**PES Institute of Technology and Management**

# Department of Computer Science & Engineering (Data Science)

Laboratory Manual

****

**Semester: VI**

**Subject: Generative AI**

**Subject Code: BAIL657C**

**Compiled By:**

**Dr. Sunitha B S**

**Head Of the Department**

**NH-206, Sagar Road, Shivamogga-577204 Ph: 08182-640733/640734 Fax: 08182-233797**

[**www.pestrust.edu.in/pesitm**](http://www.pestrust.edu.in/pesitm)

**PROGRAM OUTCOMES**

|  |  |
| --- | --- |
| **PO's** | **PO Description** |
| **PO1** | **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2** | **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO3** | **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural,  societal, and environmental considerations. |
| **PO4** | **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| **PO5** | **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and  modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. |
| **PO6** | **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| **PO7** | **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| **PO8** | **Ethics:** Apply ethical principles and commit to professional ethics and  responsibilities and norms of the engineering practice. |
| **PO9** | **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| **PO10** | **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to  comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| **PO11** | **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| **PO12** | **Life-long learning:** Recognize the need for, and have the preparation and ability to  engage in independent and life-long learning in the broadest context of technological change. |

**PROGRAM SPECIFIC OUTCOMES**

|  |  |
| --- | --- |
| **PSO's** | **PSO Description** |
| **PSO1** | An ability to design and analyze algorithms by applying theoretical concepts to build complex  and computer- based systems in the domain of System Software, Computer Networks & Security, Web technologies, Data Science and Analytics. |
| **PSO2** | Be able to develop various software solutions by applying the techniques of Data Base Management, Complex Mathematical Models, Software Engineering practices and Machine Learning with Artificial Intelligence. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Generative** **AI** | | | Semester | 6 |
| Course Code | | **BAIL657C** | CIE Marks | 50 |
| Teaching Hours/Week (L:T:P: S) | | 0:0:1:0 | SEE Marks | 50 |
| Credits | | 01 | Exam Hours | 100 |
| Examination type (SEE) | | Practical | | |
| **Course** **objectives:**   * Understand the principles and concepts behind generative AI models * Explain the knowledge gained to implement generative models using Prompt design frameworks. * Apply various Generative AI applications for increasing productivity. * Develop Large Language Model-based Apps. | | | | |
| **Sl.NO** | **Experiments** | | | |
| 1. | Explore pre-trained word vectors. Explore word relationships using vector arithmetic. Perform arithmetic operations and analyze results. | | | |
| 2. | Use dimensionality reduction (e.g., PCA or t-SNE) to visualize word embeddings for Q 1. Select 10 words from a specific domain (e.g., sports, technology) and visualize their embeddings. Analyze clusters and relationships. Generate contextually rich outputs using embeddings. Write a program to generate 5 semantically similar words for a given input. | | | |
| 3. | Train a custom Word2Vec model on a small dataset. Train embeddings on a domain-specific corpus (e.g., legal, medical) and analyze how embeddings capture domain-specific semantics. | | | |
| 4. | Use word embeddings to improve prompts for Generative AI model. Retrieve similar words using word embeddings. Use the similar words to enrich a GenAI prompt. Use the AI model to generate responses for the original and enriched prompts. Compare the outputs in terms of detail and relevance. | | | |
| 5. | Use word embeddings to create meaningful sentences for creative tasks. Retrieve similar words for a seed word. Create a sentence or story using these words as a starting point. Write a program that: Takes a seed word. Generates similar words. Constructs a short paragraph using these words. | | | |
| 6. | Use a pre-trained Hugging Face model to analyze sentiment in text. Assume a real-world application, Load the sentiment analysis pipeline. Analyze the sentiment by giving sentences to input. | | | |
| 7. | Summarize long texts using a pre-trained summarization model using Hugging face model. Load the summarization pipeline. Take a passage as input and obtain the summarized text. | | | |
| 8. | Install langchain, cohere (for key), langchain-community. Get the api key( By logging into Cohere and obtaining the cohere key). Load a text document from your google drive . Create a prompt template to display the output in a particular manner. | | | |
| 9. | Take the Institution name as input. Use Pydantic to define the schema for the desired output and create a custom output parser. Invoke the Chain and Fetch Results. Extract the below Institution related details from Wikipedia: **The** **founder** **of** **the** **Institution.** **When** **it** **was** **founded.** **The** **current** **branches** **in** **the** **institution** **.** **How** **many** **employees** **are** **working** **in** **it.** **A** **brief** **4-line** **summary** **of** **the** **institution.** | | | |
| 10 | Build a chatbot for the Indian Penal Code. We'll start by downloading the official Indian Penal Code document, and then we'll create a chatbot that can interact with it. Users will be able to ask questions about the Indian Penal Code and have a conversation with it. | | | |

|  |
| --- |
| **Course** **outcomes** **(Course** **Skill** **Set):**  At the end of the course the student will be able to:   * Develop the ability to explore and analyze word embeddings, perform vector arithmetic to investigate word relationships, visualize embeddings using dimensionality reduction techniques * Apply prompt engineering skills to real-world scenarios, such as information retrieval, text generation. * Utilize pre-trained Hugging Face models for real-world applications, including sentiment analysis and text summarization. * Apply different architectures used in large language models, such as transformers, and understand their   advantages and limitations. |
| **Assessment** **Details** **(both** **CIE** **and** **SEE)**  The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together  **Continuous** **Internal** **Evaluation** **(CIE):**  CIE marks for the practical course are **50** **Marks**.  The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.   * Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session. * Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks. * Total marks scored by the students are scaled down to **30** **marks** (60% of maximum marks). * Weightage to be given for neatness and submission of record/write-up on time. * Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus. * In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce. * The suitable rubrics can be designed to evaluate each student’s performance and learning ability. * The marks scored shall be scaled down to **20** **marks** (40% of the maximum marks).   The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student. |
| **Semester** **End** **Evaluation** **(SEE):**   * SEE marks for the practical course are 50 Marks. * SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the Head of the Institute. |

