ATME COLLEGE OF ENGINEERING

13th KM Stone, Bannur Road, Mysore - 560028

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(DATA SCIENCE)

(ACADEMIC YEAR 2025-26)

LABORATORY MANUAL

SUBJECT: GENERATIVE AI

SUB CODE: BAIL657C

SEMESTER: VI-2022 CBCS Scheme

Prepared by **Approved by**

Mrs. Madhu Nagaraj
Faculty In-charge

Dr. Anitha D B
HOD, CSE- DS

INSTITUTIONAL VISION AND MISSION

Objectives

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well asthe academia at national and transnational levels
- To develop academic, professional and financial alliances with the industry as well asthe academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

Vision

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

Mission

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow’s society.
- To strive to attain ever-higher benchmarks of educational excellence.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

Vision of the Department

- To impart technical education in the field of data science of excellent quality with a high level of professional competence, social responsibility, and global awareness among the students

Mission

- To impart technical education that is up to date, relevant and makes students competitive and employable at global level
- To provide technical education with a high sense of discipline, social relevance in an intellectually, ethically and socially challenging environment for better tomorrow
- Educate to the global standards with a benchmark of excellence and to kindle the spirit of innovation.

Program Outcomes(PO)

- **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

- **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.
- **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.

- **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.

- **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.

- **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
- **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.

- **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

- **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.
- **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.
- **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 (PSOs)

PSO1: Develop relevant programming skills to become a successful data scientist

- PSO2: Apply data science concepts and algorithms to solve real world problems of the society
- PSO3: Apply data science techniques in the various domains like agriculture, education healthcare for better society

Program Educational Objectives (PEOs):

PEO1: Develop cutting-edge skills in data science and its related technologies, such as machine learning, predictive analytic, and data engineering.

PEO2: Design and develop data-driven solutions to real-world problems in a business, research, or social environment.

PEO3: Apply data engineering and data visualization techniques to discover, investigate, and interpret data.

PEO4: Demonstrate ethical and responsible data practices in problem solving

PEO5: Integrate fields within computer science, optimization, and statistics to develop better solutions

SL. No	Particulars
1	Program 1: Explore pre-trained word vectors. Explore word relationships using vector arithmetic. Perform arithmetic operations and analyze results.
2	Program 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	Program 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	Program 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
5	Program 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	Program 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	Program 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	Program 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	Program 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	Program 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.

Syllabus

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:

a. Develop the ability to explore and analyze word embeddings, perform vector arithmetic to investigate word relationships, visualize embeddings using dimensionality reduction techniques
b. Apply prompt engineering skills to real-world scenarios, such as information retrieval, text generation.
c. 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**.

1. 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.
2. Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
3. Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
4. Weightage to be given for neatness and submission of record/write-up on time.
5. Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
6. 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.
7. The suitable rubrics can be designed to evaluate each student's performance and learning ability.
8. 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
- 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

About Gen AI

Generative AI leverages deep learning techniques such as neural networks and transformers to create new data instances that resemble training data. This field has seen rapid advancements with the rise of generative adversarial networks (GANs) and diffusion models. With the explosion of large-scale models such as OpenAI's GPT series and Google's Bard, AI is reshaping industries by enabling automated creativity and innovation.

Benefits of the Course

1. **Comprehensive Hands-on Learning** – The course provides hands-on experience with generative models, allowing students to work on real-world datasets and build custom AI models.
2. **Industry-Relevant Skills Development** – Students gain expertise in AI model fine-tuning, embedding techniques, and practical applications, making them industry-ready.
3. **Enhancing Creativity and Problem Solving** – The ability to generate human-like content fosters new approaches to solving challenges in media, business automation, and personalized recommendations.
4. **Expanding Career Opportunities** – As AI adoption grows, demand for experts in AI model training, ethical AI development, and prompt engineering increases across domains.
5. **Encouraging AI-Driven Innovation** – Generative AI allows businesses to explore new ideas faster, optimize processes, and build AI-powered creative solutions.

Applications of Generative AI

- **Advanced Chatbots and Conversational AI** – Virtual assistants can respond more naturally and offer human-like interaction.
- **AI in Finance** – Generative AI models are being used for fraud detection, algorithmic trading, and financial forecasting.
- **Code Generation and Software Development** – Tools like GitHub Copilot assist developers by suggesting relevant code snippets and debugging solutions.
- **AI in Marketing and Advertising** – Personalized ad generation, automated social media content creation, and customer sentiment analysis.

Advantages of Learning Generative AI

1. **Ethical AI Considerations** – Understanding bias in AI models and the implications of AI-generated content ensures responsible development and deployment.
2. **Cutting-Edge Research Opportunities** – Generative AI plays a role in groundbreaking research across computational creativity and AI ethics.
3. **AI-powered Automation and Efficiency Gains** – AI-generated content speeds up workflows in content creation, graphic design, and personalized communication.

Course Content Overview

The course delves deeper into:

- **Fine-tuning Pre-trained Models:** Optimizing LLMs for domain-specific tasks.
- **Exploring Transformer Architectures:** Understanding self-attention mechanisms and how they contribute to generative capabilities.
- **Deploying AI Models in Production:** Building scalable AI applications for real-world use cases.
- **Developing Responsible AI:** Addressing bias, fairness, and explainability in generative AI systems.

Lab Programs

Program 1:

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

Theory:

Word Embeddings in NLP

Word Embeddings are numeric representations of words in a lower-dimensional space, capturing semantic and syntactic information. They play a vital role in Natural Language Processing (NLP) tasks.

What is Word Embedding in NLP?

Word Embedding is an approach for representing words and documents. Word Embedding or Word Vector is a numeric vector input that represents a word in a lower-dimensional space. It allows words with similar meanings to have a similar representation.
Word Embeddings are a method of extracting features out of text so that we can input those features into a machine learning model to work with text data.

Need for Word Embedding?

To reduce dimensionality
To use a word to predict the words around it.
Inter-word semantics must be captured.

How are Word Embeddings used?

They are used as input to machine learning models.
Take the words —-> Give their numeric representation —-> Use in training or inference.
To represent or visualize any underlying patterns of usage in the corpus that was used to train them.

Challenges in building word embedding from scratch

Training word embeddings from scratch is possible but it is quite challenging due to large trainable parameters and sparsity of training data. These models need to be trained on a large number of datasets with rich vocabulary and as there are large number of parameters, it makes the training slower. So, it's quite challenging to train a word embedding model on an individual level.

Pre Trained Word Embeddings

There's a solution to the above problem, i.e., using pre-trained word embeddings. Pre-trained word embeddings are trained on large datasets and capture the syntactic as well as semantic meaning of the words. This technique is known as transfer learning in which you take a model which is trained on large datasets and use that model on your own similar tasks.
There are two broad classifications of pre trained word embeddings – word-level and character-level. We'll be looking into two types of word-level embeddings i.e. Word2Vec and GloVe and how they can be used to generate embeddings.

Word2Vec

Word2Vec is one of the most popular pre trained word embedding's developed by Google. It is trained on Good news dataset which is an extensive dataset. As the name suggests, it represents each word with a collection of integers known as a vector. The vectors are calculated such that they show the semantic relation between words.
A popular example of how semantic relation is made is the king queen example:

King - Man + Woman ~ Queen

Word2vec is a feed-forward neural network which consists of two main models – Continuous Bag-of-Words (CBOW) and Skip-gram model. The continuous bag of words model learns the target word from the adjacent words whereas in the skip-gram model, the model learns the adjacent words from the target word. They are completely opposite of each other.

Firstly, the size of context window is defined. Context window is a sliding window which runs through the whole text one word at a time. It basically refers to the number of words appearing on the right and left side of the focus word. eg. if size of the context window is set to 2, then it will include 2 words on the right as well as left of the focus word.
Focus word is our target word for which we want to create the embedding / vector representation

Suppose we have a sentence – "He poured himself a cup of coffee". The target word here is "himself".
Continuous Bag-Of-Words –
input = ["He", "poured", "a", "cup"]
output = ["himself"]
Skip-gram model –
input = ["himself"]
output = ["He", "poured", "a", "cup"]

GloVe

Given by Stanford, GloVe stands for Global Vectors for Word Representation. It is a popular word embedding model which works on the basic idea of deriving the relationship between words using statistics. It is a count based model that employs co-occurrence matrix. A co occurrence matrix tells how often two words are occurring globally. Each value is a count of a pair of words occurring together.
Glove basically deals with the spaces where the distance between words is linked to to their semantic similarity.

GloVe calculates the co-occurrence probabilities for each word pair. It divides the co occurrence counts by the total number of co-occurrences for each word:
F(w_{i}, w_{j}, w_{k}) = \frac {P_{ik}}{P_{jk}}
For example, the co-occurrence probability of "cat" and "mouse" is calculated as: Co occurrence Probability("cat", "mouse") = Count("cat" and "mouse") / Total Co occurrences("cat")

In this case:
Count("cat" and "mouse") = 1
Total Co-occurrences("cat") = 2 (with "chases" and "mouse")
So, Co-occurrence Probability("cat", "mouse") = 1 / 2 = 0.5

Conclusion

In conclusion, word embedding techniques such as TF-IDF, Word2Vec, and GloVe play a crucial role in natural language processing by representing words in a lower-dimensional space, capturing semantic and syntactic information.
Facts

pip install genism

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: gensim in c:\programdata\anaconda3\lib\site-packages (4.3.0)
Requirement already satisfied: numpy>=1.18.5 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from gensim) (1.24.4)
Requirement already satisfied: scipy>=1.7.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from gensim) (1.10.1)
Requirement already satisfied: smart-open>=1.8.1 in c:\programdata\anaconda3\lib\site-packages (from
gensim) (5.2.1)
Requirement already satisfied: FuzzyTM>=0.4.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from gensim) (2.0.9)
Requirement already satisfied: pandas in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from FuzzyTM>=0.4.0->gensim) (1.5.3)
Requirement already satisfied: pyfume in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from FuzzyTM>=0.4.0->gensim) (0.3.4)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\anaconda3\lib\site-packages
(from pandas->FuzzyTM>=0.4.0->gensim) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from
pandas->FuzzyTM>=0.4.0->gensim) (2023.3.post1)
Requirement already satisfied: simpful==2.12.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from pyfume->FuzzyTM>=0.4.0->gensim) (2.12.0)
Requirement already satisfied: fst-pso==1.8.1 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from pyfume->FuzzyTM>=0.4.0->gensim) (1.8.1)
Requirement already satisfied: miniful in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from fst-pso==1.8.1->pyfume->FuzzyTM>=0.4.0->gensim) (0.0.6)
Requirement already satisfied: six>=1.5 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from python-dateutil>=2.8.1->pandas->FuzzyTM>=0.4.0->gensim) (1.17.0)
```

Gensim: A Python library for NLP and word embeddings.

50d, 100d, 6B: Dimensions of GloVe vectors and the size of the training corpus.

GloVe embeddings are converted to Word2Vec format for compatibility with libraries like Gensim, which require the Word2Vec format for efficient vector operations and model functionality.

