Prompt Engineering

Course Learning Objectives	• Develop a strong foundation in Generative AI concepts, historical context, and diverse generative models. • Acquire hands-on coding expertise in TensorFlow and PyTorch for model training, evaluation, debugging, and optimization. • Explore advanced generative models like conditional GANs, unsupervised learning, and novel architectures. • Apply generative AI techniques to diverse domains, including image manipulation, text-to-image synthesis, and creative applications. • Address ethical challenges, biases, and societal impacts, and gain proficiency in deploying generative models responsibly.
Course Outcomes	• Comprehend Generative AI fundamentals, historical evolution, and various model types for practical application. • Develop practical skills through coding exercises, model training, and optimization using TensorFlow and PyTorch.
	• Analyse the working principles of advanced generative models and address associated challenges. • Apply generative AI techniques effectively in real-world scenarios, demonstrating creativity and versatility. • Recognize and address ethical considerations, biases, and societal impacts, adopting responsible AI practices.

Course Duration: 45 Hours

Module Wise Course Content and Outcome			
Module Name	**Module Content**	**Learning Outcome**	**Duration (hrs)**
Module1: Introduction to Generative AI	Overview of Generative AI. Historical context and evolution. Types of generative models: GANs, VAEs, etc. Use cases and applications.	Participants will gain a foundational understanding of generative AI concepts and its historical context.	4
Module 2: Fundamentals of Generative Models	In-depth exploration of GANs and VAEs. Architectural components and working principles. Mathematical foundations and loss functions. Case studies illustrating successful applications.	Participants will comprehend the working principles and differences between GANs and VAEs.	6
Module3: Implementation with TensorFlow and PyTorch	Hands-on coding exercises with TensorFlow. Practical implementation using PyTorch. Model training and evaluation. Debugging and optimization techniques.	Participants will gain practical skills in implementing generative models using popular frameworks.	8

Module 4: Advanced Generative Models	Conditional GANs and their applications. Unsupervised learning and generative clustering. Novel architectures and advancements in generative models. Challenges and considerations.	Participants will explore advanced generative models, understand their applications, and be aware of challenges.	6
Module5: Application Domains	Image generation and manipulation. Text-to-image synthesis. Style transfer in images. Creative applications in art and design.	Participants will apply generative models to various domains, fostering creativity and innovation.	7
Module6: Performance Metrics and Evaluation	Metrics for evaluating generative model performance. Challenges in assessing generated content. Real-world evaluation strategies.	Participants will understand how to measure and evaluate the performance of generative models.	4
Module7: Ethical Considerations in Generative AI	Bias and fairness in generative models. Ethical challenges and societal impacts. Responsible AI practices.	Participants will grasp the ethical considerations and responsibilities associated with generative AI.	3
Module8: Deployment and Integration	Strategies for deploying generative models. Integration into existing systems. Deployment challenges and solutions.	Participants will learn how to deploy and integrate generative models into practical applications.	5

Module9: Project Work and Case Studies	Work on generative AI projects. Analysis of case studies. Peer review and feedback.	Participants will apply their knowledge to real-world projects and learn from case studies.	2
Module 10: Future Trends and Emerging Technologies	Current trends in generative AI. Emerging technologies and research directions.	Participants will stay informed about the latest trends and advancements in the field.	2

Course Duration: 45 Hours

Test Projects:

Use Case 1: Text Classification for Customer Support Tickets

Description:

In this use case, we aim to implement a text classification system for customer support tickets. The goal is to automatically categorize incoming support tickets into predefined categories, such as technical issues, billing inquiries, and general feedback. The system will enhance the efficiency of the customer support process by automating the ticket categorization process.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a labeled dataset of historical customer support tickets with predefined categories.

2. Data Preprocessing:

- Cleaning and preprocessing text data, handling stopwords, and tokenization.

3. Model Selection:

- Choosing a pre-trained text classification model from Hugging Face's Transformers library.

4. Fine-Tuning:

- Fine-tuning the selected model on the collected dataset to adapt it to the specific domain of customer support tickets.

5. Integration:

- Integrating the trained model into the company's ticketing system or customer support platform.

6. Inference:

- Using the model to automatically classify incoming support tickets into relevant categories in real-time.

Tasks:

1. Data Collection:

- Gather a labeled dataset of historical customer support tickets.

2. Data Preprocessing:

- Clean and preprocess text data.

- Handle stopwords and perform tokenization.

3. Model Selection:

- Choose a pre-trained text classification model from Hugging Face's Transformers library.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset.

5. Integration:

- Integrate the trained model into the company's ticketing system or customer support platform.

6. Inference:

- Use the model to automatically classify incoming support tickets in real-time.

Evaluation:

The evaluation will consist of the following components:

1. Accuracy Metrics:

- Measure the accuracy of the model in correctly classifying support tickets.

2. Live Evaluation:

- Conduct a live session where you explain your approach, the implemented features, and demonstrate the system's effectiveness.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and areas for improvement.

This project aims to showcase your proficiency in implementing text classification for real-world applications, specifically in the context of customer support tickets. The evaluation provides an opportunity to receive feedback and refine your skills in natural language processing and model integration.

Use Case 2: Spam Mail Filtering

Description:

This use case focuses on implementing a spam mail filtering system to automatically identify and filter out unwanted spam emails. The goal is to enhance the user experience by reducing the influx of irrelevant and potentially harmful emails. The system will leverage machine learning techniques to classify emails as either spam or legitimate based on their content.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a labeled dataset of emails, with clear indications of whether each email is spam or legitimate.

2. Data Preprocessing:

- Cleaning and preprocessing email text data.

- Handling issues like HTML tags, special characters, and email headers.

3. Model Selection:

- Choosing a suitable pre-trained text classification model, such as a Naive Bayes classifier or a machine learning model from Hugging Face's Transformers library.