|  |
| --- |
| * The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University. * All laboratory experiments are to be included for practical examination. * (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners. * Students can pick one question (experiment) from the questions lot prepared by the examiners jointly. * Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.   General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)  Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.  The minimum duration of SEE is 02 hours |
| **Suggested** **Learning** **Resources:**  **Books:**   1. Modern Generative AI with ChatGPT and OpenAI Models: Leverage the Capabilities of OpenAI's LLM for Productivity and Innovation with GPT3 and GPT4, by Valentina Alto, Packt Publishing Ltd, 2023. 2. Generative AI for Cloud Solutions: Architect modern AI LLMs in secure, scalable, and ethical cloud environments, by Paul Singh, Anurag Karuparti ,Packt Publishing Ltd, 2024.   **Web** **links** **and** **Video** **Lectures** **(e-Resources):**   * + https://[www.w3schools.com/gen\_ai/index.php](http://www.w3schools.com/gen_ai/index.php)   + https://youtu.be/eTPiL3DF27U   + https://youtu.be/je6AlVeGOV0   + https://youtu.be/RLVqsA8ns6k   + https://youtu.be/0SAKM7wiC-A   + https://youtu.be/28\_9xMyrdjg   + https://youtu.be/8iuiz-c-EBw   + https://youtu.be/7oQ8VtEKcgE   + https://youtu.be/seXp0VWWZV0 |

**1. Explore pre-trained word vectors. Explore word relationships using vector arithmetic. Perform arithmetic operations and analyze results.**

**Python Code:**

from gensim.downloader import load

# Load the pre-trained GloVe model (50 dimensions)

print("Loading pre-trained GloVe model (50 dimensions)...")

model = load("glove-wiki-gigaword-50")

# Function to perform vector arithmetic and analyze relationships

def ewr():

result = model.most\_similar(positive=['king', 'woman'], negative=['man'], topn=1)

print("\nking - man + woman = ?", result[0][0])

print("similarity:", result[0][1])

result = model.most\_similar(positive=['paris', 'italy'], negative=['france'], topn=1)

print("\nparis - france + italy = ?", result[0][0])

print("similarity:", result[0][1])

# Example 3: Find analogies for programming

result = model.most\_similar(positive=['programming'], topn=5)

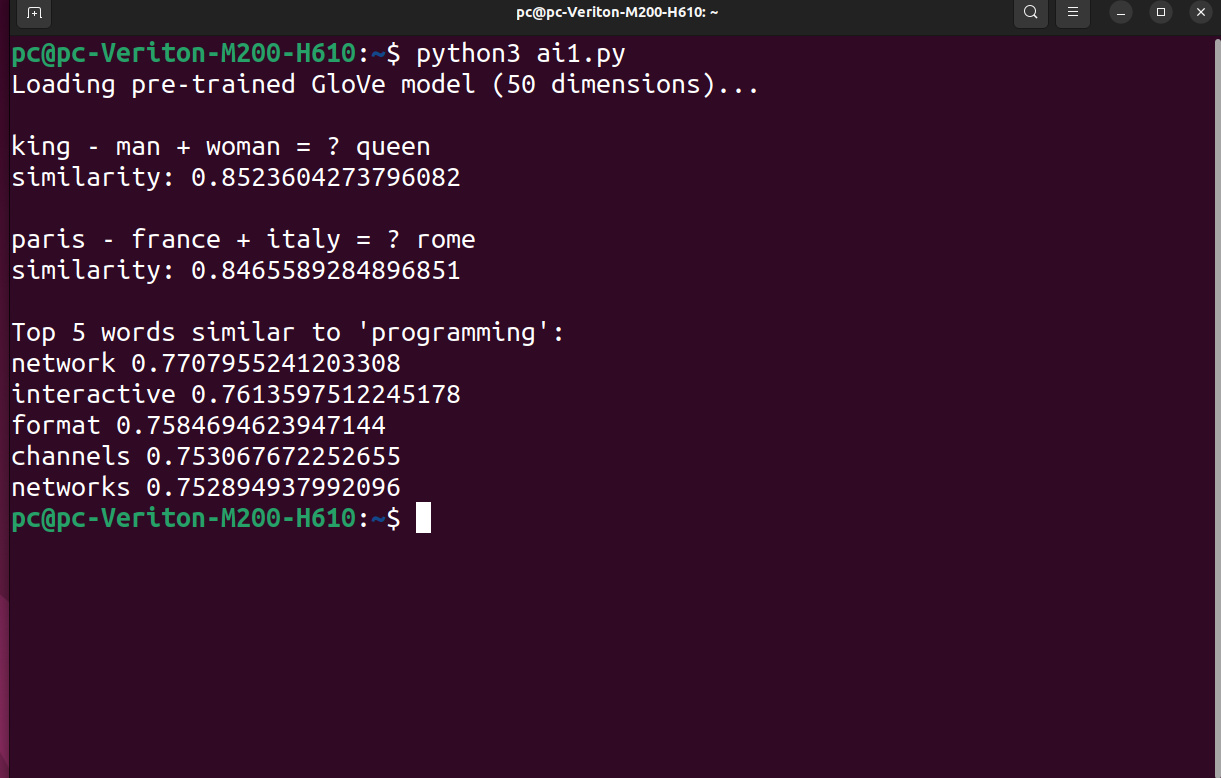
print("\nTop 5 words similar to 'programming':")

for word, similarity in result:

print(word, similarity)

ewr()

