Summer Training
Report On

Technical Exploration of the Multiple Facets of Artificial Intelligence Functionality

Submitted in partial fulfilment of
the requirements for the degree of

Bachelor of Technology in
Computer Science Engineering

Submitted by
SAMEER
(CSE/21/130)

Under the guidance of
NIKHIL KUMAR

Submitted to
MS. SONIKA VASESI
ASST. PROFESSOR
B.M. Institute of Engineering & Technology
Behind Fazilpur Power Sub Station, Sec-10, Sonipat-131001

ACKNOWLEDGMENT

It gives me immense pleasure in presenting this training report on the topic **Technical Exploration of the Multiple Facets of Artificial Intelligence Functionality**. I would like to take this opportunity to express my deepest gratitude to the people, who have contributed their valuable time for helping me to successfully complete this training.

I express my deep sense of gratitude to my mentor **Mr. NIKHIL KUMAR** for their guidance and support in completing my training and project, which helped me gain this much information of the field.

I would like to express my special thanks to the Principal **Dr. Harish Mittal**, BMIET, who gave me the golden opportunity to do the training and wonderful project. **Prof. Sonika Vasesi**, my training mentor assigned by the college, is a boon to me in completing the report.

This training is a result of the contribution of many. They have enriched my knowledge by making suggestions based on their experience. I also thank my family members for their cooperation in completing the project successfully. Finally, I would like to thank my colleagues and all the people who were directly and indirectly involved in the activity.

_____

Sameer
CSE/21/130

C.ID: 0a27185

CodSoft

CERTIFICATE

OF COMPLETION
PROUDLY PRESENTED TO

Sameer

has successfully completed 4 weeks of a virtual internship program in

Artificial Intelligence

with wonderful remarks at **CODSOFT** from **05/08/2023 to 05/09/2023.**

We were truly amazed by his/her showcased skills and invaluable contributions to the tasks and projects throughout the internship.

Founder

contact@codsoft.in www.codsoft.in Date: **07/09/2023**

ABSTRACT

This internship report presents a comprehensive overview of my experience and contributions during my internship in the field of Artificial Intelligence (AI). The internship, conducted at CodSoft, spanned 05/08/2023 to 05/09/2023 and provided valuable insights into the practical application of AI technologies in real-world scenarios.

A significant portion of the report is dedicated to detailing the projects and tasks undertaken during the internship. These projects encompassed a diverse range of AI applications, from natural language processing and computer vision to predictive modeling and recommendation systems. Each project is discussed in terms of its objectives, methodologies, challenges faced, and the solutions implemented.

Additionally, the report delves into the impact of AI on various industries, emphasizing the potential for innovation and optimization in sectors such as healthcare, finance, and autonomous systems. It explores the ethical considerations associated with AI and highlights the importance of responsible AI development and deployment.

Furthermore, the report reflects on the skills and knowledge gained during the internship, including proficiency in programming languages, data manipulation, model development, and project management. It also touches upon the professional growth and networking opportunities provided by the internship experience.

In conclusion, this internship report underscores the significance of AI in shaping the future of technology and presents a detailed account of the practical applications and challenges encountered during the internship period. It serves as a valuable resource for both academic and industry professionals interested in the dynamic field of Artificial Intelligence

TABLE OF CONTENTS

1. INTRODUCTION

1.1 WHAT IS AI?

AI, or Artificial Intelligence, refers to the simulation of human intelligence in machines that are programmed to think and learn like humans. It involves the development of computer systems or software that can perform tasks that typically require human intelligence. These tasks can include problem-solving, speech recognition, natural language understanding, decision-making, and even visual perception. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—such as discovering proofs for mathematical theorems or playing chess—with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match full human flexibility over wider domains or in tasks requiring much everyday knowledge. On the other hand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, voice or handwriting recognition, and chatbots.

1.2 IMPACT OF AI

The impact of AI, or Artificial Intelligence, on society, the economy, and various industries has been significant and continues to grow. Here are some key aspects of AI's impact:

1.**Automation of Tasks:** AI has the potential to automate repetitive and mundane tasks, which can increase efficiency and productivity in various industries. For example, in manufacturing, robots and AI systems can handle repetitive assembly line tasks, reducing the need for human labour.

2.**Improved Decision-Making**: AI can analyse vast amounts of data quickly and make data-driven recommendations or decisions. This is particularly valuable in fields like finance, where AI algorithms can identify trends and make investment predictions.

3.**Healthcare Advancements**: AI has been used to analyse medical images, assist in diagnosing diseases, and even develop new drugs. It can help doctors make more accurate diagnoses and improve patient outcomes.

4.**Enhanced Customer Experiences**: In the business world, AI-powered chatbots and virtual assistants are being used to provide better customer support and streamline customer interactions. Personalization through AI-driven recommendations is also common in e-commerce and entertainment.

5.**Autonomous Vehicles**: AI is a fundamental technology in the development of self-driving cars and drones. These technologies have the potential to transform transportation, making it safer and more efficient.

6.**Natural Language Processing**: AI-driven language understanding and generation have led to improvements in voice assistants, translation services, and even content creation.