```
from gensim.scripts.glove2word2vec import glove2word2vec
from gensim.models import KeyedVectors

# Paths to the GloVe file and output Word2Vec file
glove_input_file = "glove.6B/glove.6B.100d.txt"  # Path to GloVe file
word2vec_output_file = "glove.6B/glove.6B.100d.word2vec.txt"  # Output file in Word2Vec format
```

```
# Convert GloVe format to Word2Vec format
glove2word2vec(glove_input_file, word2vec_output_file)
#glove2word2vec("glove.6B.100d.txt", "glove.6B.100d.word2vec.txt")

# Load the converted Word2Vec model
model = KeyedVectors.load_word2vec_format(word2vec_output_file, binary=False)

# Test the loaded model
print(model.most_similar("king"))
```

Output:

```
C:\Users\ANITHA D B\AppData\Local\Temp\ipykernel_7604\71331760.py:9: DeprecationWarning: Call to deprecated `glove2word2vec` (KeyedVectors.load_word2vec_format(.., binary=False, no_header=True) loads GLoVE text vectors.).
  glove2word2vec(glove_input_file, word2vec_output_file)
[('prince', 0.7682328820228577), ('queen', 0.7507690787315369), ('son', 0.7020888328552246), ('brother', 0.6985775232315063), ('monarch', 0.6977890729904175), ('throne', 0.6919989585876465), ('kingdom', 0.6811409592628479), ('father', 0.6802029013633728), ('emperor', 0.6712858080863953), ('ii', 0.6676074266433716)]
```

Explore Word Relationships

Example 1: Find Similar Words

```
similar_to_mysore = model.similar_by_vector(model['mysore'], topn=5)
print(f"Words similar to 'mysore': {similar_to_mysore}")
```

ouput:

```
Words similar to 'mysore': [('mysore', 1.0), ('travancore', 0.6994104385375977), ('cochin', 0.6752076148986816), ('hyderabad', 0.6592637896537781), ('jaipur', 0.6591896414756775)]
```

Example 2: Gender Analogy (actor - man + woman = queen)

```
# Perform vector arithmetic
result_vector_1 = model['actor'] - model['man'] + model['woman']

# Find the most similar word
result_1 = model.similar_by_vector(result_vector_1, topn=1)
print(f"'actor - man + woman' = {result_1}")
```

output:

```
'actor - man + woman' = [('actress', 0.9160683155059814)]
```

Example 3: Country-City Relationship (India - Delhi + Bangalore)

```
# Perform vector arithmetic
result_vector_2 = model['india'] - model['delhi'] + model['washington']

# Find the most similar word
result_2 = model.similar_by_vector(result_vector_2, topn=3)
print(f"'India - Delhi + Washington' = {result_2}")
```

Output:

```
'India - Delhi + Washington' = [('states', 0.8375228643417358), ('united', 0.8281229734420776), ('washington', 0.8155243396759033)]
```

Perform Arithmetic Operations

```
# Example 1: Scaling Vectors
scaled_vector = model['hotel'] * 2  # Scales the 'king' vector by a factor of 2
result_2 = model.similar_by_vector(scaled_vector, topn=3)
result_2
```

Output:

```
[('hotel', 1.0),
 ('hotels', 0.7933705449104309),
 ('restaurant', 0.7762866020202637)]
```

```
#### Example 2: Normalizing Vectors
import numpy as np
normalized_vector = model['fish'] / np.linalg.norm(model['fish'])
result_2 = model.similar_by_vector(normalized_vector, topn=3)
```

Output:

```
average_vector = (model['king'] + model['woman'] + model['man']) / 3
result_2 = model.similar_by_vector(average_vector, topn=3)
result_2
```

Output:

```
[('man', 0.9197071194648743),
 ('woman', 0.8637868165969849),
 ('father', 0.8270207047462463)]
```

## Model Comparision

```
# Paths to the GloVe file and output Word2Vec file
glove_input_file = "glove.6B/glove.6B.50d.txt"  # Path to GloVe file
word2vec_output_file = "glove.6B/glove.6B.50d.word2vec.txt"  # Output file in Word2Vec format

# Convert GloVe format to Word2Vec format
glove2word2vec(glove_input_file, word2vec_output_file)

# Load the converted Word2Vec model
model_50d = KeyedVectors.load_word2vec_format(word2vec_output_file, binary=False)
```

output:

```
C:\Users\ANITHA D B\AppData\Local\Temp\ipykernel_7604\1425925041.py:6: DeprecationWarning: Call to deprecated `glove2word2vec` (KeyedVectors.load_word2vec_format(.., binary=False, no_header=True) loads GLoVE text vectors.).
  glove2word2vec(glove_input_file, word2vec_output_file)
```

Similarity: Measures how close two vectors are in direction using cosine similarity. A value closer to 1 indicates high similarity, while 0 means orthogonal and -1 means opposite directions.
Distance: Measures how far two vectors are from each other in space (e.g., Euclidean distance). A smaller distance indicates the vectors are more similar in magnitude and direction.

### Calculate similarity between two words

```
word1 = "hospital"
word2 = "doctor"

# Similarity in 50d
similarity_50d = model_50d.similarity(word1, word2)

# Similarity in 100d
similarity_100d = model_100d.similarity(word1, word2)

# Results
print(f"Similarity (50d) between '{word1}' and '{word2}': {similarity_50d:.4f}")
print(f"Similarity (100d) between '{word1}' and '{word2}': {similarity_100d:.4f}")
```

Output:

```
Similarity (50d) between 'hospital' and 'doctor': 0.6724
Similarity (100d) between 'hospital' and 'doctor': 0.6901
```

### Calculate distance between two words

```
# Calculate distance between two words
distance_50d = model_50d.distance(word1, word2)
distance_100d = model_100d.distance(word1, word2)


# Results
print(f"Distance (50d) between '{word1}' and '{word2}': {distance_50d:.4f}")
print(f"Distance (100d) between '{word1}' and '{word2}': {distance_100d:.4f}")
```

Output:

```
Distance (50d) between 'hospital' and 'doctor': 0.3276
Distance (100d) between 'hospital' and 'doctor': 0.3099
```

### **Analysis of Results**

1. **'actor - man + woman' = actress (0.916)**
   - The result confirms that the model has captured gender analogies, where subtracting "man" and adding "woman" to "actor" produces the semantically related word "actress."

2. **'India - Delhi + Washington' = ['states', 0.838], ['united', 0.828], ['washington', 0.816]**
   - The arithmetic operation shows that "India - Delhi + Washington" produces words like "states" and "united," suggesting a shift from a city to broader political entities, such as countries or states.

3. **Scaling Vectors ('hotel' * 2) = [('hotel', 1.0), ('hotels', 0.793), ('restaurant', 0.776)]**
   - The scaled vector results in "hotel" being the most similar to itself, and its plural form "hotels" is the second most similar, followed by related terms like "restaurant."

4. **Normalizing Vectors ('fish') = [('fish', 1.0), ('shrimp', 0.779), ('salmon', 0.761)]**
   - Normalizing the vector for "fish" leads to very similar words like "shrimp" and "salmon," which are

semantically related types of fish.

5. **Averaging Vectors ('king' + 'woman' + 'man') / 3 = [('man', 0.920), ('woman', 0.864), ('father', 0.827)]**
   - Averaging the vectors of "king," "woman," and "man" results in "man" and "woman" being the most similar words, indicating that the averaged vector represents a central concept of human relationships.

6. **Similarity and Distance Calculation for 'hospital' and 'doctor':**
   - **Similarity**: 0.6724 (50d) vs. 0.6901 (100d)
     - The similarity between "hospital" and "doctor" is higher in the 100d model, indicating that the higher-dimensional model captures the relationship between these words more accurately.
   - **Distance**: 0.3276 (50d) vs. 0.3099 (100d)
     - The distance between "hospital" and "doctor" is smaller in the 100d model, confirming that the 100d model finds them closer in the vector space, aligning with the similarity results.

Conclusion
- Higher-dimensional models (100d) generally provide more accurate and nuanced word relationships, both in terms of **similarity** and **distance**.
- Arithmetic operations like scaling, averaging, and vector shifts (analogies) allow deeper exploration of word meanings and relationships, and these can vary slightly with model dimensions.

Program 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.

Theory:

To visualize word embedding's, t-SNE (t-distributed Stochastic Neighbour Embedding) is generally preferred over PCA (Principal Component Analysis) because t-SNE is better at capturing complex, non-linear relationships between words, while PCA focuses on preserving global variance and linear patterns, which can be less effective for visualizing the semantic similarities between words in an embedding space.

1. Word Embedding's
Word embedding's are dense vector representations of words in a continuous vector space.
They capture semantic relationships between words, such as similarity, analogy, and context.
Popular pre-trained word embedding models include:
Word2Vec: Trained on large text corpora using a neural network.
GloVe: Global Vectors for Word Representation, based on matrix factorization.
FastText: Extends Word2Vec by representing words as n-grams of characters.

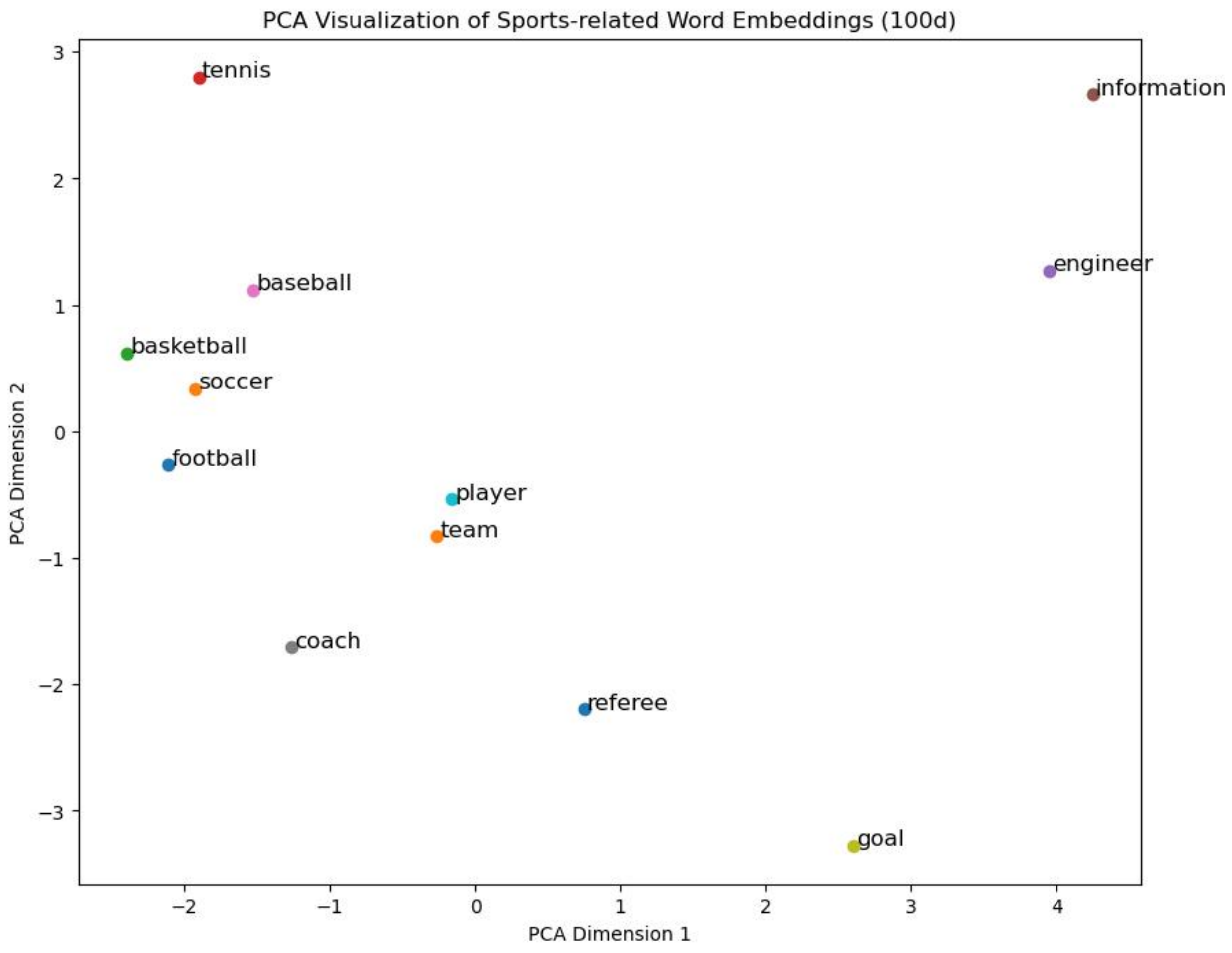
2. Dimensionality Reduction
Word embedding's are high-dimensional (e.g., 300 dimensions). To visualize them, we reduce their dimensionality to 2D or 3D using:
PCA (Principal Component Analysis): A linear technique that projects data onto orthogonal axes.
t-SNE (t-Distributed Stochastic Neighbour Embedding): A non-linear technique that preserves local relationships between points.
3. Semantic Similarity
Semantic similarity measures how closely related two words are in meaning. For example:
Similar Words: "king" and "queen" are semantically similar.

Analogies: "king - man + woman = queen" demonstrates relationships between words.
Both PCA (Principal Component Analysis) and t-SNE (t-Distributed Stochastic Neighbour Embedding) are dimensionality reduction techniques used to visualize high-dimensional data (e.g., word embedding) in 2D or 3D. Below, I explain the parameters of their function calls in Python's scikit-learn library, along with their default values.

Key points to remember:
t-SNE for local structure:
t-SNE excels at preserving local similarities between data points, meaning nearby words in the embedding space will also appear close together in the visualization, making it ideal for understanding semantic relationships between words.
PCA for global variance:
PCA prioritizes capturing the most variance in the data, which can be less useful for visualizing the nuanced relationships between words in an embedding space.
What is t-SNE ?

t-SNE (t-distributed Stochastic Neighbour Embedding) is an unsupervised non-linear dimensionality reduction technique for data exploration and visualizing high-dimensional data. Non-linear dimensionality reduction means that the algorithm allows us to separate data that cannot be separated by a straight line.
t-SNE gives you a feel and intuition on how data is arranged in higher dimensions. It is often used to visualize complex datasets into two and three dimensions, allowing us to understand more about underlying patterns and relationships in the data.
t-SNE vs PCA
Both t-SNE and PCA are dimensional reduction techniques with different mechanisms that work best with different types of data.
PCA (Principal Component Analysis) is a linear technique that works best with data that has a linear structure. It seeks to identify the underlying principal components in the data by projecting onto lower dimensions, minimizing variance, and preserving large pairwise distances. Read our Principal Component Analysis (PCA) tutorial to understand the inner workings of the algorithms with R examples.
But, t-SNE is a nonlinear technique that focuses on preserving the pairwise similarities between data points in a lower-dimensional space. t-SNE is concerned with preserving small pairwise distances whereas, PCA focuses on maintaining large pairwise distances to maximize variance.
In summary, PCA preserves the variance in the data. In contrast, t-SNE preserves the relationships between data points in a lower-dimensional space, making it quite a good algorithm for visualizing complex high-dimensional data.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from gensim.models import KeyedVectors

# Load pre-trained GloVe embeddings (100d model)
        model_100d =
KeyedVectors.load_word2vec_format("glove.6B/glove.6B.100d.word2vec.txt", binary=False)

        # Select 10 words from a specific domain (sports) # Included other words to show how
embeddings are different
        words = ['football', 'soccer', 'basketball', 'tennis','engineer','information', 'baseball',
'coach', 'goal', 'player', 'referee', 'team']
word_vectors = np.array([model_100d[word] for word in words])

# Dimensionality reduction using PCA
# Using PCA to reduce to 2D for visualization
pca = PCA(n_components=2)
pca_result = pca.fit_transform(word_vectors)

# Plotting the words in 2D space
plt.figure(figsize=(10, 8))
for i, word in enumerate(words):
   plt.scatter(pca_result[i, 0], pca_result[i, 1])
   plt.text(pca_result[i, 0] + 0.02, pca_result[i, 1], word, fontsize=12)
plt.title("PCA Visualization of Sports-related Word Embeddings (100d)")
plt.xlabel("PCA Dimension 1")
plt.ylabel("PCA Dimension 2")
plt.show()

# 5 Semantically Similar Words Generator Function
def get_similar_words(word, model, topn=5):
    similar_words = model.similar_by_word(word, topn=topn)
    return similar_words

# Example: Get 5 words similar to "football"
similar_words_football = get_similar_words('football', model_100d, topn=5)
print(f"Words similar to 'football': {similar_words_football}")
```



Output:

```
Words similar to 'football': [('soccer', 0.8732221722602844), ('basketball', 0.8555637001991272), ('league', 0.815336287021637), ('rugby', 0.8007532954216003), ('hockey', 0.7833694815635681)]
```

```
# Select the words you want to print embeddings for
words_to_print = ['football', 'soccer']

# Print their embeddings
for word in words_to_print:
    if word in model_100d:
        print(f"Vector embedding for '{word}':\n{model_100d[word]}\n")
    else:
print(f"Word '{word}' not found in the embeddings model.")
```

Output:

Vector embedding for 'football':

```
Vector embedding for 'football':
[ 0.43865    0.10537    0.45972   -1.0724    -1.2471     0.76351
  0.47528    0.083857  -0.9127    -0.27328   -0.018591  -1.184
  0.22748    0.16847   -0.52158    0.11339    1.3757     0.11892
 -0.37683    0.51149   -0.8833     0.96259    0.18143   -0.407
  0.036181  -0.74432   -0.0027401 -0.70068    0.53103    0.45114
 -0.72884    1.0631    -0.28008   -0.63848    0.15645   -0.46927
 -1.0071     1.033     -1.4354    -0.27485    0.048984   0.13951
  0.43072   -0.78791    0.41097    0.58509    1.0155    -0.1839
  0.27487   -0.90866   -0.30441   -0.17396    0.020941   0.62813
  0.10978   -2.3885    -0.56364   -0.27193    0.98728    0.70608
 -0.512      0.52636   -0.78503   -0.68714    0.38121    0.097582
 -0.20237    0.43208   -0.30527    0.57925    0.62619   -0.47415
  0.33834   -0.28421   -0.097465   0.19597    0.54849    0.59918
 -0.41576    0.1021     0.6766     0.0042009 -0.12354   -0.76613
 -0.27436   -0.68248   -1.0789    -0.16708    0.81671    0.026999
 -0.38707    0.40448   -1.0995     0.64718   -0.12802   -0.26084
 -0.96701    0.88078    1.012     -0.022223 ]

Vector embedding for 'soccer':
[ 8.3777e-01  5.1890e-01  6.4015e-01 -6.2606e-01 -9.7474e-01  1.0127e+00
  6.2729e-02  4.4316e-01 -8.3299e-01  7.9888e-02 -1.1815e-02 -1.1265e+00
  1.2554e-01 -3.4206e-01 -5.1422e-01  3.8526e-01  1.0032e+00 -1.5172e-03
 -2.2684e-01  3.5658e-01 -6.2449e-01  8.7271e-01  3.6670e-01  4.6462e-01
 -1.0046e-01 -4.4798e-01 -2.1813e-01 -5.6423e-01  5.6665e-01  5.1601e-01
-5.6511e-01  7.1919e-01 -6.5347e-01 -9.5952e-02  5.6028e-01 -4.9956e-01
 -7.4757e-01  6.8516e-01 -1.4518e+00 -1.1207e-01  1.0241e-01  3.0537e-02
  1.1326e-02 -8.6873e-01  6.3622e-01  4.9539e-01  3.0538e-01  7.7133e-02
  7.4048e-02 -7.1163e-01 -1.9159e-01 -3.4168e-01 -4.7185e-01  5.6794e-01
  3.7454e-01 -1.9207e+00 -8.6040e-01  5.7058e-01  1.0700e+00  9.2101e-01
 -6.4825e-01  5.3516e-01 -1.5556e-01 -9.0021e-01 -1.7459e-01  3.3146e-02
 -5.7512e-01  2.9963e-01 -4.0008e-01 -1.0765e-01  4.1384e-01 -7.2178e-01
  1.1442e-01 -2.1291e-01  5.4949e-02  1.3213e-01  7.8766e-01  8.9291e-02
 -6.6689e-01  3.3998e-01  9.7163e-01 -8.4871e-02  1.7542e-01 -4.6039e-01
 -8.5885e-02 -7.5960e-01 -1.5071e+00  2.1545e-01  2.1209e-01 -4.4837e-01
 -2.5882e-01  3.3814e-01 -4.7979e-01  2.1059e-01  2.3621e-01 -3.6699e-01
 -8.1440e-01  5.4515e-01  9.7946e-01  2.3367e-01]
```

Program 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.

Theory:

This program builds a domain-specific Word2Vec model for a medical corpus, visualizes the learned word embeddings using t-SNE, and retrieves similar words using the trained model.

1. Word Embeddings & Word2Vec

   Word Embeddings are numerical vector representations of words that capture their meanings and relationships.

   Word2Vec is a popular neural network-based algorithm that learns word embeddings from a corpus.

   There are two architectures for Word2Vec:

o CBOW (Continuous Bag of Words): Predicts a word based on surrounding context words.

o Skip-Gram: Predicts surrounding context words given a central word.

In this program, we use the Word2Vec model to train embeddings on a medical corpus, capturing the domain-specific relationships between words.

2. Pre-processing the Corpus

   Tokenization: Splitting sentences into words.

   Lowercasing: Normalizing text to avoid duplication due to case differences

3. Training the Word2Vec Model

   Hyperparameters used in training:

   - vector_size=100: Each word is represented by a 100-dimensional vector.
   - window=5: Words within a 5-word context window influence each other.
   - min_count=1: Even words occurring once are included in training.
   - workers=4: Uses 4 CPU threads for parallel processing.
   - epochs=50: The training iterates over the dataset 50 times for better learning.

4. Extracting Word Embeddings for Visualization

   Embeddings: After training, every word has a 100-dimensional vector representation.

   Dimensionality Reduction: We apply t-SNE (t-distributed Stochastic Neighbor Embedding) to reduce 100 dimensions to 2D for visualization.

5. Visualizing Word Embeddings using t-SNE

   Why use t-SNE?

- Helps visualize high-dimensional data in 2D space.
- Maintains the relative closeness of similar words.
- Scatter Plot
- Words that are semantically related will cluster together.
- Example: "vaccine" and "infection" may appear close to each other.

6. Finding Semantically Similar Words

   Uses the trained Word2Vec model to retrieve top-N most similar words.

   Cosine Similarity is used to find similarity between word vectors:

Important Steps

1.Tokenization: Converts sentences into lists of lowercase tokens for processing.
2.Word2Vec Training:
 vector_size: Sets the embedding dimension to 50.
 window: Uses a context window of 3 words.
 sg: Skip-gram (sg=1) is used, which works better for smaller datasets.
 epochs: The number of training iterations.
3.Visualization: PCA reduces the high-dimensional word vectors to 2D for visualization, helping to understand semantic relationships.
4.Semantic Analysis: The most_similar method identifies words that are semantically similar based on embeddings.

Important Steps
1. **Tokenization**: Converts sentences into lists of lowercase tokens for processing.
2. **Word2Vec Training**:
 - \`vector_size\`: Sets the embedding dimension to 50.
 - \`window\`: Uses a context window of 3 words.
 - \`sg\`: Skip-gram (sg=1) is used, which works better for smaller datasets.
 - \`epochs\`: The number of training iterations.
3. **Visualization**: PCA reduces the high-dimensional word vectors to 2D for visualization, helping to understand semantic relationships.
4. **Semantic Analysis**: The \`most_similar\` method identifies words that are semantically similar based on embeddings.

Example: Legal Corpus

```
from gensim.models import Word2Vec
from gensim.utils import simple_preprocess
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

```
legal_corpus = [
    "The court ruled in favor of the plaintiff.",
    "The defendant was found guilty of negligence.",
    "A breach of contract case was filed.",
    "The agreement between parties must be honored.",
    "The lawyer presented compelling evidence.",
    "Legal documents must be drafted carefully.",
    "The jury deliberated for several hours.",
    "A settlement was reached between the parties.",
    "The plaintiff claimed damages for losses incurred.",
    "The contract outlined the obligations of both parties."
]
```

```
# Example legal corpus
legal_corpus = [
    "The court ruled in favor of the plaintiff.",
    "The defendant was found guilty of negligence.",
    "A breach of contract case was filed.",
    "The agreement between parties must be honored.",
    "The lawyer presented compelling evidence.",
    "Legal documents must be drafted carefully.",
    "The jury deliberated for several hours.",
    "A settlement was reached between the parties.",
    "The plaintiff claimed damages for losses incurred.",
    "The contract outlined the obligations of both parties."
 ]

 # Preprocess the corpus
tokenized_corpus = [simple_preprocess(sentence) for sentence in legal_corpus]

# Train the Word2Vec model
legal_word2vec = Word2Vec(
   sentences=tokenized_corpus,
   vector_size=50,  # Embedding dimension
   window=3,        # Context window size
   min_count=1,     # Minimum word frequency
   sg=1,            # Skip-gram model
   epochs=100       # Training epochs
)

# Save the model for later use
legal_word2vec.save("legal_word2vec.model")
# Train the Word2Vec model
legal_word2vec = Word2Vec(
   sentences=tokenized_corpus,
   vector_size=50,  # Embedding dimension
   window=3,        # Context window size
   min_count=1,     # Minimum word frequency
   sg=1,            # Skip-gram model
   epochs=100       # Training epochs
)

# Save the model for later use
legal_word2vec.save("legal_word2vec.model")
# Train the Word2Vec model
legal_word2vec = Word2Vec(
   sentences=tokenized_corpus,
   vector_size=50,  # Embedding dimension
   window=3,        # Context window size
   min_count=1,     # Minimum word frequency
   sg=1,            # Skip-gram model
   epochs=100       # Training epochs
)

# Save the model for later use
legal_word2vec.save("legal_word2vec.model")
# Analyze embeddings: Display vector for a specific word
word = "lawyer"
if word in legal_word2vec.wv:
   print(f"Vector embedding for '{word}':\n{legal_word2vec.wv[word]}\n")
else:
   print(f"Word '{word}' not found in the Word2Vec model.")
```

Output:

```
Vector embedding for 'lawyer':
[ 0.00373483  0.01353383  0.00585796 -0.01324683  0.01500349 -0.01261986
  0.01892563  0.00698961 -0.0087639  -0.01023367 -0.00875896 -0.01318524
  0.01972703 -0.00463062  0.01525868 -0.01837575  0.0055629  -0.00126356
  0.01417167 -0.01969541  0.01564029 -0.00948072 -0.0107858  -0.01128642
 -0.00610619 -0.00604345 -0.00693252 -0.01396556  0.00086967 -0.00136903
 -0.00358557  0.00685404 -0.01432065 -0.00657563  0.00952303  0.01720192
 -0.01858611  0.01418636  0.01038651 -0.00818817  0.01832661 -0.01858529
  0.01404059  0.01154918  0.00326395 -0.01036671 -0.00841038 -0.00736812
  0.00374052  0.00413726]
```

```
# Visualize embeddings using PCA
words_to_visualize = ["court", "plaintiff", "defendant", "agreement", "lawyer", "evidence",
"contract", "settlement", "jury", "damages"]
word_vectors = [legal_word2vec.wv[word] for word in words_to_visualize]
```

```
word_vectors
```

Output:

```
[array([-0.01018794, -0.0037532 , -0.01479373,  0.00535417,  0.00549183,
       -0.00194653, -0.00904275, -0.00120178,  0.01239534,  0.005502  ,
       -0.01752885, -0.00888894,  0.00678894,  0.00598825, -0.01972261,
        0.01158325, -0.01438892, -0.01200779,  0.00463451, -0.01056976,
        0.00906795,  0.01991566, -0.00384839,  0.01845003,  0.00452612,
        0.02153785,  0.0106156 , -0.0164802 , -0.0075984 ,  0.01259563,
        0.01069134,  0.01610584,  0.01608272,  0.01619358, -0.02157517,
        0.00898223, -0.00762749,  0.00642556,  0.01106042,  0.00757853,
        0.01795846,  0.00227335, -0.00347768,  0.01356644, -0.00962057,
        0.00016249,  0.01841913, -0.01246461,  0.00897428, -0.01424266],
      dtype=float32),
 array([-0.01382369,  0.0011437 , -0.01449836, -0.00296123,  0.00624782,
        0.00982854,  0.00482432,  0.00674102, -0.01035763,  0.0125059 ,
       -0.01251031,  0.00691753, -0.01420641,  0.00583739, -0.01051113,
       -0.00608784, -0.00284186,  0.01448168,  0.00904517, -0.01370935,
        0.00321437, -0.01510567,  0.01918532,  0.01829434, -0.00426899,
        0.00343321,  0.00058916,  0.01309333, -0.0183534 ,  0.00069488,
        0.0132396 ,  0.0028656 ,  0.00451153, -0.01875341,  0.01468391,
       -0.01201477, -0.00313035,  0.00620906, -0.0025351 ,  0.00151398,
        0.0066815 , -0.01435055, -0.02045849,  0.01987134,  0.01433266,
       -0.01331776,  0.00661788, -0.00128313,  0.01081608, -0.01213262],
      dtype=float32),
 array([-0.01735372,  0.00324048, -0.00153466, -0.01745638, -0.01992291,
       -0.00444436,  0.01032289,  0.00879421, -0.01455397, -0.01536747,
       -0.01001598, -0.00653257, -0.0128883 , -0.01829896, -0.0059358 ,
       -0.01495283, -0.00974466, -0.00899372, -0.00697665, -0.00573702,
       -0.01700949,  0.0003172 ,  0.01874463,  0.01480774, -0.01384414,
       -0.00612229,  0.00565069, -0.01732042,  0.00195022,  0.01274919,
        0.01080692, -0.01920946, -0.00795812, -0.01638088, -0.00148547,
        0.01870203,  0.01399906,  0.00958245,  0.00941055, -0.00658938,
        0.02153141, -0.01520018, -0.01545056, -0.00327682,  0.00024155,
       -0.00606883, -0.00135896,  0.01406399,  0.00023136, -0.00163963],
      dtype=float32),
 array([ 0.00544287,  0.01555425, -0.00307019,  0.01717134,  0.00693973,
       -0.0170452 , -0.00679161, -0.00333116,  0.00903156, -0.00295565,
       -0.00578579,  0.01436892,  0.02001157, -0.00213262, -0.01079166,
        -0.007799  , -0.00751752, -0.01736892,  0.00095211, -0.01050753,
         0.00615652,  0.01262798, -0.00638232, -0.01966911,  0.00394809,
        -0.01237557,  0.0045896 , -0.00610946,  0.01346509,  0.00106505,
         0.00631289, -0.00667795, -0.00218376,  0.01535427,  0.00144457,
        -0.0117866 , -0.01405202,  0.00186158,  0.0125593 ,  0.0105592 ,
        -0.01650265,  0.01693893,  0.0074757 ,  0.0163192 ,  0.02056517,
        -0.01457632, -0.01834234,  0.01092377,  0.01994778,  0.00864314],
       dtype=float32),
 array([ 0.00373483,  0.01353383,  0.00585796, -0.01324683,  0.01500349,
        -0.01261986,  0.01892563,  0.00698961, -0.0087639 , -0.01023367,
        -0.00875896, -0.01318524,  0.01972703, -0.00463062,  0.01525868,
        -0.01837575,  0.0055629 , -0.00126356,  0.01417167, -0.01969541,
         0.01564029, -0.00948072, -0.0107858 , -0.01128642, -0.00610619,
        -0.00604345, -0.00693252, -0.01396556,  0.00086967, -0.00136903,
        -0.00358557,  0.00685404, -0.01432065, -0.00657563,  0.00952303,
         0.01720192, -0.01858611,  0.01418636,  0.01038651, -0.00818817,
         0.01832661, -0.01858529,  0.01404059,  0.01154918,  0.00326395,
        -0.01036671, -0.00841038, -0.00736812,  0.00374052,  0.00413726],
       dtype=float32),
 array([ 0.00550566,  0.00103798, -0.00515228,  0.01945088,  0.00499871,
         0.00736707, -0.0011947 ,  0.00298867,  0.01247635, -0.00248031,
         0.00660107, -0.00238972,  0.01178223,  0.00798718,  0.00505932,
        -0.00936528, -0.00755702,  0.00989482, -0.01304692, -0.00193519,
        -0.00039899,  0.0078729 , -0.01549838,  0.01741308, -0.0023178 ,
        -0.00983727,  0.00754468, -0.0027872 , -0.01603217, -0.00921708,
        -0.00134961, -0.01871502,  0.002125  ,  0.00480915, -0.00744796,
         0.00537565,  0.00629158,  0.01973929,  0.0024904 ,  0.00340102,
         0.00710946, -0.00441335, -0.01761757,  0.01698   , -0.0031966 ,
        -0.0194808 , -0.01307702, -0.00849545,  0.00867249,  0.01145031],
       dtype=float32),
 array([ 0.00298495, -0.00700375, -0.01431873, -0.01400242, -0.01991   ,
        -0.01428693, -0.00105788, -0.00551727, -0.0153189 , -0.0100668 ,
         0.00858984, -0.01069687,  0.01958971,  0.00508815, -0.01531299,
         0.02237322,  0.01962719,  0.01488377, -0.01710452,  0.00707861,
         0.01021231,  0.01304598,  0.01277774,  0.00337116, -0.00486931,
         0.01909359,  0.01800028,  0.01032766, -0.00758116, -0.00048564,
         0.00164387, -0.0189941 , -0.01410591, -0.00047871, -0.00010738,
        -0.01102702, -0.0061726 , -0.01550268,  0.0161471 , -0.00069464,
        -0.00545253,  0.01093123,  0.01718066, -0.00617879,  0.0186232 ,
         0.01143978,  0.01163601, -0.00062903,  0.01886317, -0.0114121 ],
       dtype=float32),
 array([-0.00976338, -0.00780081,  0.019559  ,  0.01830878, -0.00654693,
         0.01011876,  0.01784409, -0.00301464,  0.01761319,  0.01262996,
         0.00393919, -0.00980376, -0.00886089, -0.00519702,  0.01557352,
        -0.01077065, -0.00356288,  0.02101176,  0.00479641,  0.01005143,
        -0.01529913,  0.00012613,  0.01357427, -0.01018804,  0.01573833,
         0.02000298, -0.00148577, -0.00239202, -0.00189635, -0.01172749,
        -0.01742925, -0.00479667, -0.00404787,  0.00869905, -0.01522061,
         0.01094797, -0.01160657, -0.0163735 ,  0.01649981,  0.01770434,
        -0.00497504,  0.00637913, -0.01261914, -0.0161758 , -0.00964475,
         0.01381735,  0.01255536,  0.01808335,  0.01568656,  0.01504712],
       dtype=float32),
 array([ 0.01492715,  0.01934095,  0.01774599, -0.00747902,  0.01891196,
        -0.0020531 ,  0.01053821,  0.00635226, -0.00197045,  0.00632444,
        -0.01059867, -0.01259004, -0.01428318,  0.00465197,  0.01267453,
         0.0028981 ,  0.00384051,  0.00779968,  0.01519439, -0.01727828,
        0.00548228, -0.01392599,  0.00899558,  0.01923842,  0.01556924,
        0.01372755,  0.01566461,  0.01412691,  0.01313736,  0.01728814,
       -0.01028677,  0.01760968,  0.01114872, -0.00440745,  0.01607866,
        0.01023387,  0.02058647,  0.00533376,  0.01917837,  0.00176341,
        0.01960336,  0.0070012 ,  0.01266869, -0.00624314,  0.01447076,
        0.01456392, -0.00432669, -0.00459186,  0.00780778, -0.01304201],
      dtype=float32),
 array([ 0.0158812 ,  0.0174168 ,  0.00195527, -0.01554414,  0.01595952,
       -0.00898288,  0.0134456 ,  0.01096715,  0.01782416, -0.02043355,
        0.01853634, -0.02043897, -0.01187125, -0.01672532, -0.01152777,
        0.01697107,  0.02129747, -0.00410723,  0.0053023 , -0.01103053,
        0.017639  ,  0.01337754,  0.0028419 , -0.00731513, -0.01343816,
        0.01128781,  0.00173393, -0.00338352,  0.01469343, -0.00834176,
       -0.01866035, -0.00566033,  0.00234445, -0.0135692 , -0.0126051 ,
       -0.01704373,  0.02071049,  0.0147259 , -0.00145971,  0.01323994,
        0.01840546, -0.0020906 ,  0.0126531 ,  0.0093615 ,  0.01196339,
        0.01458218, -0.00961088, -0.00608193,  0.00305689,  0.01033708],
      dtype=float32)]
```

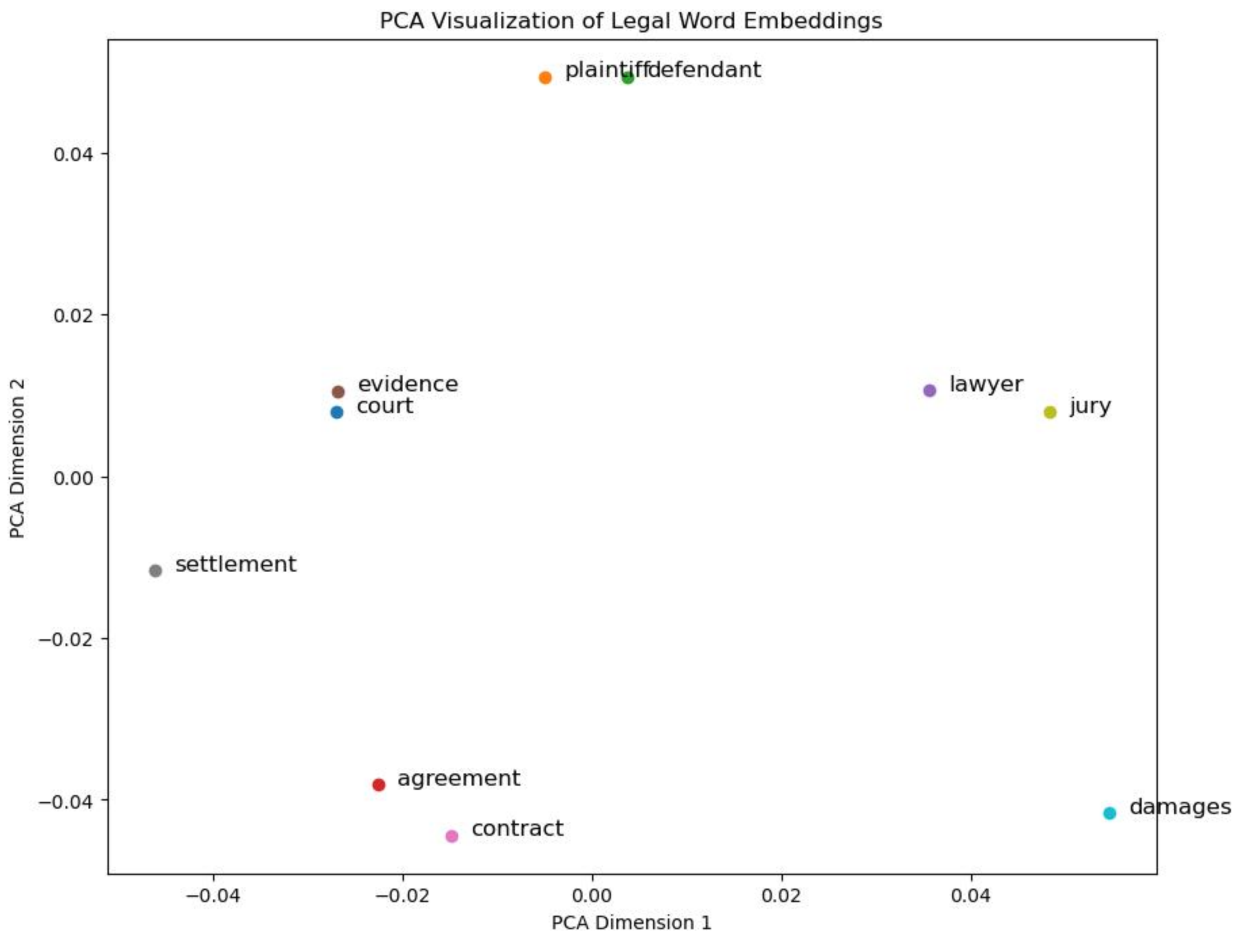
```
# Dimensionality reduction
pca = PCA(n_components=2)
reduced_vectors = pca.fit_transform(word_vectors)
reduced_vectors
```

Output:

```
array([[ 0.02688162, -0.00792018],
       [ 0.00493226, -0.04934309],
       [-0.00377306, -0.04936944],
       [ 0.02256997,  0.03808062],
       [-0.0355795 , -0.01066101],
       [ 0.02682294, -0.01050709],
       [ 0.01486912,  0.0443972 ],
       [ 0.04605154,  0.01166099],
       [-0.0482769 , -0.0079725 ],
       [-0.05449799,  0.0416345 ]])
```

```
# Plot embeddings
plt.figure(figsize=(10, 8))
for i, word in enumerate(words_to_visualize):
    plt.scatter(reduced_vectors[i, 0], reduced_vectors[i, 1])
    plt.text(reduced_vectors[i, 0] + 0.002, reduced_vectors[i, 1], word, fontsize=12)
plt.title("PCA Visualization of Legal Word Embeddings")
plt.xlabel("PCA Dimension 1")
plt.ylabel("PCA Dimension 2")
plt.show()
```

Output:


PCA Visualization of Legal Word Embeddings
plaintiff
defendant
evidence
court
lawyer
jury
settlement
agreement
contract
damages
PCA Dimension 1
PCA Dimension 2
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04

```
# Find similar words
similar_words = legal_word2vec.wv.most_similar("lawyer", topn=5)
print(f"Words similar to 'lawyer': {similar_words}")
```

Output:

```
Words similar to 'lawyer': [('carefully', 0.29186686873435974), ('claimed', 0.27888569235801697), ('jury',
0.21892617642879486), ('damages', 0.1961500644683838), ('negligence', 0.1820133775472641)]
```

### Example: Legal and Medical / Healthcare Corpus

```
from gensim.models import Word2Vec
from gensim.utils import simple_preprocess
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Enhanced legal and medical corpus
enhanced_corpus = [
    # Legal domain
    "The court ordered the immediate release of the detained individual due to lack of evidence.",
    "A new amendment was introduced to ensure the protection of intellectual property rights.",
    "The defendant pleaded not guilty, citing an alibi supported by credible witnesses.",
    "The plaintiff accused the company of violating environmental regulations.",
    "A settlement agreement was reached through arbitration, avoiding a lengthy trial.",
    "The legal team presented a compelling argument to overturn the previous judgment.",
    "Contractual obligations must be fulfilled unless waived by mutual consent.",
    "The jury found the accused guilty of fraud and embezzlement.",
    "The appeal was dismissed as the evidence presented was deemed inadmissible.",
    "The attorney emphasized the importance of adhering to constitutional rights.",

    # Medical domain
    "The patient was admitted to the emergency department with severe chest pain.",
    "The surgeon successfully performed a minimally invasive procedure to remove the tumor.",
    "Clinical trials showed significant improvement in patients treated with the experimental drug.",
    "Regular screening is essential for early detection of chronic illnesses such as diabetes.",
    "The doctor recommended physical therapy to improve mobility after surgery.",
    "The hospital implemented stringent protocols to prevent the spread of infectious diseases.",
    "The nurse monitored the patient's vital signs hourly to ensure stability.",
    "Vaccination campaigns have drastically reduced the prevalence of polio worldwide.",
    "The radiologist identified a small abnormality in the CT scan requiring further investigation.",
    "Proper nutrition and exercise are vital components of a healthy lifestyle."
]
```

```
# Preprocess the corpus
tokenized_corpus = [simple_preprocess(sentence) for sentence in enhanced_corpus]
tokenized_corpus
```

Output;

```
[['the',
  'court',
  'ordered',
  'the',
  'immediate',
  'release',
 'of',
 'the',
 'detained',
 'individual',
 'due',
 'to',
 'lack',
 'of',
 'evidence'],
['new',
 'amendment',
 'was',
 'introduced',
 'to',
 'ensure',
 'the',
 'protection',
 'of',
 'intellectual',
 'property',
 'rights'],
['the',
 'defendant',
 'pleaded',
 'not',
 'guilty',
 'citing',
 'an',
 'alibi',
 'supported',
 'by',
 'credible',
 'witnesses'],
['the',
 'plaintiff',
 'accused',
 'the',
 'company',
 'of',
 'violating',
 'environmental',
 'regulations'],
['settlement',
 'agreement',
 'was',
 'reached',
 'through',
 'arbitration',
 'avoiding',
 'lengthy',
 'trial'],
['the',
 'legal',
 'team',
 'presented',
 'compelling',
 'argument',
 'to',
 'overturn',
 'the',
 'previous',
 'judgment'],
['contractual',
 'obligations',
 'must',
 'be',
 'fulfilled',
 'unless',
 'waived',
 'by',
 'mutual',
 'consent'],
['the',
 'jury',
 'found',
 'the',
 'accused',
 'guilty',
 'of',
 'fraud',
 'and',
 'embezzlement'],
['the',
 'appeal',
 'was',
 'dismissed',
 'as',
 'the',
 'evidence',
 'presented',
 'was',
 'deemed',
 'inadmissible'],
['the',
 'attorney',
 'emphasized',
 'the',
 'importance',
 'of',
 'adhering',
 'to',
 'constitutional',
 'rights'],
['the',
 'patient',
 'was',
 'admitted',
 'to',
 'the',
 'emergency',
 'department',
 'with',
 'severe',
 'chest',
 'pain'],
['the',
 'surgeon',
 'successfully',
 'performed',
 'minimally',
 'invasive',
 'procedure',
 'to',
 'remove',
 'the',
 'tumor'],
['clinical',
 'trials',
 'showed',
 'significant',
 'improvement',
 'in',
 'patients',
 'treated',
 'with',
 'the',
 'experimental',
 'drug'],
['regular',
 'screening',
 'is',
 'essential',
 'for',
 'early',
 'detection',
 'of',
 'chronic',
 'illnesses',
 'such',
 'as',
 'diabetes'],
['the',
 'doctor',
 'recommended',
 'physical',
 'therapy',
 'to',
 'improve',
 'mobility',
 'after',
 'surgery'],
['the',
 'hospital',
 'implemented',
 'stringent',
 'protocols',
 'to',
 'prevent',
 'the',
 'spread',
 'of',
 'infectious',
 'diseases'],
['the',
 'nurse',
 'monitored',
 'the',
 'patient',
 'vital',
 'signs',
 'hourly',
 'to',
 'ensure',
 'stability'],
['vaccination',
 'campaigns',
 'have',
 'drastically',
 'reduced',
 'the',
 'prevalence',
 'of',
 'polio',
 'worldwide'],
['the',
 'radiologist',
 'identified',
 'small',
 'abnormality',
 'in',
 'the',
 'ct',
 'scan',
 'requiring',
 'further',
 'investigation'],
['proper',
 'nutrition',
 'and',
 'exercise',
 'are',
 'vital',
 'components',
 'of',
 'healthy',
 'lifestyle']]
```