**Output :**



**2. Use dimensionality reduction (e.g., PCA or t-SNE) to visualize word embeddings for Q 1. Select 10 words from a specific domain (e.g., sports, technology) and visualize their embeddings. Analyze clusters and relationships. Generate contextually rich outputs using embeddings. Write a program to generate 5 semantically similar words for a given input.**

**Python Code:**

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from gensim.downloader import load

# Dimensionality reduction using PCA

def rd(ems):

pca = PCA(n\_components=2)

r = pca.fit\_transform(ems)

return r

# Visualize word embeddings

def visualize(words, ems):

plt.figure(figsize=(10, 6))

for i, word in enumerate(words):

x, y = ems[i]

plt.scatter(x, y, marker='o', color='blue')

plt.text(x + 0.02, y + 0.02, word, fontsize=12)

plt.show()

# Generate semantically similar words

def gsm(word):

sw = model.most\_similar(word, topn=5)

for word, s in sw:

print(word, s)

# Load pre-trained GloVe model from Gensim API

print("Loading pre-trained GloVe model (50 dimensions)...")

model = load("glove-wiki-gigaword-50")

words = ['football', 'basketball', 'soccer', 'tennis', 'cricket']

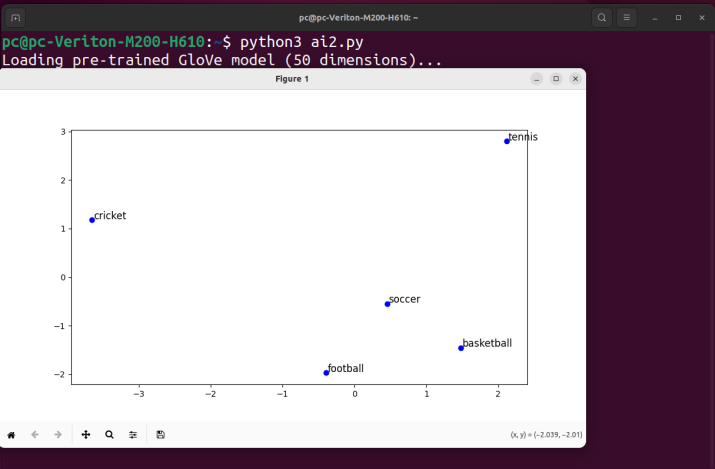
ems = [model[word] for word in words]

e = rd(ems)

visualize(words, e)

gsm("programming")

**Output:**



**3. Train a custom Word2Vec model on a small dataset. Train embeddings on a domain-specific corpus (e.g., legal, medical) and analyze how embeddings capture domain-specific semantics.**

**Python Code:**

from gensim.models import Word2Vec

# Custom Word2Vec model

def cw(corpus):

model = Word2Vec(

sentences=corpus,

vector\_size=50, # Dimensionality of word vectors

window=5, # Context window size

min\_count=1, # Minimum frequency for a word to be considered

workers=4, # Number of worker threads

epochs=10, # Number of training epochs

)

return model

# Analyze trained embeddings

def anal(model, word):

sw = model.wv.most\_similar(word, topn=5)

for w, s in sw:

print(w, s)

# Example domain-specific dataset (medical/legal/etc.)

corpus = [

"The patient was prescribed antibiotics to treat the infection.".split(),

"The court ruled in favor of the defendant after reviewing the evidence.".split(),

"Diagnosis of diabetes mellitus requires specific blood tests.".split(),

"The legal contract must be signed in the presence of a witness.".split(),

"Symptoms of the disease include fever, cough, and fatigue.".split(),

]

model = cw(corpus)

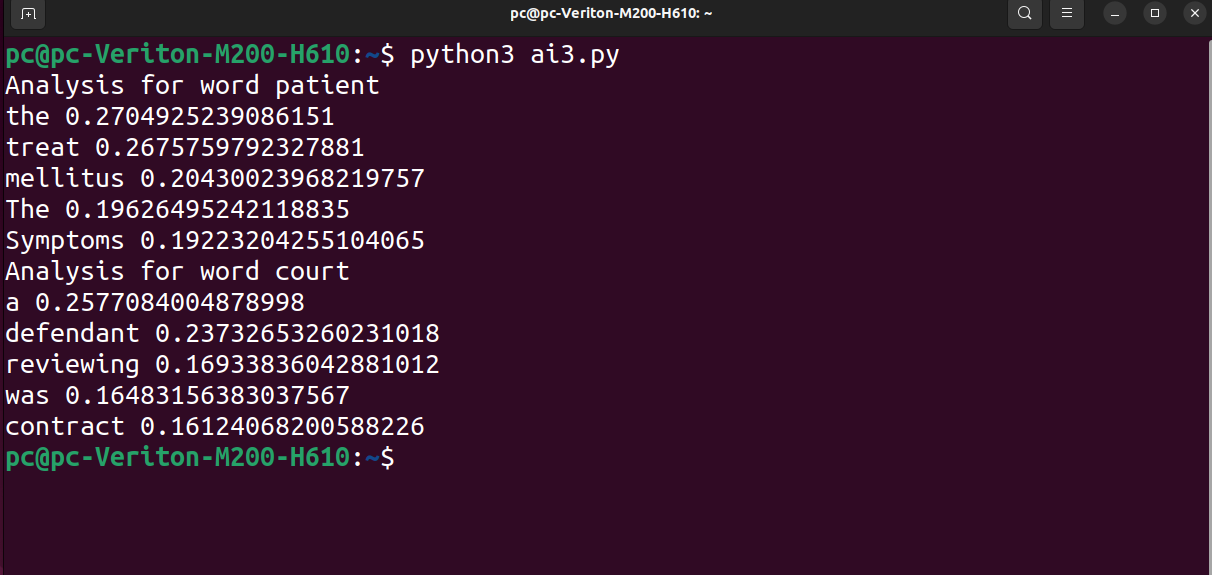
print("Analysis for word patient")

anal(model, "patient")

print("Analysis for word court")

anal(model, "court")