7.**Challenges and Concerns**: The impact of AI is not without its challenges. Concerns related to privacy, bias in AI algorithms, job displacement due to automation, and ethical considerations in AI development and use are areas of concern that require careful management.

8.**Economic Disruption**: AI can lead to economic disruption as some industries undergo major changes due to automation. However, it can also create new job opportunities in fields related to AI development and maintenance.

9.**Global Competition**: The development and deployment of AI technologies have led to global competition among countries and companies. Governments are investing in AI research to gain a competitive edge in the global economy.

10.**Ethical and Regulatory Considerations**: As AI becomes more powerful, there is a growing need for ethical guidelines and regulations to ensure responsible AI development and deployment.

The impact of AI is broad and multifaceted, affecting various aspects of our lives. While it offers numerous benefits in terms of automation, decision support, and technological advancement, it also presents challenges and ethical considerations that require careful attention as AI continues to evolve and integrate into society.

1.3 DIVERSE RANGE OF AI

AI encompasses a diverse range of subfields and applications, each with its own set of techniques and purposes. Here are some of the diverse areas within AI:

Machine Learning (ML): ML is a fundamental component of AI that focuses on developing algorithms and models that allow computers to learn from and make predictions or decisions based on data. It includes subfields like supervised learning, unsupervised learning, and reinforcement learning.

Deep Learning: Deep learning is a subset of ML that uses artificial neural networks to model and solve complex problems. It's particularly powerful in tasks involving image recognition, natural language processing, and speech recognition.

Natural Language Processing (NLP): NLP focuses on enabling computers to understand, process, and generate human language. Applications range from chatbots and virtual assistants to language translation and sentiment analysis.

Computer Vision: Computer vision aims to give machines the ability to interpret and understand visual information from the world, including images and videos. It has applications in facial recognition, object detection, and autonomous vehicles.

Robotics: Robotics combines AI with physical systems to create machines that can perform tasks autonomously or semi-autonomously. Examples include industrial robots, drones, and surgical robots.

Expert Systems: Expert systems mimic the decision-making abilities of human experts in specific domains. They are used in fields like healthcare for medical diagnosis and in finance for investment advice.

Reinforcement Learning: This subfield of ML focuses on agents learning how to make decisions through interaction with an environment and maximizing cumulative rewards. It's commonly used in game-playing AI and robotics.

Generative Adversarial Networks (GANs): GANs are a type of neural network used for generating data, such as realistic images or even text. They consist of a generator and a discriminator that compete with each other, leading to high-quality data generation.

AI in Healthcare: AI is used in healthcare for tasks like disease diagnosis, drug discovery, and patient data analysis. It can improve patient care and assist medical professionals.

Autonomous Vehicles: AI plays a crucial role in the development of self-driving cars and autonomous drones, which have the potential to transform transportation.

AI in Finance: AI is used in financial institutions for fraud detection, algorithmic trading, credit scoring, and risk assessment.

AI in Gaming: AI techniques are employed in the gaming industry to create realistic computer opponents, procedural content generation, and game testing.

2. TASKS

To get familiarized with the field of Artificial Intelligence a series of tasks were assigned. The tasks assigned to the intern in the field of Artificial Intelligence are both challenging and intellectually stimulating. This internship offers a unique chance to gain practical experience in the dynamic and rapidly evolving field of AI, contributing to cutting-edge research and innovation while honing their skills in AI development and implementation.
The various tasks assigned are discussed in detail in the following pages.

3. CHATBOT WITH RULE-BASED RESPONSES

3.1 WHAT ARE CHATBOTS?

Chatbots are computer programs or AI applications designed to simulate human conversation. They are typically used to interact with users via text-based or voice-based communication channels, such as messaging apps, websites, and voice assistants. Chatbots are programmed to understand and respond to user queries and provide information or perform tasks automatically, without the need for human intervention. Chatbots have gained popularity across various industries, including customer service, e-commerce, healthcare, and finance, as they offer efficient and automated ways to engage with users, enhance user experiences, and streamline business operations.

3.2 TYPES OF CHATBOTS

There are different types of chatbots:

1.Rule-Based Chatbots: These chatbots follow predefined rules and decision trees. They can handle simple interactions but may struggle with more complex or context-aware conversations.

2.AI-Powered Chatbots: These chatbots utilize machine learning and AI algorithms to understand and respond to user inputs. They can adapt and improve their responses over time based on user interactions.

3.Voice Assistants: Voice-activated chatbots like Siri, Google Assistant, and Alexa respond to spoken commands and questions, often integrating with other smart devices.

4.Transactional Chatbots: These chatbots focus on specific tasks, such as processing online orders, booking appointments, or providing financial services.

3.3 RULE-BASED CHATBOTS

A rule-based chatbot is an automated system designed to answer questions that are commonly asked by customers and perform simple tasks. It follows a set of preprogrammed rules that are designed to answer almost any customer enquiry or resolve their problems by following the appropriate response. This makes it extremely useful for customers looking for self-service options or a way to quickly route them to the most appropriate person to handle their unique query.