4. Fine-Tuning:

- Fine-tuning the selected model on the collected dataset to adapt it to the characteristics of spam and legitimate emails.

5. Evaluation:

- Assessing the model's performance using metrics such as precision, recall, and F1 score to ensure effective spam detection.

Tasks:

1. Data Collection:

- Gather a labeled dataset of emails with clear indications of spam or legitimacy.

2. Data Preprocessing:

- Clean and preprocess email text data.

- Handle issues like HTML tags, special characters, and email headers.

3. Model Selection:

- Choose a pre-trained text classification model suitable for spam mail filtering.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to adapt it to spam and legitimate email patterns.

5. Evaluation:

- Assess the model's accuracy, precision, recall, and F1 score to measure its effectiveness in classifying emails.

Evaluation:

The evaluation will consist of the following components:

1. Accuracy Metrics:

- Measure the accuracy of the model in correctly classifying emails as spam or legitimate.

2. Live Evaluation:

- Conduct a live session where you explain your approach, the implemented features, and demonstrate the system's effectiveness.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and areas for improvement.

This project aims to showcase your ability to implement a practical spam mail filtering solution using machine learning techniques. The evaluation provides an opportunity to demonstrate the effectiveness of your model and receive valuable feedback for improvement.

Use Case 3: Toxicity Detection in Online Comments

Description:

This use case involves the implementation of a toxicity detection system for online comments. The objective is to automatically identify and flag comments containing offensive, harmful, or inappropriate content. The system will use natural language processing (NLP) techniques and machine learning models to categorize comments based on their toxicity levels, contributing to a safer and more respectful online environment.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a labeled dataset of online comments, indicating the toxicity levels of each comment.

2. Data Preprocessing:

- Cleaning and preprocessing text data, addressing challenges such as handling profanity, slang, and diverse linguistic expressions.

3. Model Selection:

- Choosing an appropriate pre-trained text classification model, such as a transformer model from Hugging Face's library.

4. Fine-Tuning:

- Fine-tuning the selected model on the collected dataset to enhance its ability to detect different forms of toxicity.

5. Evaluation:

- Assessing the model's performance using metrics like precision, recall, and F1 score to ensure accurate toxicity detection.

Tasks:

1. Data Collection:

- Collect a labeled dataset of online comments with annotations indicating toxicity levels.

2. Data Preprocessing:

- Clean and preprocess text data, addressing challenges associated with offensive language, slang, and varied linguistic expressions.

3. Model Selection:

- Choose a pre-trained text classification model suitable for toxicity detection.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to improve its performance in identifying toxic comments.

5. Evaluation:

- Evaluate the model's accuracy, precision, recall, and F1 score to measure its effectiveness in detecting toxicity.

Evaluation:

The evaluation will consist of the following components:

1. Accuracy Metrics:

- Measure the accuracy of the model in correctly classifying comments based on toxicity levels.

2. Live Evaluation:

- Conduct a live session where you explain your approach, showcase implemented features, and demonstrate the system's effectiveness.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and areas for improvement.

This project aims to demonstrate your ability to develop a practical solution for toxicity detection in online comments using natural language processing and machine learning. The evaluation provides an opportunity to showcase the effectiveness of your model and receive valuable feedback for further enhancement.

Use Case 4: Automated Customer Support Chatbot

Description:

This use case involves the development of an automated customer support chatbot to enhance user experience and streamline customer query resolution. The chatbot will use natural language processing (NLP) and machine learning techniques to understand and respond to customer queries, providing instant assistance and reducing the workload on human customer support agents.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a dataset of historical customer interactions, including queries and corresponding resolutions.

2. Intent Recognition:

- Implementing techniques for recognizing the intent behind customer queries, such as identifying whether the user is seeking information, troubleshooting, or requesting assistance.

3. Entity Recognition:

- Incorporating entity recognition to extract relevant information from customer queries, such as order numbers, product names, or account details.

4. Dialog Management:

- Building a dialog management system to maintain context and provide coherent responses throughout the conversation.

5. Model Selection:

- Choosing or building a suitable natural language understanding (NLU) model to comprehend customer queries accurately.

Tasks:

1. Data Collection:

- Collect a dataset of historical customer interactions, including a variety of queries and resolutions.

2. Intent Recognition:

- Implement techniques for recognizing the intent behind customer queries to understand the user's purpose.

3. Entity Recognition:

- Develop entity recognition capabilities to extract relevant information from customer queries.

4. Dialog Management:

- Build a dialog management system to maintain context and provide coherent responses throughout the conversation.

5. Model Selection:

- Choose or build a natural language understanding (NLU) model capable of accurately comprehending customer queries.

Evaluation:

The evaluation will consist of the following components:

1. Accuracy Metrics:

- Measure the accuracy of the chatbot in correctly recognizing intents and extracting entities from customer queries.

2. Live Evaluation:

- Conduct a live session where you demonstrate the chatbot's effectiveness in understanding and responding to user queries.

3. Feedback:

- Receive feedback on your chatbot's implementation, highlighting strengths and areas for improvement.

This project aims to showcase your ability to develop a functional and efficient automated customer support chatbot. The evaluation provides an opportunity to demonstrate the effectiveness of your chatbot and receive valuable feedback for further refinement.

Use Case 5: Content Organization for News Aggregators Using Text Clustering

Description:

This use case involves implementing a content organisation system for news aggregators using text clustering techniques. The goal is to automatically group similar news articles together, providing users with a more organised and streamlined reading experience. Text clustering will be employed to group articles based on their content similarity, helping users discover related news stories more efficiently.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Data Collection:

   - Gathering a diverse dataset of news articles from various sources.

2. Text Preprocessing:

   -Cleaning and preprocessing text data, handling tasks such as tokenization, stemming, and removing stop words.

3. Feature Extraction:

- Extracting relevant features from the news articles, such as TF-IDF (Term Frequency-Inverse Document Frequency) vectors.