```
# Train Word2Vec
domain_word2vec = Word2Vec(
    sentences=tokenized_corpus,
    vector_size=100,  # Higher embedding dimension for better representation
    window=5,         # Wider context window
    min_count=1,      # Include all words
    sg=1,             # Skip-gram model
    epochs=150        # More training iterations
)
```

```
# Save the model
domain_word2vec.save("enhanced_domain_word2vec.model")
```

```
# Analyze embeddings: Get vectors for specific words
words_to_analyze = ["court", "plaintiff", "doctor", "patient", "guilty", "surgery"]
for word in words_to_analyze:
    if word in domain_word2vec.wv:
        print(f"Vector embedding for '{word}':\n{domain_word2vec.wv[word]}\n")
    else:
        print(f"Word '{word}' not found in the Word2Vec model.")
```

output;

```
Vector embedding for 'court':
[-0.00520213  0.05436571  0.0196009   0.00766893  0.04851889 -0.22194375
  0.15068555  0.2671535  -0.16717364 -0.04062838 -0.054865   -0.17729442
 -0.06285486  0.16066416  0.00799252  0.00430546 -0.04130681 -0.11852198
 -0.11586928 -0.32001996  0.07377547  0.00634967  0.01555517 -0.04018658
 -0.05180506 -0.06574838  0.01809591 -0.04998898 -0.05094941  0.00987862
  0.17092119 -0.03111312  0.12419216 -0.07877786 -0.07952873  0.22328345
  0.12608306 -0.0951244  -0.07667849 -0.1501351   0.04725789 -0.15457962
 -0.06896634  0.13114625  0.11142956  0.03642106 -0.06946036 -0.02198208
  0.01422113  0.05933676  0.09983439 -0.12603386  0.07056595  0.02597529
 -0.02668819  0.0757888  -0.00033602  0.05289464 -0.16172495  0.12800941
  0.07429419  0.10103885  0.08504409 -0.01794797 -0.06241613  0.14987893
  0.15474467  0.18398537 -0.17408288  0.13962157 -0.11823418  0.09919562
  0.07957372 -0.05181967  0.15559544  0.0681076  -0.0985308   0.02557893
 -0.11090399 -0.02128516 -0.01085772  0.11211726 -0.14611867  0.20995773
 -0.10311343  0.06910679  0.14604773  0.10655196  0.10023539 -0.02284993
```

```
0.14183174  0.13799591  0.00409749  0.11127966  0.21348046  0.03055387
  0.11364785 -0.1445034   0.11242675 -0.04190433]

Vector embedding for 'plaintiff':
[-0.03223411  0.06478627  0.00088969 -0.00806353  0.05694845 -0.21240263
  0.13640128  0.26523107 -0.13281158 -0.04770363 -0.02368818 -0.1402928
 -0.03685566  0.12257947  0.00039671  0.00741028 -0.01043882 -0.11464308
 -0.09540985 -0.3000543   0.0647751   0.00074026  0.00411286 -0.05273201
 -0.02684729 -0.04762366  0.02497391 -0.04300669 -0.04396778 -0.00184753
  0.14383827 -0.04924785  0.08860843 -0.08550214 -0.06152922  0.24551614
  0.10724474 -0.13455397 -0.05984696 -0.15700217  0.02755019 -0.14089336
 -0.07535081  0.0659988   0.11539416  0.020872   -0.05348673 -0.02727061
  0.01346072  0.03318129  0.09382757 -0.10529419  0.0414049   0.07656677
 -0.01830849  0.07164428  0.01196256  0.05545417 -0.13542365  0.1291954
  0.08052401  0.06550701  0.09594982 -0.03788032 -0.07346537  0.16846505
  0.13681169  0.14530386 -0.15170906  0.14640196 -0.09068518  0.0789521
  0.05557212 -0.02400086  0.11684093  0.06631403 -0.11164055  0.01440321
 -0.10535935 -0.00458972 -0.02664629  0.1090111  -0.12968238  0.18052402
 -0.09392222  0.08443088  0.12474449  0.09482376  0.11001488 -0.01367659
  0.12273199  0.1101999   0.02236929  0.09491293  0.19617565  0.01282949
  0.11568122 -0.1593218   0.10664962 -0.04113806]

Vector embedding for 'doctor':
[-3.76006439e-02  8.11468363e-02 -1.18198330e-02  1.22082625e-02
  5.35595044e-03 -2.21441105e-01  1.31108329e-01  3.10447901e-01
 -2.11071640e-01  7.52886664e-03 -6.67306557e-02 -1.76628768e-01
 -4.83631082e-02  1.88437983e-01 -2.80619003e-02  3.20329741e-02
 -2.19840016e-02 -1.36392176e-01 -1.02166705e-01 -3.58890593e-01
  4.39012572e-02  4.81801666e-03  1.11632412e-02 -6.98464885e-02
 -4.50425185e-02 -4.01994735e-02 -6.03534980e-04 -7.15099052e-02
 -7.36634061e-02  2.14629583e-02  2.10165456e-01 -6.25279024e-02
  1.19931854e-01 -1.26935437e-01 -8.21741298e-02  2.74210095e-01
  9.49538499e-02 -1.17289513e-01 -9.49264839e-02 -1.75545543e-01
  3.37264240e-02 -2.08480164e-01 -8.98559391e-02  1.35834515e-01
  1.21459514e-01  5.26671447e-02 -7.85357356e-02 -1.38883330e-02
  3.44770006e-03  5.95685691e-02  1.30519092e-01 -1.28386602e-01
  9.01534930e-02  7.31256530e-02 -1.94634255e-02  1.17376871e-01
  1.67697188e-04  4.33479100e-02 -1.57258630e-01  1.38467610e-01
  8.46170783e-02  7.77027458e-02  8.34437460e-02 -2.43678018e-02
 -8.29226896e-02  1.89361051e-01  1.67503580e-01  2.07188442e-01
 -1.92358971e-01  1.90954044e-01 -8.66395757e-02  8.63512680e-02
  8.16990361e-02 -2.30716318e-02  1.48350254e-01  9.33871120e-02
 -1.03444301e-01  3.32759172e-02 -1.03499167e-01  2.95007881e-02
 -4.18480560e-02  1.48850128e-01 -1.25358477e-01  2.33333096e-01
 -1.20942295e-01  1.06142171e-01  1.28692985e-01  1.23203449e-01
  1.00113675e-01 -1.41250789e-02  1.63177848e-01  1.50014937e-01
 -1.95683893e-02  1.19940504e-01  2.54336447e-01  2.12510210e-02
  1.35626718e-01 -1.89367294e-01  1.02768317e-01 -7.30541497e-02]

Vector embedding for 'patient':
[ 0.00135616  0.06625096  0.02714886 -0.03324671  0.05406597 -0.2076351
  0.14450136  0.27830392 -0.1474757  -0.05214735 -0.02860676 -0.218962
 -0.05803476  0.11022121 -0.03196976  0.0245685   0.0070367  -0.12605277
 -0.11396559 -0.3183468   0.07659787  0.01132763  0.00593386 -0.04407553
 -0.05708291 -0.05022431  0.03657781 -0.05108569 -0.0220301   0.00680075
  0.14817646 -0.03874053  0.13069744 -0.11300313 -0.10196024  0.2306353
  0.13352849 -0.12474146 -0.07811124 -0.14196448  0.03165774 -0.15317255
 -0.04029788  0.10843351  0.11978162  0.03644174 -0.07184896 -0.00125591
  0.01996329  0.04686815  0.12031849 -0.13361286  0.07784432  0.03898075
 -0.05535794  0.07788541  0.02375661  0.06319185 -0.13593689  0.13807625
  0.04011758  0.07736681  0.10920981 -0.01097703 -0.08413535  0.1694132
  0.1142689   0.17812304 -0.16391632  0.13841556 -0.08013699  0.09719803
  0.07872047 -0.04311903  0.14359443  0.06323478 -0.05998136  0.03068179
 -0.10644887  0.00854869 -0.04508544  0.13762434 -0.12336963  0.1855616
 -0.11391655  0.09752344  0.1405091   0.12214459  0.11253129 -0.01929942
  0.13898279  0.15566415  0.01292162  0.08838749  0.19901091  0.03416261
  0.12509196 -0.13636002  0.11566975 -0.02010318]

Vector embedding for 'guilty':
[-0.01413389  0.06656995 -0.00734866 -0.03095385  0.06509437 -0.2517697
  0.14954449  0.29895368 -0.15728544 -0.07182206 -0.06310162 -0.20050046
 -0.08547995  0.15693647 -0.0186175   0.01778842 -0.05446635 -0.12549472
 -0.11124176 -0.31952748  0.03580405  0.01365704  0.03395955 -0.03605738
 -0.06030127 -0.04814158  0.03859452 -0.09555041 -0.05513439  0.0372526
  0.19865839 -0.07835107  0.10888778 -0.11142128 -0.10577497  0.29005775
  0.11180676 -0.13126965 -0.07538164 -0.1596524   0.06402622 -0.17310387
 -0.09087672  0.04137763  0.09426072  0.02597058 -0.06627226 -0.02641308
  0.03379544  0.0561525   0.13159601 -0.16362782  0.08867155  0.10736878
 -0.04391972  0.10295371  0.04891674  0.00565069 -0.163432    0.08589575
  0.1108232   0.05997586  0.11241774 -0.04420831 -0.06642649  0.15975468
  0.1490166   0.12801382 -0.21193038  0.1502985  -0.10489336  0.09517636
  0.0673286  -0.03900745  0.15302955  0.0800889  -0.13577344  0.05731111
 -0.12092727  0.00424497 -0.00455176  0.11054221 -0.15298396  0.20722686
 -0.15278348  0.03610937  0.10936919  0.14354476  0.09363212 -0.00813364
  0.1714467   0.15730394 -0.02156785  0.11239511  0.24912179  0.03659537
  0.0892475  -0.202413    0.11249497 -0.05155509]