Rule-based chatbots are incredibly easy to use. When you sign up for one, you will have to think about the common enquiries your customers have and convert them into questions. Once the bot has enough information, it'll follow a list of rules to generate a response, and answer the customer's question. This process is much faster than having to go through a set of questions each time you receive an inquiry - which can add up quickly when you're dealing with many customers every day.

3.4 TASK DESCRIPTION

Build a simple chatbot that responds to user inputs based on predefined rules. Use if-else statements or pattern matching techniques to identify user queries and provide appropriate responses. This will give you a basic understanding of natural language processing and conversation flow.

4. CODE IMPLEMENTATION

CODE

```python
import re
#pre-defined rules
rules=[
    (r"hello|hi", "Hello! How can I help you?"),
    (r"who are you?", "I am Alex, your virtual assistant. What can I do for you today?"),
    (r"what is a chatbot?", " A chatbot is a computer program that simulates and processes human conversation either written or spoken, a
    (r"how are you?", "I am doing well, thank you for asking"),
    (r"bye", "Goodbye! Have a great day!")

]
#ai's responses
def alex_response(user_input):
    user_input=user_input.lower()
    for rule, response in rules :
        if re.search(rule, user_input):
            return response

    return "I am sorry, I don't think I understand that."

while True:
    user_input=input("You:")
    if user_input.lower()=="exit":
        print("Goodbye!")
        break
    elif user_input.lower()=="bye":
        response= alex_response(user_input)
        print("Alex:",response)
        break
    else:
        response= alex_response(user_input)
        print("Alex:",response)
```

OUTPUT

```
PS C:\Users\Sameer\Desktop\assignment> & C:/Users/Sameer/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Sameer/Desktop/alex.py
You:hi
Alex: Hello! How can I help you?
You:how are you?
Alex: I am doing well, thank you for asking
You:who are you?
Alex: I am Alex, your virtual assistant. What can I do for you today?
You:what is a chatbot?
Alex:  A chatbot is a computer program that simulates and processes human conversation either written or spoken, allowing humans to interact with digital d
evices as if they were communicating with a real person.
You:bye
Alex: Goodbye! Have a great day!
PS C:\Users\Sameer\Desktop\assignment>
```

5. TIC-TAC-TOE AI

5.1 TASK DESCRIPTION

Implement an AI agent that plays the classic game of Tic-Tac-Toe against a human player. You can use algorithms like Minimax with or without Alpha-Beta Pruning to make the AI player unbeatable.
This project will help you understand game theory and basic search algorithms.

5.2 GAME THEORY

In the context of Artificial Intelligence (AI), game theory is often used to model and analyse decision-making in situations where AI agents interact with each other or with human players. Here are some ways game theory is applied in AI:

1.Multi-Agent Systems: Game theory is fundamental in multi-agent systems, where multiple AI agents or entities interact to achieve their objectives. These agents can be competing, cooperating, or both. Game theory helps model their interactions and strategies.

2.Reinforcement Learning: In reinforcement learning, AI agents learn optimal strategies through trial and error. Game theory concepts are used to model these interactions, particularly in competitive environments like adversarial reinforcement learning.

3.Adversarial Games: Game theory is extensively applied in adversarial settings, such as chess and go, where AI agents compete against human players or other AI agents. It helps in developing strategies to maximize the AI's chances of winning.

4.Auction Mechanisms: Auctions involve multiple bidders competing for limited resources. Game theory is used to design auction mechanisms that ensure fair and efficient resource allocation, such as in online advertising or spectrum auctions.

5.Algorithmic Game Theory: This field combines algorithms and game theory to analyse and design systems where self-interested agents make decisions. It's used in various AI applications, including resource allocation and market design.

6.Cooperative Games: While game theory often focuses on competition, it's also applied in scenarios where AI agents cooperate. Cooperative game theory helps model how agents can form coalitions and distribute rewards or resources fairly.

7.Nash Equilibria: AI researchers use Nash equilibria to analyze and find stable points in multi-agent systems, which helps predict how agents will behave when pursuing their goals.

8.Robotic Coordination: In robotics, multiple AI-controlled robots may need to coordinate their actions to complete tasks efficiently. Game theory provides tools for analysing and optimizing their coordination strategies.

5.3 MINIMAX ALGORITHM

The minimax algorithm is a decision-making algorithm used in two-player games, such as chess, checkers, and tic-tac-toe, where one player (the "maximizer") seeks to maximize their chances of winning, and the other player (the "minimizer") seeks to minimize the maximizer's chances. The primary goal of the minimax algorithm is to determine the best move for the maximizer, assuming that the minimizer also plays optimally. It is a fundamental concept in AI game-playing agents and serves as the basis for many game-playing AI systems. It provides a structured approach to decision-making in competitive games, allowing AI agents to make strategic choices that maximize their chances of winning.

Here's how the minimax algorithm works:

1.Game Tree: The minimax algorithm begins by constructing a game tree that represents all possible moves and outcomes of the game from the current state. This tree is usually built recursively, with each level representing a player's turn.