4. Text Clustering:

- Implementing text clustering algorithms, such as K-means or hierarchical clustering, to group similar articles together.

5. Evaluation:

- Assessing the performance of the text clustering model, using metrics like silhouette score or coherence score.

Tasks:

1. Data Collection:

- Collect a diverse dataset of news articles from various sources, ensuring representation across different topics.

2. Text Preprocessing:

- Clean and preprocess the text data, including tasks like tokenization, stemming, and removing stop words.

3. Feature Extraction:

- Extract relevant features from the news articles, such as TF-IDF vectors, to represent the content numerically.

4. Text Clustering:

- Implement a text clustering algorithm, selecting an appropriate method for grouping similar articles.

5. Evaluation:

- Evaluate the effectiveness of the text clustering model using suitable metrics to measure the quality of article groupings.

Evaluation:

The evaluation will consist of the following components:

1. Cluster Quality Metrics:

- Measure the quality of text clusters using metrics such as silhouette score or coherence score.

2. Live Evaluation:

- Conduct a live session where you demonstrate the effectiveness of the content organization system for news aggregators.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and areas for improvement.

This project aims to showcase your ability to implement an efficient content organization system for news aggregators using text clustering techniques. The evaluation provides an opportunity to demonstrate the effectiveness of your clustering model and receive valuable feedback for further refinement.

Use Case 6: Meeting Transcript Summarization

Description:

This use case involves developing a meeting transcript summarization system to automatically generate concise summaries of meeting discussions. The objective is to enhance productivity by providing participants with a condensed version of key points, action items, and decisions made during the meeting. The system will employ natural language processing (NLP) techniques to analyze and summarize the content of meeting transcripts.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a dataset of meeting transcripts with associated summaries or key points.

-

2. Text Preprocessing:

- Cleaning and preprocessing meeting transcript data, addressing issues such as removing irrelevant information and handling variations in language.

3. Feature Extraction:

- Extracting relevant features from the meeting transcripts, such as identifying key sentences or important phrases.

4. Abstractive Summarization:

- Implementing abstractive summarization techniques to generate concise and coherent summaries.

5. Evaluation:

- Assessing the quality of generated summaries using metrics such as ROUGE (Recall-Oriented Understudy for Gisting Evaluation).

Tasks:

1. Data Collection:

   - Collect a dataset of meeting transcripts with associated summaries or key points.

2. Text Preprocessing:

- Clean and preprocess meeting transcript data, removing irrelevant information and addressing variations in language.

3. Feature Extraction:

- Extract relevant features from meeting transcripts, identifying key sentences or important phrases.

4. Abstractive Summarization:

- Implement abstractive summarization techniques to generate concise and coherent meeting summaries.

-

5. Evaluation:

- Evaluate the quality of generated summaries using metrics like ROUGE to measure the effectiveness of the summarization system.

Evaluation:

The evaluation will consist of the following components:

1. ROUGE Metrics:

- Measure the precision, recall, and F1 score using ROUGE metrics to evaluate the quality of generated summaries.

2. Live Evaluation:

- Conduct a live session where you demonstrate the effectiveness of the meeting transcript summarization system.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and areas for improvement.

This project aims to showcase your ability to implement an abstractive summarization system for meeting transcripts, providing a valuable tool for efficient information retrieval and decision-making. The evaluation provides an opportunity to demonstrate the effectiveness of your summarization model and receive valuable feedback for further refinement.

Use Case 7: Medical Report Summarization

Description:

This use case involves the development of a medical report summarization system to automatically generate concise summaries of complex medical reports. The objective is to assist healthcare professionals by providing succinct and relevant information from lengthy medical documents, enabling quicker decision-making and enhancing overall patient care. Natural language processing (NLP) techniques and transformer- based models will be employed for accurate summarization.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a dataset of diverse medical reports, including radiology reports, pathology reports, and clinical notes.

2. Text Preprocessing:

- Cleaning and preprocessing medical report data, addressing challenges specific to healthcare terminology and jargon.

3. Feature Extraction:

- Extracting relevant features from medical reports, such as key findings, diagnoses, and treatment plans.

4. Model Selection and Fine-Tuning:

- Choosing a transformer-based model from Hugging Face's Transformers library.

- Fine-tuning the selected model on the collected medical report dataset.

5. Abstractive Summarization:

- Implementing abstractive summarization techniques using the chosen transformer- based model to generate concise and accurate summaries.

6. Evaluation:

- Assessing the quality of generated summaries using domain-specific evaluation metrics, considering medical accuracy and relevance.

Tasks:

1. Data Collection:

- Collect a diverse dataset of medical reports, covering various medical specialties.

2. Text Preprocessing:

- Clean and preprocess medical report data, addressing challenges specific to healthcare terminology and jargon.

3. Feature Extraction:

- Extract relevant features from medical reports, such as key findings, diagnoses, and treatment plans.

4. Model Selection and Fine-Tuning:

- Choose a transformer-based model from Hugging Face's Transformers library.
- Fine-tune the selected model on the collected medical report dataset.

5. Abstractive Summarization:

- Implement abstractive summarization techniques using the chosen and fine-tuned transformer-based model.

6. Evaluation:

- Evaluate the quality of generated summaries using domain-specific metrics, ensuring medical accuracy and relevance.

Evaluation:

The evaluation will consist of the following components:

1. Domain-Specific Metrics:

- Measure the accuracy and relevance of generated medical report summaries using metrics tailored to the healthcare domain.

2. Live Evaluation:

- Conduct a live session where you demonstrate the effectiveness of the medical report summarization system.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and areas for improvement.

This project aims to showcase your ability to develop a practical and accurate medical report summarization system using transformer-based models. The evaluation provides an opportunity to demonstrate the effectiveness of your summarization model and receive valuable feedback for further refinement.