**Output:**



**4. Use word embeddings to improve prompts for Generative AI model. Retrieve similar words using word embeddings. Use the similar words to enrich a GenAI prompt. Use the AI model to generate responses for the original and enriched prompts. Compare the outputs in terms of detail and relevance.**

**Python Code:**

from gensim.downloader import load

import torch

from transformers import pipeline

# Load pre-trained word embeddings (GloVe)

model = load("glove-wiki-gigaword-50")

torch.manual\_seed(42)

# Define contextually relevant word enrichment

def enrich(prompt):

ep = ""# Start with the original prompt

words = prompt.split() # Split the prompt into words

for word in words:

sw = model.most\_similar(word, topn=3)

print("Test Data\n",sw)

enw=[]

for s,w in sw:

enw.append(s)

ep+=" " + " ".join(enw)

return ep

# Example prompt to be enriched

op = "lung cancer"

ep = enrich(op)

# Display the results

print("Original Prompt:", op)

print("Enriched Prompt:", ep)

generator = pipeline("text-generation", model="gpt2")#, tokenizer="gpt2")

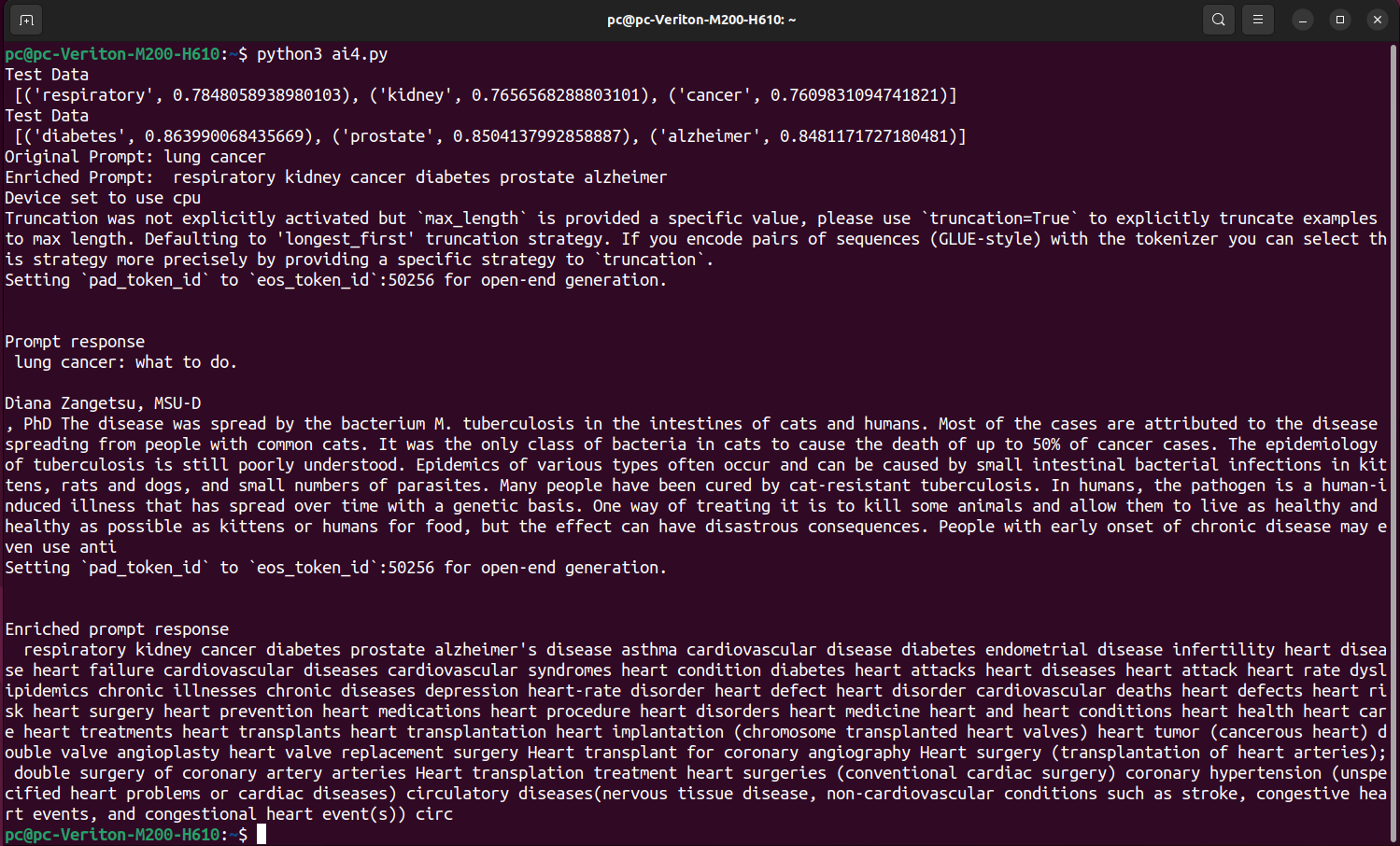
response = generator(op, max\_length=200, num\_return\_sequences=1, no\_repeat\_ngram\_size=2)#top\_p=0.95, temperature=0.7)

print("\n\nPrompt response\n",response[0]["generated\_text"])

response = generator(ep, max\_length=200, num\_return\_sequences=1, no\_repeat\_ngram\_size=2) #top\_p=0.95, temperature=0.7)

print("\n\nEnriched prompt response\n",response[0]["generated\_text"])

**Output :**



**5. Use word embeddings to create meaningful sentences for creative tasks. Retrieve similar words for a seed word. Create a sentence or story using these words as a starting point. Write a program that: Takes a seed word. Generates similar words. Constructs a short paragraph using these words.**

**Python Code:**

from gensim.downloader import load

import random

# Load the pre-trained GloVe model

print("Loading pre-trained GloVe model (50 dimensions)...")

model = load("glove-wiki-gigaword-50")

print(model)

print("Model loaded successfully!")