2.Maximizer and Minimizer: In each level of the tree, the algorithm alternates between the maximizer and minimizer. The maximizer selects moves that maximize their chances of winning, while the minimizer selects moves that minimize the maximizer's chances.

3.Evaluation Function: To assess the quality of each game state, an evaluation function is used. This function assigns a score or value to each state, indicating how favourable it is for the maximizer. In chess, for example, this might involve factors like piece values, board position, and material advantage.

4.Recursion: The minimax algorithm explores the game tree by recursively evaluating the possible moves and outcomes. At each level, the algorithm computes the best move for the current player (either maximizing or minimizing) based on the evaluation function and the scores of the child nodes.

5.Backpropagation: The algorithm propagates the best move choices up the tree, ultimately determining the best move for the initial state of the game. This move is the one that leads to the highest score for the maximizer.

6.Pruning (Optional): To improve efficiency, the minimax algorithm often employs pruning techniques, such as alpha-beta pruning, to eliminate branches of the game tree that are guaranteed to be suboptimal. This reduces the number of nodes that need to be evaluated.

6. CODE IMPLEMENTATION

CODE

tic-tac-toe > trial.py > winner

```python
def initialize_board():
    return [' '] * 9
#board
def display_board(board):
    print(board[0] + '|' + board[1] + '|' + board[2])
    print('-+-+-')
    print(board[3] + '|' + board[4] + '|' + board[5])
    print('-+-+-')
    print(board[6] + '|' + board[7] + '|' + board[8])

#winning combinations
def winner(board, player):
    win_combos = [
        [0, 1, 2], [3, 4, 5], [6, 7, 8],
        [0, 3, 6], [1, 4, 7], [2, 5, 8],
        [0, 4, 8], [2, 4, 6]
    ]
    for combo in win_combos:
        if all(board[i] == player for i in combo):
            return True
    return False

#checking if board is full
def board_full(board):
    return all(cell != ' ' for cell in board)

#apply minimax algorithm
def minimax(board, depth, maximizing_player):
    if winner(board, 'X'):
        return 1
    if winner(board, 'O'):
        return -1
    if board_full(board):
        return 0
```

```
    if maximizing_player:
        max_eval = float('-inf')
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'X'
                eval = minimax(board, depth + 1, False)
                board[i] = ' '
                max_eval = max(max_eval, eval)
        return max_eval
    else:
        min_eval = float('inf')
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'O'
                eval = minimax(board, depth + 1, True)
                board[i] = ' '
                min_eval = min(min_eval, eval)
        return min_eval

def find_best_move(board):
    best_move = -1
    best_eval = float('-inf')
    for i in range(9):
        if board[i] == ' ':
            board[i] = 'X'
            eval = minimax(board, 0, False)
            board[i] = ' '
            if eval > best_eval:
                best_eval = eval
                best_move = i
    return best_move
```

```python
#game function
def play_game():
    board = initialize_board()
    display_board(board)

    while True:
        move = int(input("Enter your move (0-8): "))
        if board[move] == ' ':
            board[move] = 'O'
        else:
            print("Invalid move! Cell is already occupied.")
            continue

        display_board(board)

        if winner(board, 'O'):
            print("You win!")
            break
        if board_full(board):
            print("It's a tie!")
            break

        ai_move = find_best_move(board)
        board[ai_move] = 'X'
        print("AI's move:")
        display_board(board)

        if winner(board, 'X'):
            print("AI wins!")
            break
        if board_full(board):
            print("It's a tie!")
            break
play_game()
```

```python
#want to play again?
while True:
    a= input("do you want to play again?[y/n]")
    if a=="y":
        play_game()
    else:
        print("thank you or playing")
        break

```

OUTPUT

```
 | |
-+-+-
 | |
-+-+-
 | |
Enter your move (0-8): 0
O| |
-+-+-
 | |
-+-+-
 | |
AI's move:
O| |
-+-+-
 |X|
-+-+-
 | |
Enter your move (0-8): 5
O| |
-+-+-
 |X|O
-+-+-
 | |
AI's move:
O|X|
-+-+-
 |X|O
-+-+-
 | |
Enter your move (0-8): 2
```

```
O|X|O
-+-+-
 |X|O
-+-+-
 | |
AI's move:
O|X|O
-+-+-
 |X|O
-+-+-
 |X|
AI wins!
do you want to play again?[y/n]N
thank you or playing
PS C:\Users\Sameer\Desktop\tic-tac-toe>
```

7. FACE DETECTION SYSTEM

7.1 TASK DESCRIPTION

Develop an AI application that can detect and recognize faces in images or videos. Use pre-trained face detection models like Haarcascades or deep learning-based face detectors.

7.2 HAARCASCADE MODEL

A Haar Cascade is a machine learning object detection algorithm used to identify objects or features within images or videos. It is particularly popular for detecting faces and various other objects in computer vision applications. The algorithm is named after its inventor, Viola and Jones, who introduced it in their 2001 paper titled "Rapid Object Detection using a Boosted Cascade of Simple Features."