Use Case 8: Content Tagging for E-learning

Description:

This use case involves the implementation of a content tagging system for e-learning platforms, aiming to enhance content organization and facilitate personalized learning experiences. The system will utilize natural language processing (NLP) techniques to automatically tag educational content, such as articles, videos, and quizzes, with relevant topics, difficulty levels, and learning objectives. This tagging system will help learners discover content aligned with their preferences and proficiency levels.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a diverse dataset of educational content from various subjects and difficulty levels.

2. Text Preprocessing:

- Cleaning and preprocessing educational content, handling tasks such as tokenization and removing irrelevant information.

3. Topic Modeling:

- Implementing topic modeling techniques to identify key topics within educational content.

4. Difficulty Level Identification:

- Developing algorithms to assess the difficulty level of educational content based on factors such as vocabulary complexity and concept intricacy.

5. Tagging System Integration:

- Integrating the content tagging system into an e-learning platform, ensuring seamless tagging of new content.

Tasks:

1. Data Collection:

- Collect a diverse dataset of educational content, including articles, videos, and quizzes, from various subjects and difficulty levels.

2. Text Preprocessing:

- Clean and preprocess educational content, including tasks such as tokenization and removing irrelevant information.

3. Topic Modeling:

- Implement topic modeling techniques to identify key topics within the educational content.

4. Difficulty Level Identification:

- Develop algorithms to assess the difficulty level of educational content based on factors such as vocabulary complexity and concept intricacy.

5. Tagging System Integration:

Integrate the content tagging system into an e-learning platform, ensuring seamless tagging of new educational content.

Evaluation:

The evaluation will consist of the following components:

1. Tagging Accuracy:

- Measure the accuracy of the content tagging system in assigning relevant tags to educational content.

2. User Feedback:

- Gather feedback from users, including learners and educators, on the effectiveness and usability of the content tagging system.

3. Live Evaluation:

- Conduct a live session where you demonstrate the content tagging system in action, showcasing its impact on personalized learning experiences.

This project aims to showcase your ability to develop an effective content tagging system for e-learning platforms, enhancing content discoverability and supporting personalized learning journeys. The evaluation provides an opportunity to demonstrate the system's accuracy and gather valuable feedback for further refinement.

Use Case 9: Handwriting Recognition

Description:

Implementation of a handwriting recognition system to convert handwritten text into digital format, enhancing user interactions with various applications. The system employs machine learning techniques to accurately interpret and transcribe handwritten input.

Learning Outcome:

Engage in this project to gain experience and understanding in the following areas:

1. Dataset Collection:

- Gather a diverse dataset of handwritten samples, encompassing different styles, languages, and complexities.

2. Preprocessing and Data Augmentation:

- Implement preprocessing techniques to enhance the quality of handwritten samples for training.

- Apply data augmentation strategies to improve the model's adaptability to variations in writing styles and conditions.

3. Model Architecture:

- Develop a sophisticated deep learning model for handwriting recognition, balancing accuracy and efficiency.

- Explore and experiment with different architectures to optimize recognition performance.

4. Language Support:

- Expand language support to accommodate a wider user base, considering both widely spoken languages and niche linguistic requirements.

- Incorporate mechanisms for easy integration of additional languages based on user feedback.

5. User Interface Integration:

- Integrate the handwriting recognition model into user interfaces, enabling seamless interaction with applications.

- Continuously refine the user interface based on user experience feedback to enhance accessibility and user-friendliness.

Tasks:

1. Dataset Collection:

- Collect a diverse dataset of handwritten samples covering various styles, languages, and complexities.

2. Preprocessing and Data Augmentation:

- Implement preprocessing techniques to enhance the quality of handwritten samples.

- Apply data augmentation strategies to improve the model's adaptability.

3. Model Architecture:

- Develop a deep learning model for handwriting recognition, experimenting with different architectures.

4. Language Support:

- Expand language support to cater to a diverse user base.

5. User Interface Integration:

- Integrate the handwriting recognition model into user interfaces for seamless interaction.

Evaluation:

1. Recognition Accuracy:

- Measure the accuracy of the handwriting recognition system with diverse handwritten samples.

- Conduct regular evaluations to maintain and improve accuracy.

2. User Feedback:

- Gather feedback from users on recognition accuracy, speed, and overall user experience.

- Implement user-suggested enhancements and address issues to optimize system performance.

3. Real-world Applications:

- Showcase successful real-world applications, such as note-taking apps, form processing, and digital annotation tools.

- Explore partnerships and collaborations for integrating the technology into various industries.

Develop an efficient handwriting recognition system that accurately transcribes handwritten input into digital format, improving user interactions and enabling diverse real-world applications. The evaluation provides insights into accuracy, user feedback, and practical applications for further refinement and development.

Use Case 10: Sentiment Analysis for Product Reviews Description:

In this use case, the objective is to develop a sentiment analysis system for product reviews. The goal is to automatically determine the sentiment expressed

in reviews, categorizing them as positive, negative, or neutral. This system will aid businesses in understanding customer opinions, improving products, and addressing issues promptly.

Learning Outcome:

By engaging in this project, you will acquire expertise in the following areas:

1. Data Collection:

- Assemble a labeled dataset comprising product reviews with associated sentiment labels.

2. Data Preprocessing:

- Clean and preprocess text data, handling issues like punctuation, special characters, and stemming.

3. Model Selection:

- Choose a suitable pre-trained sentiment analysis model from Hugging Face's Transformers library.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to enhance its performance on product reviews.

5. Integration:

- Integrate the trained model into a platform where businesses can input reviews and receive sentiment analysis results.

6. Inference:

- Utilize the model to automatically analyze the sentiment of product reviews in real- time.

Tasks:

1. Data Collection:

- Curate a labeled dataset of product reviews covering a diverse range of products.

2. Data Preprocessing:

- Clean and preprocess text data, addressing issues specific to sentiment analysis.

3. Model Selection:

- Choose a pre-trained sentiment analysis model that aligns with the requirements of product reviews.