# Function to construct a meaningful paragraph

def create\_paragraph(iw, sws):

paragraph = f"The topic of {iw} is fascinating, often linked to terms like\n"

random.shuffle(sws) # Shuffle to add variety

for word in sws:

paragraph += str(word) + ", "

paragraph = paragraph.rstrip(", ") + "."

return paragraph

iw = "cricket"

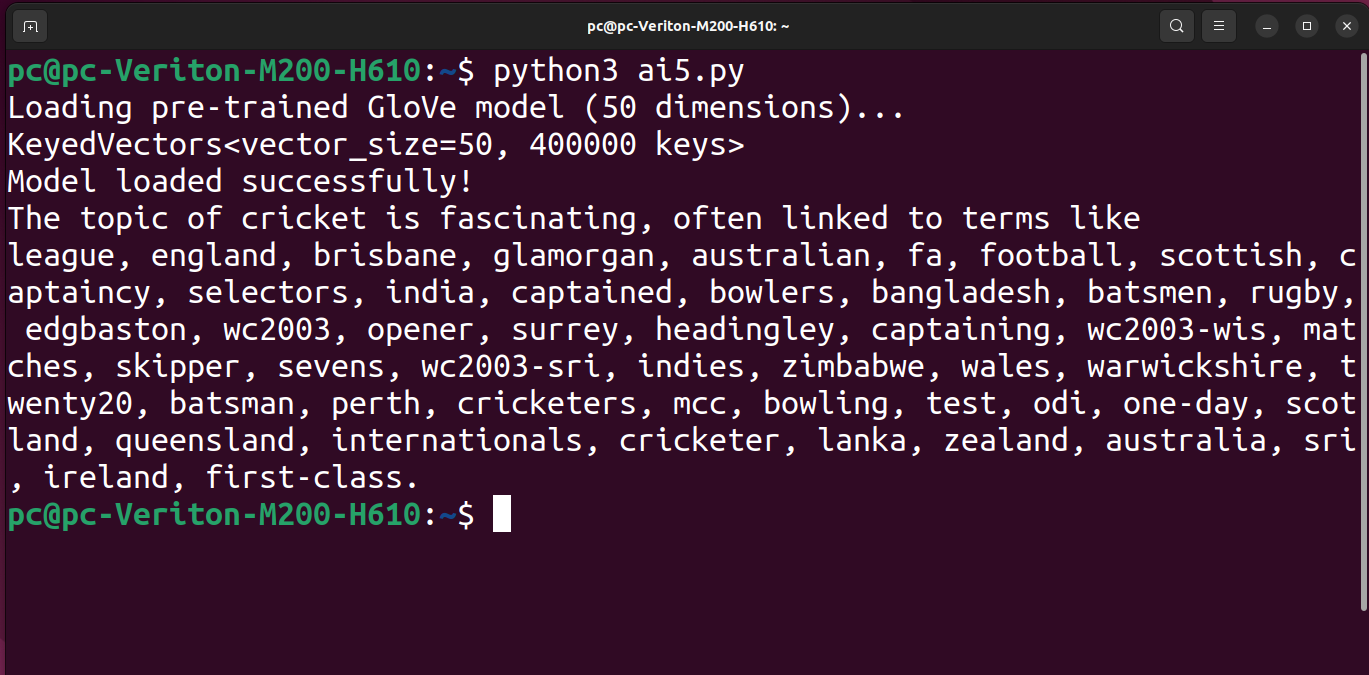
sws = model.most\_similar(iw, topn=50)

words = [word for word, s in sws]

paragraph = create\_paragraph(iw, words)

print(paragraph)

**Output:**



**6. Use a pre-trained Hugging Face model to analyze sentiment in text. Assume a real-world application, Load the sentiment analysis pipeline. Analyze the sentiment by giving sentences to input.**

**Python Code:**

from transformers import pipeline

# Specify the model explicitly

sentiment\_analyzer = pipeline(

"sentiment-analysis",

model="distilbert/distilbert-base-uncased-finetuned-sst-2-english"

)

customer\_feedback = [

"The product is amazing! I love it!",

"Terrible service, I am very disappointed.",

"This is a great experience, I will buy again.",

"Worst purchase I’ve ever made. Completely dissatisfied.",

"I'm happy with the quality, but the delivery was delayed."

]

for feedback in customer\_feedback:

sentiment\_result = sentiment\_analyzer(feedback)

sentiment\_label = sentiment\_result[0]['label']

