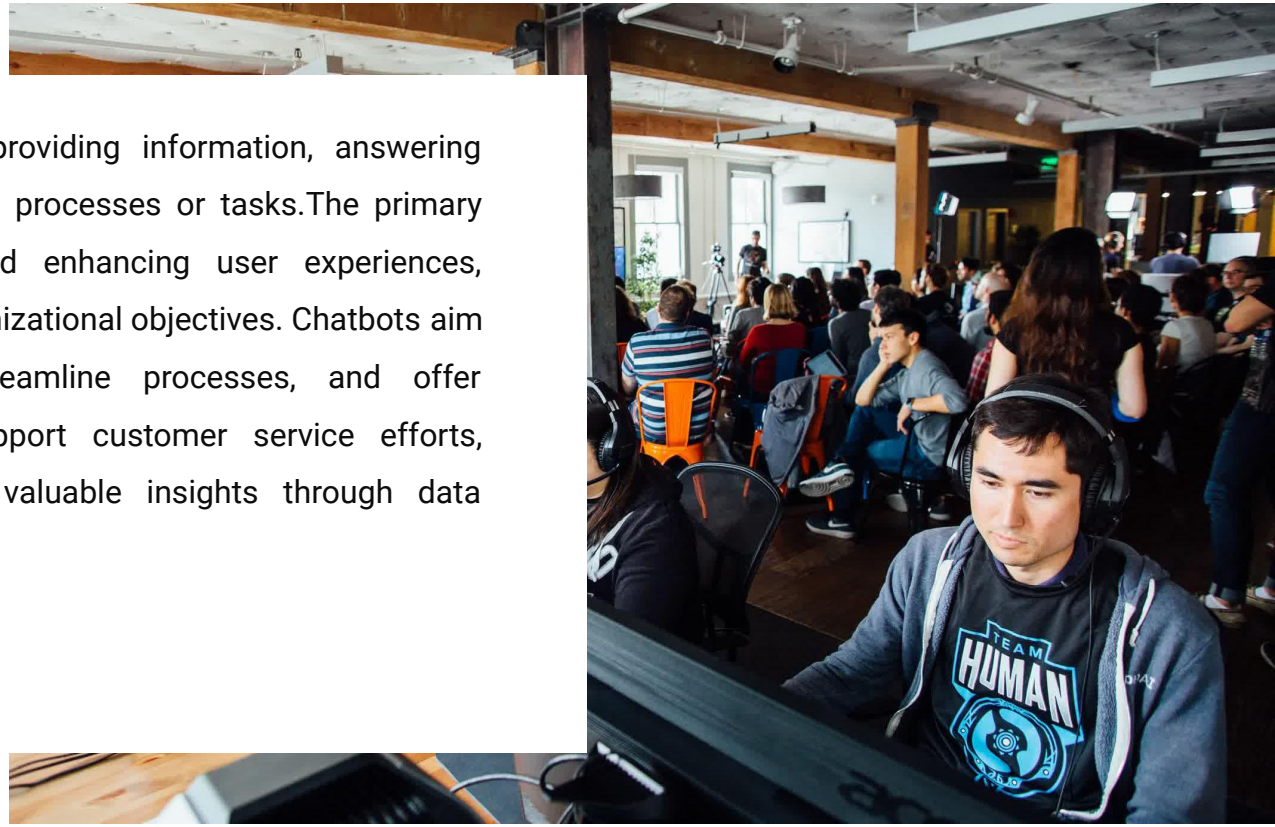
- Personal Assistance
- Question Answering
- Appointment Scheduling
- Language Translation
- Summarization

ChatBOt Usage

Goals



Chatbots aim to assist users by providing information, answering questions, and guiding them through processes or tasks.The primary goals of a chatbot revolve around enhancing user experiences, increasing efficiency, and driving organizational objectives. Chatbots aim to provide timely assistance, streamline processes, and offer personalized interactions. They support customer service efforts, facilitate transactions, and gather valuable insights through data collection.



THANK YOU