4. Fine-Tuning:

- Fine-tune the model on the collected dataset, ensuring it captures nuances in product sentiment.

5. Integration:

- Integrate the sentiment analysis model into a user-friendly platform for businesses to analyze product reviews.

6. Inference:

- Demonstrate the model's effectiveness by performing real-time sentiment analysis on a set of diverse product reviews.

Evaluation:

The evaluation will encompass the following aspects:

1. Accuracy Metrics:

- Assess the accuracy of the model in correctly predicting the sentiment of product reviews.

2. Live Evaluation:

- Present a live session explaining the methodology, implemented features, and showcase the system's effectiveness in real-time.

3. Feedback:

- Receive feedback on your implementation, emphasizing strengths and suggesting areas for improvement.

This project is designed to showcase your proficiency in sentiment analysis and your ability to implement it in a practical context, addressing the needs of businesses in understanding customer sentiments toward their products.

Use Case 11: Named Entity Recognition (NER) for Legal Documents

Description:

In this use case, the objective is to implement a Named Entity Recognition (NER) system tailored for legal documents. The goal is to automatically identify and classify entities such as names of individuals, organizations, dates, and legal terminology within legal texts. This system will streamline information extraction processes in the legal domain, enhancing efficiency and accuracy.

Learning Outcome:

By undertaking this project, you will gain expertise in the following areas:

1. Data Collection:

- Assemble a labeled dataset of legal documents with annotated entities.

2. Data Preprocessing:

- Clean and preprocess legal text data, addressing challenges specific to the legal domain.

3. Model Selection:

- Choose a pre-trained NER model from Hugging Face's Transformers library suitable for legal entity recognition.

4. Fine-Tuning:

- Fine-tune the selected model on the collected legal dataset to adapt it to the intricacies of legal language.

5. Integration:

- Integrate the trained NER model into a platform where legal professionals can extract and analyze entities from legal documents.

6. Inference:

- Utilize the model to automatically recognize and categorize named entities in real- time within legal documents.

Tasks:

1. Data Collection:

- Compile a labeled dataset of legal documents, ensuring diversity in legal contexts and document types.

2. Data Preprocessing:

- Clean and preprocess legal text data, addressing challenges such as legalese, complex sentence structures, and specific legal terminologies.

3. Model Selection:

- Choose a pre-trained NER model that aligns with the requirements of recognizing entities in legal documents.

4. Fine-Tuning:

- Fine-tune the model on the collected legal dataset, ensuring it captures the nuances of named entities in legal contexts.

5. Integration:

- Integrate the NER model into a user-friendly platform for legal professionals to extract entities from legal documents.

6. Inference:

- Demonstrate the model's effectiveness by performing real-time entity recognition on a variety of legal documents.

Evaluation:

The evaluation will involve assessing the following aspects:

1. Entity Recognition Accuracy:

- Measure the accuracy of the NER model in correctly identifying and categorizing named entities in legal documents.

2. Live Evaluation:

- Present a live session explaining the methodology, implemented features, and showcase the system's effectiveness in real-time legal document analysis.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and suggesting areas for improvement, especially in handling legal language intricacies.This project aims to showcase your expertise in implementing NER systems for specialized domains, with a focus on the legal field.

Use Case 12: Topic Modeling for Research Papers

Description:

In this use case, the goal is to develop a topic modeling system for academic research papers. The objective is to automatically identify and categorize the main topics present in a collection of research papers. This system will assist researchers, students, and institutions in efficiently navigating through vast amounts of academic literature and staying informed about current trends and subjects in their field.

Learning Outcome:

By working on this project, you will gain experience in the following key areas:

1. Data Collection:

- Assemble a dataset of research papers with relevant metadata, including titles, abstracts, and keywords.

2. Data Preprocessing:

- Clean and preprocess text data from research papers, handling challenges such as diverse writing styles and technical language.

3. Model Selection:

- Choose a pre-trained topic modeling model from Hugging Face's Transformers library or other relevant sources.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to adapt it to the specific characteristics of research papers.

5. Integration:

- Integrate the trained topic modeling model into a platform where users can input research papers and receive information about the main topics.

6. Inference:

- Use the model to automatically extract and present the main topics from research papers in real-time.

Tasks:

1. Data Collection:

- Compile a dataset of research papers, ensuring diversity in topics, authors, and publication sources.

2. Data Preprocessing:

- Clean and preprocess text data from research papers, considering challenges specific to academic writing.

3. Model Selection:

- Choose a pre-trained topic modeling model suitable for academic research paper analysis.

4. Fine-Tuning:

- Fine-tune the model on the collected dataset, ensuring it captures the nuances and diversity of topics in research papers.

5. Integration:

- Integrate the topic modeling model into a user-friendly platform, allowing users to explore topics within research papers easily.

6. Inference:

- Demonstrate the model's effectiveness by performing real-time topic extraction on a diverse set of research papers.

Evaluation:

The evaluation will include the following components:

1. Topic Extraction Accuracy:

- Measure the accuracy of the topic modeling model in correctly identifying and categorizing the main topics in research papers.

2. Live Evaluation:

- Present a live session explaining the methodology, implemented features, and showcase the system's effectiveness in real-time research paper analysis.

3. Feedback:

- Receive feedback on your implementation, highlighting strengths and suggesting areas for improvement, particularly in handling the nuances of academic writing.

This project aims to demonstrate your proficiency in implementing topic modeling systems for specialized domains, with a focus on academic research papers.

Use Case 13: Automated Code Review Assistant

Description:

In this use case, the aim is to create an automated code review assistant to assist software developers in the code review process. The system will analyze source code submissions, identify potential issues, and provide constructive feedback. This will help maintain coding standards, improve code quality, and streamline the code review workflow.

Learning Outcome:

By engaging in this project, you will gain experience in the following areas:

1. Data Collection:

- Assemble a dataset of code snippets or full source code files with associated review comments or annotations.