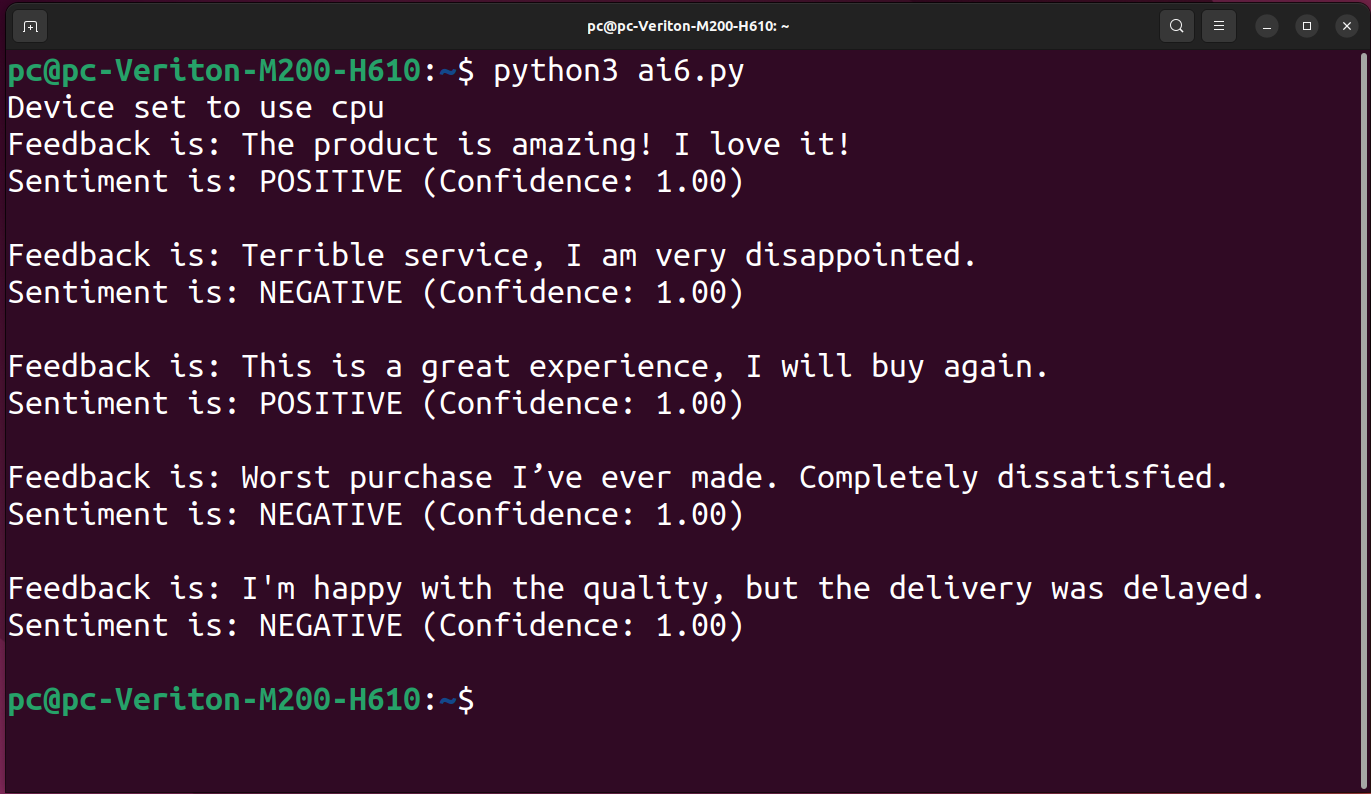
sentiment\_score = sentiment\_result[0]['score']

# Display sentiment results

print(f"Feedback is: {feedback}")

print(f"Sentiment is: {sentiment\_label} (Confidence: {sentiment\_score:.2f})\n")

**Output:**



**7. Summarize long texts using a pre-trained summarization model using Hugging face model. Load the summarization pipeline. Take a passage as input and obtain the summarized text.**

**Python Code:**

from transformers import pipeline

# Specify the model explicitly

summarizer = pipeline(

"summarization",

model="facebook/bart-large-cnn") # You can replace this with any other summarization model if needed

# Function to summarize a given passage

def summarize\_text(text):

# Summarizing the text using the pipeline

summary = summarizer(text, max\_length=150, min\_length=50, do\_sample=False)

return summary[0]['summary\_text']

text = """

Natural language processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language.

The ultimate goal of NLP is to enable computers to understand, interpret, and generate human language in a way that is valuable.

NLP techniques are used in many applications, such as speech recognition, sentiment analysis, machine translation, and chatbot functionality.

Machine learning algorithms play a significant role in NLP, as they help computers to learn from vast amounts of language data and improve their ability to process and generate text.

However, NLP still faces many challenges, such as handling ambiguity, understanding context, and processing complex linguistic structures.

Advances in NLP have been driven by deep learning models, such as transformers, which have significantly improved the performance of many NLP tasks.

"""