Here's how the Haar Cascade object detection algorithm works:

1.Haar Features: Haar features are simple rectangular filters that are applied to a portion of an image. These features can detect variations in intensity (brightness) in different regions of an image. Examples of Haar features include edge features, line features, and rectangular region features.

2.Integral Image: To efficiently compute Haar features for various image regions, an integral image is used. The integral image is a transformed version of the original image that allows for fast computation of sums of pixel values within rectangular regions.

3.Training: The Haar Cascade algorithm requires training with positive and negative examples. Positive examples are images or subregions of images containing the object or feature of interest (e.g., faces), while negative examples are images that do not contain the object. During training, the algorithm learns to distinguish between positive and negative examples by selecting and combining an optimal set of Haar features.

3.Cascade Classifier: The trained Haar Cascade classifier is organized into multiple stages or levels, each consisting of a subset of Haar features. The stages are arranged in a cascade, where early stages are less computationally intensive

and aim to quickly reject regions of the image that are unlikely to contain the object. If a region passes all stages, it is considered a positive detection.

4.Sliding Window: To detect objects in an image, a sliding window is used to move across the image, applying the cascade classifier at various positions and scales. At each step, the classifier evaluates whether the current image region contains the object of interest.

5.Thresholding: The classifier employs a thresholding mechanism to make decisions at each stage. If a region's features meet a certain threshold, it proceeds to the next stage; otherwise, it is rejected as a negative example.

Haar Cascade classifiers are commonly used for tasks such as face detection, eye detection, pedestrian detection, and more. They are known for their real-time or near-real-time performance and have been utilized in a variety of applications, including security systems, video surveillance, and augmented reality. OpenCV, a popular computer vision library, provides pre-trained Haar Cascade classifiers for various objects, making it easier to implement object detection in applications.

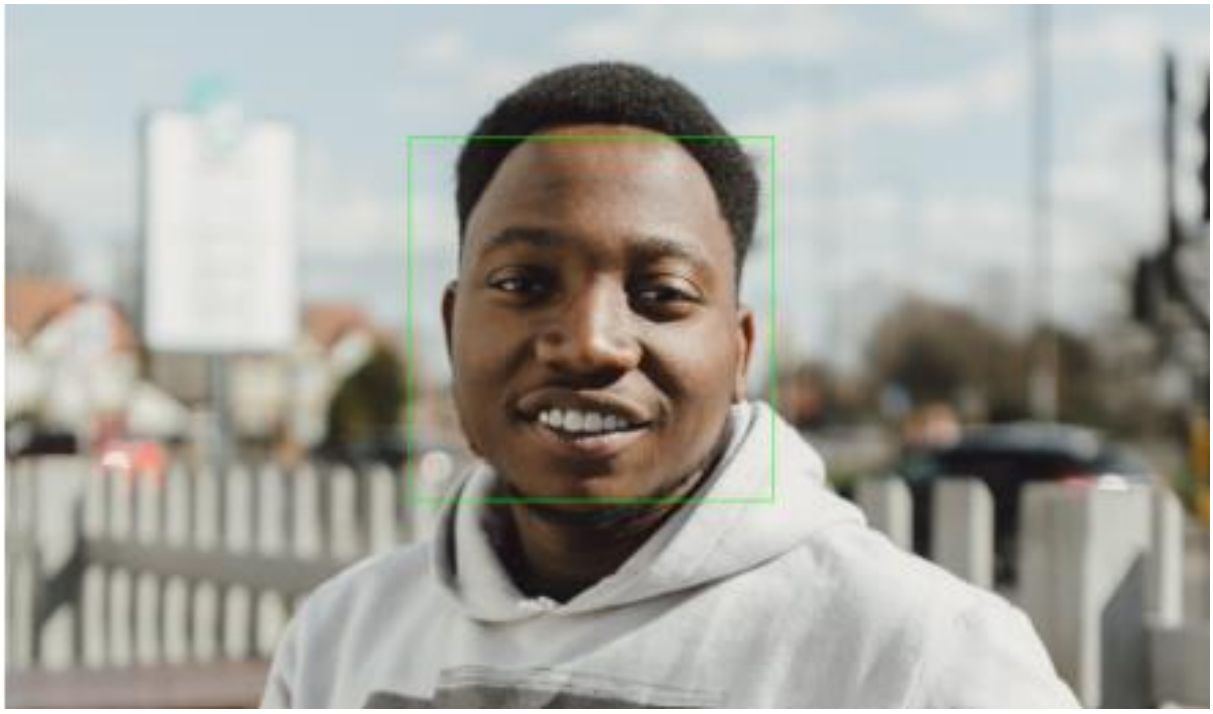
8. CODE IMPLEMENTATION

CODE

image > face_detection.py > ...

```python
import cv2
import matplotlib.pyplot as plt

# Load the image
image_path = input('Enter the name of image:')
img = cv2.imread(image_path+'.jpg')

# Check if the image was loaded successfully
if img is None:
    raise ValueError("Image not found at the specified path.")

# Convert the image to grayscale
grey = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Load the Haar Cascade classifier for face detection
face_classifier = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

# Detect faces in the grayscale image
faces = face_classifier.detectMultiScale(
    grey, scaleFactor=1.1, minNeighbors=5
)

# Draw rectangles around detected faces
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Convert BGR image to RGB
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Display the image with rectangles
plt.figure(figsize=(12, 8))
plt.imshow(img_rgb)
plt.axis('off')
plt.show()
```

OUTPUT



9. RECOMMENDATION SYSTEM

9.1 TASK DESCRIPTION

Create a simple recommendation system that suggests items to users based on their preferences. You can use techniques like collaborative filtering or content-based filtering to recommend movies, books, or products to users.