2. Data Preprocessing:

- Clean and preprocess code data, handling formatting variations and extracting relevant features.

3. Model Selection:

- Choose or design a code analysis model suitable for identifying common coding issues and best practices.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to adapt it to the specific coding standards and practices of the development team.

5. Integration:

- Integrate the trained code review model into popular version control systems or code collaboration platforms used by the development team.

6. Inference:

- Use the model to automatically analyze code submissions during the code review process and provide feedback to developers.

Tasks:

1. Data Collection:

- Gather a dataset of code snippets or full source code files, ensuring representation of various programming languages and common coding issues.

2. Data Preprocessing:

- Clean and preprocess code data, extracting relevant features and ensuring consistency in formatting.

3. Model Selection:

- Choose or design a code analysis model that aligns with the requirements of identifying coding issues and providing constructive feedback.

4. Fine-Tuning:

- Fine-tune the model on the collected dataset, considering the specific coding standards and practices of the development team.

5. Integration:

- Integrate the code review model into the development team's workflow, such as incorporating it into version control systems or code collaboration platforms.

6. Inference:

- Demonstrate the model's effectiveness by automatically reviewing and providing feedback on code submissions in real-time.

Evaluation:

The evaluation will include the following components:

1. Code Review Accuracy:

- Measure the accuracy of the code review model in correctly identifying coding issues and providing relevant feedback.

2. Live Evaluation:

- Present a live session explaining the methodology, implemented features, and showcase the system's effectiveness in real-time code review.

3. Feedback:

- Receive feedback on your implementation, emphasizing strengths and suggesting areas for improvement, especially in aligning with the team's coding standards.

This project aims to showcase your ability to implement practical solutions for code analysis and automate aspects of the code review process to enhance development workflows.

Use Case 14: Smart Home Automation System(Alexa)

Description:

In this use case, the objective is to design and implement a smart home automation system that leverages artificial intelligence for improved efficiency and user experience. The system will enable users to control various devices in their homes, such as lights, thermostats, and security cameras, using voice commands or automated routines. This project aims to showcase the integration of AI into everyday environments to enhance convenience and energy efficiency.

Learning Outcome:

By engaging in this project, you will gain experience and understanding in the following areas:

1. Device Integration:

- Integrate various smart home devices into a unified system, allowing for centralized control.

2. Natural Language Processing (NLP):

- Implement NLP capabilities to enable voice commands for controlling devices.

3. Automation Rules:

- Design and implement automation rules based on user preferences and routines.

4. Machine Learning for Predictive Automation:

- Use machine learning to predict user preferences and automate device control based on historical data.

5. Security Integration:

- Integrate security features to ensure the privacy and safety of smart home users.

6. User Interface:

- Develop a user-friendly interface, such as a mobile app or a web portal, for easy control and monitoring of the smart home system.

Tasks:

1. Device Integration:

- Integrate a variety of smart home devices, including lights, thermostats, cameras, and smart plugs, into the automation system.

2. Natural Language Processing (NLP):

- Implement NLP capabilities to recognize and process voice commands for controlling smart home devices.

3. Automation Rules:

- Design and implement automation rules that allow users to create custom routines for their smart home.

4. Machine Learning for Predictive Automation:

- Use machine learning algorithms to analyze user behavior and predict preferences for automated device control.

5. Security Integration:

- Implement security features, such as secure authentication and data encryption, to protect user privacy and prevent unauthorized access.

6. User Interface:

- Develop a user-friendly interface, either as a mobile app or a web portal, allowing users to easily control and monitor their smart home devices.

Evaluation:

The evaluation will include the following components:

1. User Experience:

- Assess the user experience of interacting with the smart home system, considering ease of use and overall satisfaction.

2. Automation Accuracy:

- Measure the accuracy of automation rules and predictive automation features in meeting user expectations.

3. Security Assessment:

- Evaluate the security measures implemented to safeguard user privacy and prevent unauthorized access.

4. Live Demonstration:

- Present a live demonstration showcasing voice commands, automation routines, and the overall functionality of the smart home system.

5. Feedback:

- Receive feedback on your implementation, focusing on user experience, automation accuracy, security measures, and potential areas for improvement. This project aims to demonstrate your ability to integrate AI into practical, everyday scenarios, emphasizing user-centric design, automation accuracy, and security considerations in the context of a smart home environment.

Use Case 15: Image Generation from Text

Description:

In this use case, the objective is to create a system that generates images based on textual descriptions. This can be applied in various domains, such as generating scene illustrations from written prompts, conceptualizing design ideas, or even assisting in creative content creation. The system will take a textual input and generate corresponding images, demonstrating the capabilities of text-to-image synthesis.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Collect a dataset of paired examples consisting of textual descriptions and corresponding images.

2. Data Preprocessing:

- Preprocess textual descriptions and images to create a compatible dataset for training.

3. Model Selection:

- Choose a suitable pre-trained image generation model, possibly leveraging architectures like DALL-E or similar frameworks.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to align it with the specific requirements of generating images from text.

5. Integration:

- Develop an interface or application where users can input text, and the system generates corresponding images.

6. Inference:

- Demonstrate the system's ability to generate images in real-time based on user- provided textual descriptions.

Tasks:

1. Data Collection:

- Assemble a dataset containing textual descriptions paired with corresponding images.

2. Data Preprocessing:

- Implement preprocessing steps to make the textual and image data compatible for model training.

3. Model Selection:

- Choose a pre-trained image generation model suitable for the text-to-image synthesis task.

4. Fine-Tuning:

- Fine-tune the selected model using the collected dataset to enhance its performance in generating images from textual prompts.

5. Integration:

- Develop an application or platform where users can interactively input text and receive generated images.

6. Inference:

- Showcase the system's real-time image generation capabilities based on user- provided textual descriptions.