# Get the summarized text

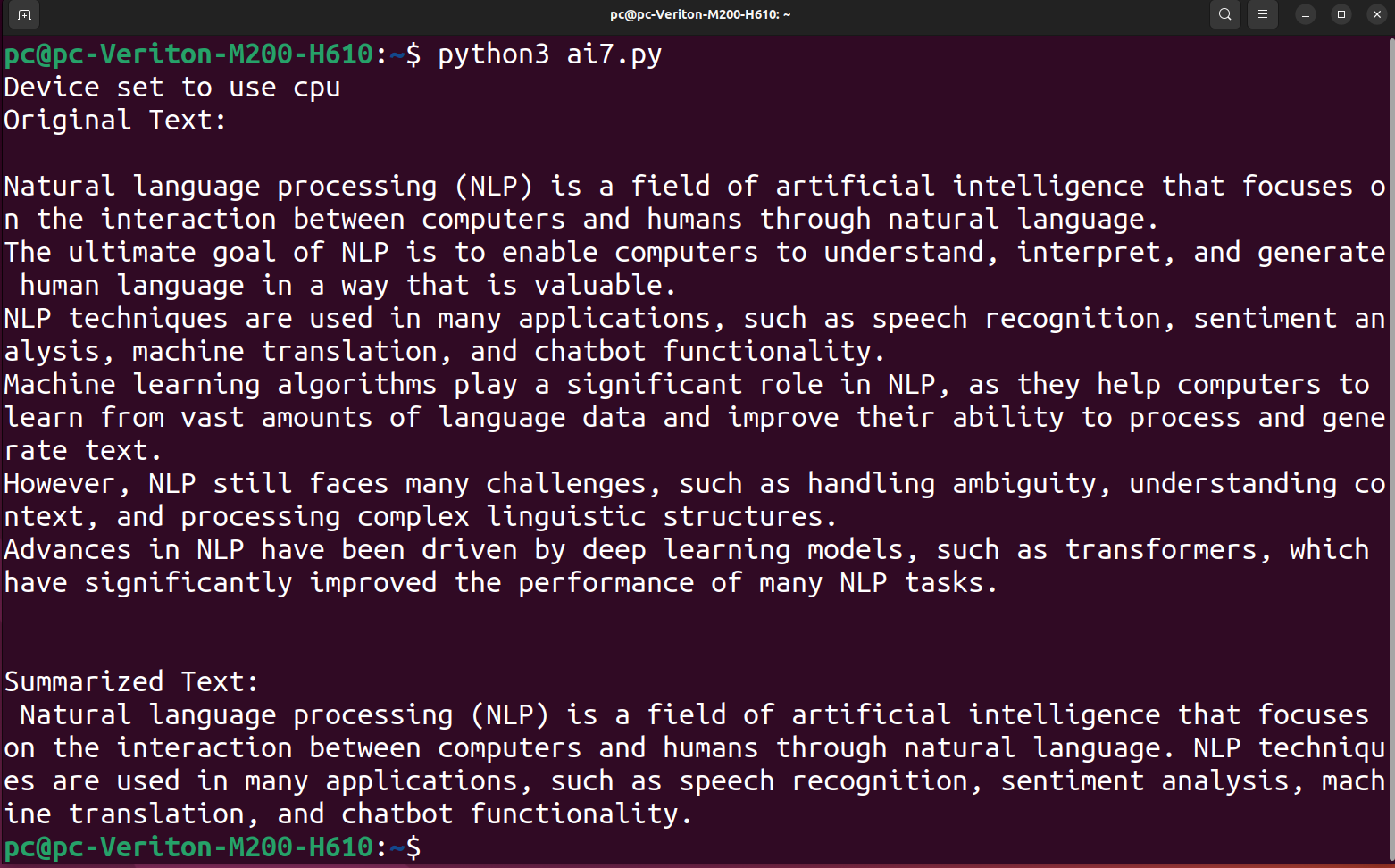
summarized\_text = summarize\_text(text)

# Display the summarized text

print("Original Text:\n", text)

print("\nSummarized Text:\n", summarized\_text)

**Output:**



**9. Take the Institution name as input. Use Pydantic to define the schema for the desired output and create a custom output parser. Invoke the Chain and Fetch Results. Extract the below Institution related details from Wikipedia: The founder of the Institution. When it was founded. The current branches in the institution . How many employees are working in it. A brief 4-line summary of the institution.**

**Python Code:**

from pydantic import BaseModel

import wikipediaapi

# Define the Pydantic Schema

class InstitutionDetails(BaseModel):

name: str

founder: str

founded: str

branches: str

employees: str

summary: str

# Helper function to extract info based on keyword

def extract\_info(content, keyword):

for line in content.split('\n'):

if keyword in line.lower():

return line.strip()

return "Not available"

# Function to Fetch and Extract Details from Wikipedia

def fetch(institution\_name):

user\_agent = "InstitutionInfoFetcher/1.0 (https://example.com; contact@example.com)"

wiki = wikipediaapi.Wikipedia('en', headers={"User-Agent": user\_agent})

page = wiki.page(institution\_name)

if not page.exists():

raise ValueError(f"No Wikipedia page found for '{institution\_name}'")

content = page.text

founder = extract\_info(content, "founder")

founded = extract\_info(content, "founded") or extract\_info(content, "established")

branches = extract\_info(content, "branch")

employees = extract\_info(content, "employee")

summary = "\n".join(content.split('\n')[:4])

return InstitutionDetails(

name=institution\_name,

founder=founder,

founded=founded,

branches=branches,

employees=employees,

summary=summary

)