9.2 CONTENT FILTERING

Content filtering in recommendation systems is a technique used to personalize recommendations based on the content attributes or characteristics of items and the preferences of users. It involves analyzing and matching the content features of items (such as text, keywords, or metadata) with user profiles to suggest relevant items. Content-based filtering is one of the two primary approaches to recommendation systems, with the other being collaborative filtering.

Here's how content filtering works in recommendation systems:

1.Item Representation: Each item in the recommendation system is represented by its content attributes. For example, in a movie recommendation system, content attributes might include movie genres, director, actors, plot keywords, and user reviews.

2.User Profiles: User profiles are created to capture the preferences and interests of individual users. These profiles are typically based on their past interactions, ratings, or explicit feedback on items. The user profile may also contain demographic information.

3.Matching Content to User Profiles: Content-based filtering calculates a similarity score between the content attributes of items and the user's profile. Common similarity measures include cosine similarity and Jaccard similarity. The higher the similarity score between an item's content and the user's profile, the more likely it is to be recommended.

4.Recommendation Generation: Based on the similarity scores, the recommendation system generates a list of items that are most similar to the user's preferences and

interests. These items are then presented to the user as personalized recommendations.

9.3 MOVIE RECOMMENDATION SYSTEM

A movie recommendation system is a type of recommendation system designed to suggest movies to users based on their preferences and past interactions with movies. These systems leverage various algorithms and techniques to provide personalized movie recommendations, enhancing the user's entertainment experience. Here's an overview of how a movie recommendation system typically works:

Data Collection and Preparation:

The recommendation system collects data on movies, including information like titles, genres, release dates, actors, directors, user ratings, reviews, and more.User data is also collected, including user profiles, preferences, ratings, and viewing history.

Data Preprocessing:

Data preprocessing involves cleaning and organizing the collected data. This may include handling missing values, standardizing data formats, and ensuring data consistency.

Feature Extraction:

Features are extracted from movie data, which can include genre, director, actor, and keywords. These features help describe the content of the movies.

User Profiling:

User profiles are created based on user interactions and preferences. These profiles may include demographic information, historical movie ratings, and viewing history.

Recommendation Algorithms:

Movie recommendation systems use various algorithms to suggest movies to users.

Scoring and Ranking:

The recommendation system scores and ranks movies based on the user's preferences and the chosen recommendation algorithm. Movies with higher scores are suggested to the user.

Filtering and Personalization:

The system can apply additional filters such as removing already-watched movies or filtering out movies with low user ratings.

Recommendations are personalized to each user based on their unique preferences and behaviour.

Presentation to Users:

Recommendations are presented to users through various interfaces, such as web apps, mobile apps, or smart TVs.Users can browse and interact with the recommended movies, read descriptions, watch trailers, and make selections.

10. CODE IMPLEMENTATION

CODE

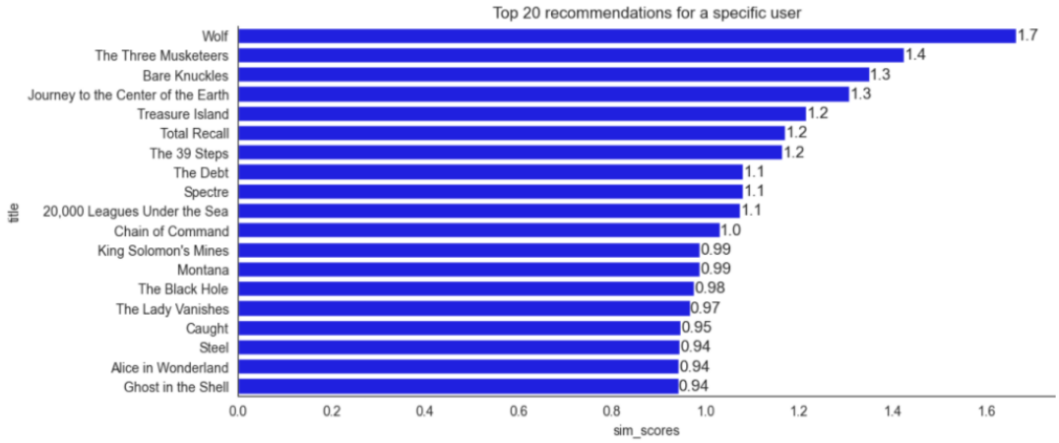
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
matplotlib.use('TkAgg')
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import ast
path=r'C:\Users\Sameer\Desktop\movies_metadata.csv'
df_data= pd.read_csv(path, low_memory=False)
df_data= df_data[df_data['vote_count'].notna()]
plt.figure(figsize=(20,10))
sns.displot(df_data['vote_count'])
plt.title("histogram of vote counts")
min_votes=np.percentile(df_data['vote_count'].values, 85)
df=df_data.copy(deep=True).loc[df_data['vote_count']> min_votes]
df=df[df['overview'].notna()]
df.reset_index(inplace=True)
def process_text(text):
    text=' '.join(text.split())
    text=text.lower()
    return text
df['overview']= df.apply(lambda x: process_text(x.overview), axis=1)
tf_idf= TfidfVectorizer(stop_words='english')
tf_idf_matrix= tf_idf.fit_transform(df['overview']);
cosine_similarity_matrix= cosine_similarity(tf_idf_matrix, tf_idf_matrix)
def index_from_title(df,title):
    return df[df['original_title']==title].index.values[0]
def title_from_index(df,index):
    return df[df.index==index].original_title.values[0]
def recommendations(original_title,df,cosine_similarity_matrix,number_of_recommendations):
    index=index_from_title(df,original_title)
    similarity_scores=list(enumerate(cosine_similarity_matrix[index]))
    similarity_scores_sorted=sorted(similarity_scores, key=lambda x: x[1], reverse=True)
    recommendations_indices=[t[0] for t in similarity_scores_sorted[1:(number_of_recommendations+1)]]
    return df.loc[recommendations_indices,'original_title']
```