Evaluation:

The evaluation will include:

1. Image Quality Metrics:

- Assess the quality of generated images based on predefined criteria.

2. Live Evaluation:

- Conduct a live session to demonstrate the system's real-time image generation capabilities, explaining your approach and the features implemented.

3. Feedback:

- Gather feedback on the system's performance, strengths, and areas for improvement during and after the live evaluation session.

Use Case 16: Speech-to-Text Transcription for Meeting Minutes

Description:

In this use case, the goal is to implement a Speech-to-Text (STT) transcription system for converting spoken language during meetings into written text. This system aims to streamline the process of creating meeting minutes, making it more efficient and accurate. By leveraging automatic transcription, businesses can save time and resources, ensuring comprehensive and precise documentation of discussions and decisions.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gather a diverse dataset of audio recordings from various meetings, ideally with different speakers and accents.

2. Data Preprocessing:

- Preprocess audio data, handle noise, and ensure compatibility for training an STT model.

3. Model Selection:

- Choose a pre-trained Speech-to-Text model, such as those available in popular libraries like Google's Speech Recognition API or Mozilla DeepSpeech.

4. Fine-Tuning:

- Fine-tune the selected model on the collected dataset to adapt it to the specific characteristics of meeting recordings.

5. Integration:

- Develop an application or integrate the STT model into existing meeting platforms to provide real-time transcription.

6. Inference:

- Demonstrate the system's ability to transcribe spoken words into written text during live meetings.

Tasks:

1. Data Collection:

- Assemble a dataset containing audio recordings of diverse meetings with different speakers.

2. Data Preprocessing:

- Implement preprocessing steps to enhance the quality of audio data and prepare it for training.

3. Model Selection:

- Choose a pre-trained STT model that aligns with the goals of accurate transcription for meeting minutes.

4. Fine-Tuning:

- Fine-tune the selected model on the collected meeting dataset to improve transcription accuracy.

5. Integration:

- Develop an application or integrate the STT model into meeting platforms for seamless transcription during live sessions.

6. Inference:

- Showcase the system's real-time transcription capabilities during live meetings, converting spoken words into written text.

Evaluation:

The evaluation will include:

1. Transcription Accuracy Metrics:

- Measure the accuracy of the system in transcribing meeting recordings compared to human-generated transcripts.

2. Real-time Performance:

- Evaluate the system's ability to provide timely transcriptions during live meetings.

3. Live Demonstration:

- Conduct a live session to demonstrate the STT system in action, explaining the approach and features implemented.

4. Feedback:

- Gather feedback on the system's performance, accuracy, and user experience during and after the live demonstration.

Use Case 17: Product Image Classification using Teachable Machine

Description:

This use case involves implementing a product image classification system using Teachable Machine, a platform that allows users to easily train machine learning models without extensive coding knowledge. The system will be designed to classify products based on images, providing a practical solution for inventory management, e- commerce platforms, or retail applications.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Collect a dataset of product images, ensuring diversity in products and variations in image conditions.

2. Data Labeling:

- Label the collected images with corresponding product categories, creating a labeled dataset for model training.

3. Teachable Machine Setup:

- Set up a project on Teachable Machine, upload the labeled dataset, and train a custom image classification model.

4. Model Evaluation:

- Evaluate the performance of the trained model using accuracy metrics provided by Teachable Machine.

5. Integration:

- Integrate the trained Teachable Machine model into an application or platform where users can upload product images for classification.

6. Inference:

- Demonstrate the system's ability to classify product images in real-time using the trained model.

Tasks:

1. Data Collection:

- Assemble a dataset containing diverse product images covering different categories.

2. Data Labeling:

- Label the product images with corresponding categories to create a labeled dataset.

3. Teachable Machine Setup:

- Create a project on Teachable Machine, upload the labeled dataset, and train a custom image classification model.

4. Model Evaluation:

- Evaluate the accuracy and performance of the trained model using Teachable Machine's metrics.

5. Integration:

- Integrate the trained Teachable Machine model into a user-friendly application or platform for product image classification.

6. Inference:

- Showcase the system's real-time product image classification capabilities using the trained Teachable Machine model.

Evaluation:

The evaluation will include:

1. Model Accuracy:

- Assess the accuracy of the Teachable Machine model in correctly classifying product images.

2. User Interface:

- Evaluate the user interface and experience of the application/platform for product image classification.

3. Real-time Performance:

- Evaluate the system's speed and efficiency in processing and classifying product images in real-time.

4. Live Demonstration:

- Conduct a live session to demonstrate the product image classification system's effectiveness, explaining the approach and features implemented.

5. Feedback:

- Gather feedback on the system's performance, accuracy, and user experience during and after the live demonstration.

Use Case 18: Tumor Detection in Medical Imaging

Description:

This use case involves the development of a tumor detection system using medical imaging data. The system will utilize machine learning algorithms to analyze medical images, such as MRIs or CT scans, to identify and classify tumors. Early detection of tumors is crucial for timely medical intervention and treatment planning.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gather a diverse dataset of medical images containing both tumor and non-tumor cases.

2. Data Preprocessing:

- Preprocess medical images, handle variations in resolution, and normalize pixel values for effective model training.

3. Model Selection:

- Choose or develop a suitable deep learning model for tumor detection in medical images, considering architectures like Convolutional Neural Networks (CNNs).

4. Fine-Tuning:

- Fine-tune the selected model on the collected medical image dataset to improve its performance in tumor detection.

5. Integration:

- Integrate the trained tumor detection model into existing medical imaging systems for real-time analysis.

6. Inference:

- Demonstrate the system's ability to detect and classify tumors in medical images in real-time.

Tasks:

1. Data Collection:

- Assemble a dataset containing medical images with labeled tumor and non-tumor cases.

2. Data Preprocessing:

- Implement preprocessing steps to ensure the medical image data is clean, normalized, and ready for model training.

3. Model Selection:

- Choose or develop a deep learning model suitable for tumour detection in medical images.