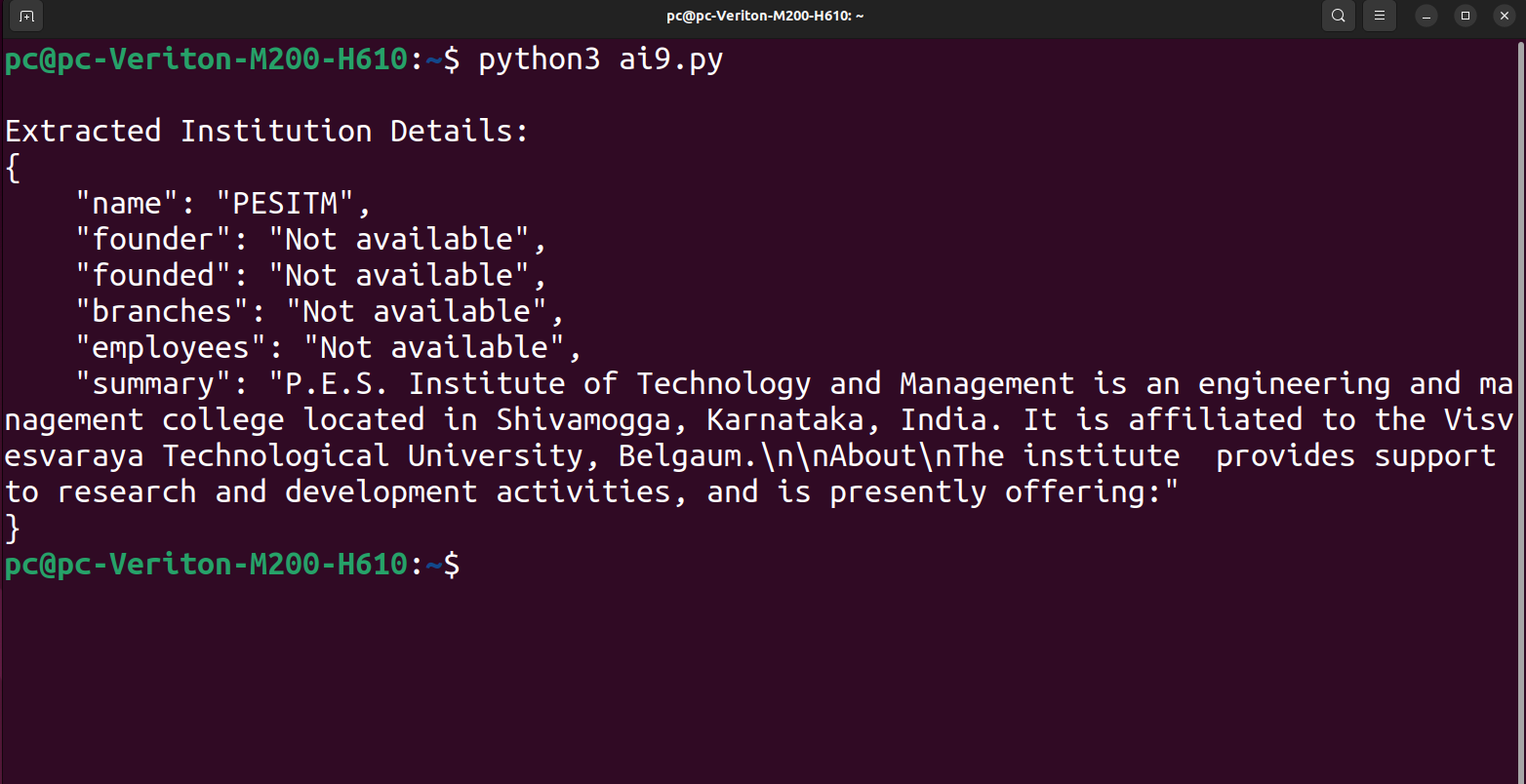
# Run the program

details = fetch("PESITM")

print("\nExtracted Institution Details:")

print(details.model\_dump\_json(indent=4))

**Output:**

****

**10. Build a chatbot for the Indian Penal Code. We'll start by downloading the official Indian Penal Code document, and then we'll create a chatbot that can interact with it. Users will be able to ask questions about the Indian Penal Code and have a conversation with it.**

**Python Code:**

import fitz # PyMuPDF

# Step 1: Extract Text from IPC PDF

def extract(file):

text = ""

with fitz.open(file) as pdf:

for page in pdf:

text += page.get\_text()

return text

# Step 2: Search for Relevant Sections in IPC

def search(query, ipc):

query = query.lower()

lines = ipc.split("\n")

results=[]

for line in lines:

if query in line.lower():

results.append(line)

#results = [line for line in lines if query in line.lower()]

if results:

return results

else:

return ["No relevant section found."]

#return results if results else ["No relevant section found."]

# Step 3: Main Chatbot Function

def chatbot():

print("Loading IPC document...")

ipc = extract("IPC.pdf")

while True:

query = input("Ask a question about the IPC: ")

if query.lower() == "exit":

print("Goodbye!")

break

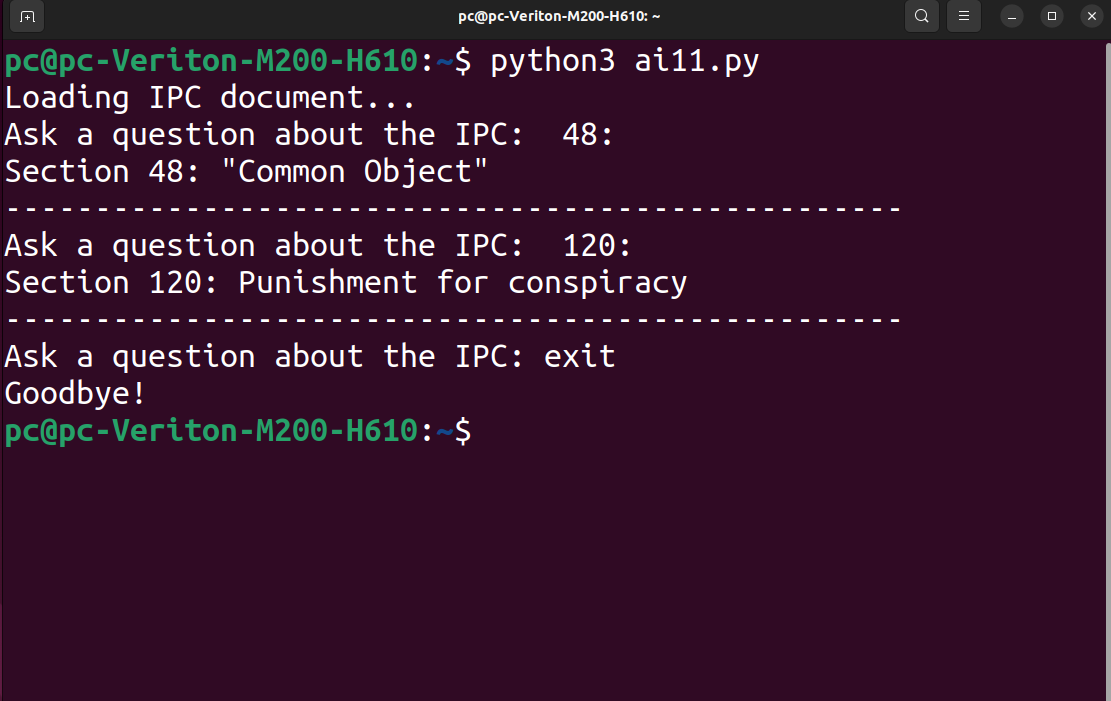
results = search(query, ipc)

print("\n".join(results))

print("-" \* 50)

chatbot()

**Output:**

****