```
path1=r'C:\Users\Sameer\Desktop\archive (1)\keywords.csv'
path2=r'C:\Users\Sameer\Desktop\archive (1)\credits.csv'
df_keywords=pd.read_csv(path1)
df_credits=pd.read_csv(path2)
df_cb=df_data.copy(deep=True)[df_data.id.apply(lambda x: x.isnumeric())]
df_cb['id']=df_cb['id'].astype(int)
df_keywords['id']=df_keywords['id'].astype(int)
df_credits['id']=df_credits['id'].astype(int)
df_movies_data=pd.merge(df_cb,df_keywords, on='id')
df_movies= pd.merge(df_movies_data,df_credits, on='id')
max_number_of_actors= 3
def return_actors(cast):
    actors=[]
    count=0
    for row in ast.literal_eval(cast):
        if count<max_number_of_actors:
           actors.append(row['name'].lower().replace(" ",""))
        else:
            break
        count+=1
    return ' '.join(actors)
df_movies['actors']=df_movies.apply(lambda x: return_actors(x.cast),axis=1)
def return_producer_screenplay_director(crew,crew_type):
    persons=[]
    for row in ast.literal_eval(crew):
        if row['job'].lower()==crew_type:
            persons.append(row['name'].lower().replace(" ",""))
    return ' '.join(persons)
df_movies['director']=df_movies.apply(lambda x:return_producer_screenplay_director(x.crew,'director'),axis=1)
df_movies['screenplay']=df_movies.apply(lambda x:return_producer_screenplay_director(x.crew,'screenplay'),axis=1)
df_movies['producer']=df_movies.apply(lambda x:return_producer_screenplay_director(x.crew,'producer'),axis=1)
w_genre=2
```

```
w_keywords=3
w_actors=3
w_director=100
w_producer=1
w_screenplay=1
def concatenate_features(df_row):
    genres=[]
    for genre in ast.literal_eval(df_row['genres']):
        genres.append(genre['name'].lower())
    genres=' '.join(genres)
    keywords=[]
    for keyword in ast.literal_eval(df_row['keywords']):
        keywords.append(keyword['name'])
        keywords=' '.join(keywords)
    return ' '.join([genres]*w_genre)+''+' '.join([keywords]*w_keywords)+''+' '.join([df_row['actors']]*w_actors)+''+' '.join([df_row['pr
df_movies['features']=df_movies.apply(concatenate_features,axis=1)
def process_text(text):
    text=' '.join(text.split())
    text=text.lower()
    return text
```

```
df_movies['features']=df_movies.apply(lambda x:process_text(x.features),axis=1)
vect=CountVectorizer(stop_words='english')
vect_matrix=vect.fit_transform(df_movies['features'])
cosine_similarity_matrix_count_based=cosine_similarity(vect_matrix,vect_matrix)