4. Fine-Tuning:

- Fine-tune the chosen model on the collected medical image dataset to enhance its ability to detect tumors.

5. Integration:

- Integrate the trained tumor detection model into existing medical imaging systems for seamless analysis.

6. Inference:

- Showcase the system's real-time tumor detection capabilities, demonstrating its accuracy in identifying and classifying tumors.

Evaluation:

The evaluation will include:

1. Detection Accuracy:

- Measure the accuracy of the system in correctly detecting and classifying tumors in medical images.

2. False Positive Rate:

- Assess the system's ability to minimize false positives, ensuring accurate tumor detection.

3. Real-time Performance:

- Evaluate the system's speed and efficiency in processing and analysing medical images in real-time.

4. Live Demonstration:

- Conduct a live session to demonstrate the tumour detection system's effectiveness, explaining the approach and features implemented.

5. Feedback:

- Gather feedback on the system's performance, accuracy, and practicality from healthcare professionals and stakeholders during and after the live demonstration.

Use Case 19: Facial Recognition for Access Control

Description:

This use case involves implementing facial recognition technology for access control in secure environments. The system will use facial recognition algorithms to identify individuals and grant or deny access based on their facial features. This technology is applicable in various scenarios such as building entrances, secure facilities, and attendance tracking.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gather a diverse dataset of facial images with labeled identities to train the facial recognition model.

2. Data Preprocessing:

- Preprocess facial images, handle variations in lighting, resolution, and pose for effective model training.

3. Model Selection:

- Choose a suitable facial recognition model, considering factors like accuracy and real-time processing capabilities.

4. Fine-Tuning:

- Fine-tune the selected model on the collected facial image dataset to enhance its performance in recognizing individuals.

5. Integration:

- Integrate the trained facial recognition model into access control systems for real- time identification.

6. Inference:

- Demonstrate the system's ability to recognize individuals in real-time and grant or deny access accordingly.

Tasks:

1. Data Collection:

- Assemble a dataset containing facial images with labeled identities, ensuring diversity in individuals and conditions.

2. Data Preprocessing:

- Implement preprocessing steps to ensure the facial image data is clean, aligned, and ready for model training.

3. Model Selection:

- Choose a pre-trained facial recognition model suitable for access control, balancing accuracy and real-time processing requirements.

4. Fine-Tuning:

- Fine-tune the chosen model on the collected facial image dataset to improve its accuracy in recognizing individuals.

5. Integration:

- Integrate the trained facial recognition model into existing access control systems for seamless real-time identification.

6. Inference:

- Showcase the system's real-time facial recognition capabilities, demonstrating its accuracy in identifying and granting/denying access.

Evaluation:

The evaluation will include:

1. Recognition Accuracy:

- Measure the accuracy of the system in correctly recognizing individuals based on facial features.

2. False Positive Rate:

- Assess the system's ability to minimize false positives, ensuring accurate access control.

3. Real-time Performance:

- Evaluate the system's speed and efficiency in processing and recognizing faces in real-time.

4. Live Demonstration:

- Conduct a live session to demonstrate the facial recognition system's effectiveness, explaining the approach and features implemented.

5. Feedback:

- Gather feedback on the system's performance, accuracy, and practicality from stakeholders during and after the live demonstration.

Use Case 20: Visual Question Answering (VQA)

Description:

In this use case, the goal is to develop a Visual Question Answering (VQA) system, which combines computer vision and natural language processing to answer questions related to images. This technology finds applications in various domains, including accessibility for visually impaired individuals, interactive robotics, and content retrieval.

Learning Outcome:

By working on this project, you will gain experience and understanding in the following areas:

1. Data Collection:

- Gathering a dataset with pairs of images and corresponding questions along with their answers.

2. Data Preprocessing:

- Preprocessing image data, including resizing and normalization.

- Tokenizing and encoding textual questions.

3. Model Architecture:

- Selecting or designing a model that can handle both image and text input for question answering.

4. Transfer Learning:

- Utilizing pre-trained models for image recognition and natural language processing.

5. Integration:

- Integrating the VQA model into a user interface or application.

6. Evaluation:

- Assessing the accuracy of the model in answering a diverse set of questions related to images.

Tasks:

1. Data Collection:

- Gather a dataset containing images, questions, and corresponding answers.

2. Data Preprocessing:

- Preprocess image data, ensuring uniformity in size and normalization.

- Tokenize and encode textual questions for model input.

3. Model Architecture:

- Choose or design a model architecture capable of handling both image and text inputs for question answering.

4. Transfer Learning:

- Leverage pre-trained models for image recognition and natural language understanding.

5. Integration:

- Build a user interface or application to showcase the VQA system.

6. Evaluation:

- Assess the model's performance by evaluating its accuracy in answering questions related to provided images.

Evaluation:

The evaluation will consist of the following components:

1. Accuracy Metrics:

- Measure the accuracy of the VQA model in providing correct answers.

2. User Experience Testing:

- Conduct user testing to evaluate the system's usability and user satisfaction.

3. Feedback:

- Gather feedback on the implementation, addressing both strengths and potential areas for improvement.

This project will demonstrate your proficiency in combining computer vision and natural language processing for a practical application. The evaluation will provide valuable insights into the effectiveness and user experience of the developed VQA system.

Student Assessment Plan:

Each of the above-mentioned test projects will be divided into tasks by the training partner for each specific institution. Such tasks will be jointly evaluated by the faculty and the training partner and the following weightage is to be followed.

- 70% weightage to the external practical assessment.
- 30% weightage to the internal assessment.

Final Test Project/External Assessment Plan:

The Final Test Project will be chosen from the list given above, jointly by the college faculty and the Training Partner. The Final Test Project will be assessed on the following tasks, for 70%.

Details	**Marks**
Task: 1	20
Task: 2	20
Task: 3	20
Task: 4	20
Task: 5	20