df_movies['features']=df_movies.apply(concatenate_features,axis=1)
vect=CountVectorizer(stop_words='english')
vect_matrix=vect.fit_transform(df_movies['features'])
cosine_similarity_matrix_count_based=cosine_similarity(vect_matrix,vect_matrix)
recommendations('Batman',df_movies,cosine_similarity_matrix_count_based,8)

```

OUTPUT


Top 20 recommendations for a specific user
Wolf
The Three Musketeers
Bare Knuckles
Journey to the Center of the Earth
Treasure Island
Total Recall
The 39 Steps
The Debt
Spectre
20,000 Leagues Under the Sea
Chain of Command
King Solomon's Mines
Montana
The Black Hole
The Lady Vanishes
Caught
Steel
Alice in Wonderland
Ghost in the Shell
1.7
1.4
1.3
1.3
1.2
1.2
1.2
1.1
1.1
1.1
1.0
0.99
0.99
0.98
0.97
0.95
0.94
0.94
0.94
title
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
sim_scores

11. SOFTWARE AND HARDWARE REQUIREMENTS

The successful completion of this training program involved the utilization of various software and hardware resources. This section outlines the specific requirements used during the training process:

SOFTWARE REQUIREMENTS:

1. Integrated Development Environments (IDEs):

1. **Visual Studio Code (VS Code**):
   Version: Latest stable version recommended
   Purpose: VS Code served as the primary integrated development environment (IDE) for working with PYTHON. It provided essential features such as code editing, debugging, and extensions for PYTHON development.

HARDWARE REQUIREMENTS:

Computer:
Processor: A modern multi-core processor, such as Intel Core i5 or equivalent.
RAM: A minimum of 8 GB of RAM to support smooth operation of IDEs,t, and multitasking.
Storage: Adequate storage space for software installations, project files, and image data.
Graphics: A standard graphics card capable of running the chosen operating system, IDEs, and image processing software.

Internet Connection:
A stable and high-speed internet connection to ensure uninterrupted access to online research materials, tutorials, and software updates.

12. FUTURE OF AI

The future of AI is a landscape of boundless possibilities, where the fusion of human ingenuity and machine intelligence promises to reshape our world in profound ways. As AI technologies continue to advance, we can anticipate a future characterized by unprecedented levels of automation and efficiency across industries. Healthcare will see AI revolutionize diagnosis, treatment, and patient care, while autonomous vehicles will redefine transportation and mobility. Natural language processing will foster more intuitive interactions with AI-powered assistants, and AI ethics and regulations will ensure responsible development and deployment. Education will become increasingly personalized, and finance will rely on AI for everything from trading to fraud detection. Moreover, AI will play a pivotal role in addressing global challenges, from climate change mitigation to cybersecurity. Yet, as we step into this AI-driven future, we must navigate complex questions of ethics and governance to ensure that AI serves as a force for the betterment of society, fostering collaboration between humans and machines, and ultimately improving the human experience.

13. CONCLUSION

In conclusion, my internship experience in the field of Artificial Intelligence has been nothing short of enlightening and rewarding. Over the course of this internship, I had the opportunity to dive deep into the dynamic world of AI, working on diverse projects that challenged my problem-solving skills and expanded my knowledge base. From collaborating with the AI team to develop and fine-tune machine learning models to gaining practical experience in data analysis and natural language processing, this internship has equipped me with invaluable skills and insights that will undoubtedly shape my future career in AI. I am grateful for the mentorship and guidance I received throughout this journey, which not only helped me grow as a professional but also fostered a passion for the limitless possibilities that AI holds. As I move forward, I carry with me the experiences, lessons, and networks forged during this internship, and I am excited to contribute to the continued advancement of AI and its applications in addressing real-world challenges.

14. WEEKLY OVERVIEW OF TRAINING ACTIVITIES

1st WEEK	DATE	DAY	NAME OF THE TOPIC/MODULE COMPLETED
	07/08/23	Monday	Introduction to Artificial Intelligence
	08/08/23	Tuesday	Introduction to Chatbots
	09/08/23	Wednesday	Rule-Based Chatbot
	10/08/23	Thursday	Python Code for Chatbot
	11/08/23	Friday	Coding for Chatbot

2nd WEEK	DATE	DAY	NAME OF THE TOPIC/MODULE COMPLETED
	14/08/23	Monday	Introduction to Game Theory
	15/08/23	Tuesday	Minimax Algorithm
	16/08/23	Wednesday	Analysis of Algorithm
	17/08/23	Thursday	Coding for TIC-TAC-TOE
	18/08/23	Friday	Coding for TIC-TAC-TOE

3rd WEEK	DATE	DAY	NAME OF THE TOPIC/MODULE COMPLETED
	21/08/23	Monday	Introduction to Face Detection
	22/08/23	Tuesday	Analysis of Haarcascade model
	23/08/23	Wednesday	Coding for Face Detection System
	24/08/23	Thursday	Code Implementation
	25/08/23	Friday	Code Analysis

4th WEEK	DATE	DAY	NAME OF THE TOPIC/MODULE COMPLETED
	28/08/23	Monday	Introduction to Recommendation Systems
	29/08/23	Tuesday	Introduction to Content-Filtering Technique
	30/08/23	Wednesday	Data Collection for Movie Recommendation System
	31/08/23	Thursday	Coding for Movie Recommendation System
	01/09/23	Friday	Code Implementation for Movie Recommendation System

REFERENCES

1. ***www.geekforgeeks.com***
2. ***www.python.org***
3. ***www.realpython.com***
4. ***www.pythonweekly.com***
5. ***www.w3schools.com***
6. ***www.kaggle.com***
7. ***www.scipy.org***
8. ***www.towardsdatascience.com***
9. ***www.hackerrank.com***
10. ***www.pybites.com***