Vector embedding for 'surgery':
[-3.12990844e-02  6.58327192e-02  2.85430159e-03  1.10345073e-02
 -7.04743201e-03 -2.36223593e-01  1.33402810e-01  3.03116202e-01
 -2.05681935e-01  6.48758421e-03 -8.28733593e-02 -1.69779241e-01
 -5.81854694e-02  1.80510432e-01 -4.00698669e-02  3.47116366e-02
 -1.62971541e-02 -1.29537463e-01 -9.92213637e-02 -3.68670791e-01
  4.55319285e-02  8.06765445e-03 -1.78291200e-04 -6.00495152e-02
 -5.73267005e-02 -4.28762138e-02 -3.84912407e-03 -6.40033185e-02
 -7.08072856e-02  4.36537573e-03  2.26468816e-01 -4.98397388e-02
  1.30335823e-01 -1.16139121e-01 -8.42535719e-02  2.86336660e-01
  1.00505255e-01 -1.20256521e-01 -9.17292535e-02 -1.76113561e-01
  2.96843071e-02 -2.00398415e-01 -9.28441510e-02  1.45912632e-01
  1.11865871e-01  5.49624115e-02 -6.89490139e-02 -1.83873083e-02
 -1.00601949e-02  6.59109801e-02  1.25353217e-01 -1.26397550e-01
  9.62558836e-02  5.71697466e-02 -2.06405111e-02  1.16529934e-01
 -8.17977940e-04  2.92389747e-02 -1.62125885e-01  1.34710684e-01
  6.75722361e-02  8.40188041e-02  8.42126012e-02 -1.94504112e-02
 -1.00880139e-01  1.89215228e-01  1.60290688e-01  2.12331533e-01
 -2.03707144e-01  2.01542258e-01 -9.25249755e-02  9.14819315e-02
  8.59961137e-02 -2.71495730e-02  1.61703631e-01  9.22792554e-02
 -1.11497119e-01  5.09562343e-02 -1.00743666e-01  3.40460427e-02
 -5.15895225e-02  1.68939248e-01 -1.28210068e-01  2.49226272e-01
 -1.33621320e-01  1.16187118e-01  1.42963469e-01  1.47219375e-01
```

```
  1.09663606e-01  5.80039807e-03  1.60661057e-01  1.45263568e-01
 -1.83158442e-02  1.16535008e-01  2.47885883e-01  1.26237087e-02
  1.36337191e-01 -1.75651938e-01  1.01963326e-01 -7.20273107e-02]
```

```
# Visualization using PCA
selected_words = ["court", "plaintiff", "defendant", "guilty", "jury",
                  "patient", "doctor", "hospital", "surgery", "emergency"]
word_vectors = [domain_word2vec.wv[word] for word in selected_words]

word_vectors
```

Output:

```
[array([-0.00520213,  0.05436571,  0.0196009 ,  0.00766893,  0.04851889,
        -0.22194375,  0.15068555,  0.2671535 , -0.16717364, -0.04062838,
        -0.054865  , -0.17729442, -0.06285486,  0.16066416,  0.00799252,
         0.00430546, -0.04130681, -0.11852198, -0.11586928, -0.32001996,
         0.07377547,  0.00634967,  0.01555517, -0.04018658, -0.05180506,
        -0.06574838,  0.01809591, -0.04998898, -0.05094941,  0.00987862,
         0.17092119, -0.03111312,  0.12419216, -0.07877786, -0.07952873,
         0.22328345,  0.12608306, -0.0951244 , -0.07667849, -0.1501351 ,
         0.04725789, -0.15457962, -0.06896634,  0.13114625,  0.11142956,
         0.03642106, -0.06946036, -0.02198208,  0.01422113,  0.05933676,
         0.09983439, -0.12603386,  0.07056595,  0.02597529, -0.02668819,
         0.0757888 , -0.00033602,  0.05289464, -0.16172495,  0.12800941,
         0.07429419,  0.10103885,  0.08504409, -0.01794797, -0.06241613,
         0.14987893,  0.15474467,  0.18398537, -0.17408288,  0.13962157,
        -0.11823418,  0.09919562,  0.07957372, -0.05181967,  0.15559544,
         0.0681076 , -0.0985308 ,  0.02557893, -0.11090399, -0.02128516,
        -0.01085772,  0.11211726, -0.14611867,  0.20995773, -0.10311343,
         0.06910679,  0.14604773,  0.10655196,  0.10023539, -0.02284993,
         0.14183174,  0.13799591,  0.00409749,  0.11127966,  0.21348046,
         0.03055387,  0.11364785, -0.1445034 ,  0.11242675, -0.04190433],
       dtype=float32),
 array([-0.03223411,  0.06478627,  0.00088969, -0.00806353,  0.05694845,
        -0.21240263,  0.13640128,  0.26523107, -0.13281158, -0.04770363,
        -0.02368818, -0.1402928 , -0.03685566,  0.12257947,  0.00039671,
         0.00741028, -0.01043882, -0.11464308, -0.09540985, -0.3000543 ,
         0.0647751 ,  0.00074026,  0.00411286, -0.05273201, -0.02684729,
        -0.04762366,  0.02497391, -0.04300669, -0.04396778, -0.00184753,
         0.14383827, -0.04924785,  0.08860843, -0.08550214, -0.06152922,
         0.24551614,  0.10724474, -0.13455397, -0.05984696, -0.15700217,
         0.02755019, -0.14089336, -0.07535081,  0.0659988 ,  0.11539416,
         0.020872  , -0.05348673, -0.02727061,  0.01346072,  0.03318129,
         0.09382757, -0.10529419,  0.0414049 ,  0.07656677, -0.01830849,
         0.07164428,  0.01196256,  0.05545417, -0.13542365,  0.1291954 ,
         0.08052401,  0.06550701,  0.09594982, -0.03788032, -0.07346537,
         0.16846505,  0.13681169,  0.14530386, -0.15170906,  0.14640196,
        -0.09068518,  0.0789521 ,  0.05557212, -0.02400086,  0.11684093,
         0.06631403, -0.11164055,  0.01440321, -0.10535935, -0.00458972,
        -0.02664629,  0.1090111 , -0.12968238,  0.18052402, -0.09392222,
         0.08443088,  0.12474449,  0.09482376,  0.11001488, -0.01367659,
         0.12273199,  0.1101999 ,  0.02236929,  0.09491293,  0.19617565,
         0.01282949,  0.11568122, -0.1593218 ,  0.10664962, -0.04113806],
       dtype=float32),
 array([ 0.00656709,  0.07256435, -0.0084228 , -0.02586134,  0.07641555,
        -0.2732658 ,  0.1540303 ,  0.32865882, -0.17496191, -0.06661771,
        -0.06085587, -0.22411431, -0.08474998,  0.18789086, -0.02127556,
        0.03096173, -0.05577651, -0.12937057, -0.11135948, -0.36175218,
        0.04432205,  0.00878906,  0.02296932, -0.05328603, -0.07712711,
       -0.06075291,  0.04381331, -0.10575836, -0.06409874,  0.04152325,
        0.21431115, -0.08531993,  0.14578514, -0.11424538, -0.11725931,
        0.29418284,  0.10676998, -0.15401532, -0.09160217, -0.1664509 9,
        0.06118093, -0.19756706, -0.08612581,  0.06556768,  0.1085471 ,
        0.04415575, -0.06776308, -0.04802901,  0.04284215,  0.0638606 ,
        0.13356939, -0.17036071,  0.08767819,  0.10464148, -0.03167466,
        0.11624619,  0.03233933,  0.01407737, -0.15678538,  0.10963659,
        0.11469187,  0.07420997,  0.09665452, -0.03110271, -0.07621247,
        0.17188032,  0.18252161,  0.14339091, -0.22514871,  0.18041831,
       -0.12061799,  0.07872933,  0.06301736, -0.04593184,  0.16801536,
        0.08114434, -0.13756713,  0.06104114, -0.13863237,  0.01738747,
       -0.01658883,  0.13528904, -0.1735411 ,  0.24808215, -0.17541201,
        0.04516907,  0.11772847,  0.14438275,  0.12152614, -0.00914207,
        0.16980448,  0.15530077, -0.0296784 ,  0.13635741,  0.24644977,
        0.03516944,  0.11169897, -0.21215999,  0.10724142, -0.03329436],
      dtype=float32),
 array([-0.01413389,  0.06656995, -0.00734866, -0.03095385,  0.06509437,
       -0.2517697 ,  0.14954449,  0.29895368, -0.15728544, -0.07182206,
       -0.06310162, -0.20050046, -0.08547995,  0.15693647, -0.0186175 ,
        0.01778842, -0.05446635, -0.12549472, -0.11124176, -0.31952748,
        0.03580405,  0.01365704,  0.03395955, -0.03605738, -0.06030127,
       -0.04814158,  0.03859452, -0.09555041, -0.05513439,  0.0372526 ,
        0.19865839, -0.07835107,  0.10888778, -0.11142128, -0.10577497,
        0.29005775,  0.11180676, -0.13126965, -0.07538164, -0.1596524 ,
        0.06402622, -0.17310387, -0.09087672,  0.04137763,  0.09426072,
        0.02597058, -0.06627226, -0.02641308,  0.03379544,  0.0561525 ,
        0.13159601, -0.16362782,  0.08867155,  0.10736878, -0.04391972,
        0.10295371,  0.04891674,  0.00565069, -0.163432  ,  0.08589575,
        0.1108232 ,  0.05997586,  0.11241774, -0.04420831, -0.06642649,
        0.15975468,  0.1490166 ,  0.12801382, -0.21193038,  0.1502985 ,
       -0.10489336,  0.09517636,  0.0673286 , -0.03900745,  0.15302955,
        0.0800889 , -0.13577344,  0.05731111, -0.12092727,  0.00424497,
       -0.00455176,  0.11054221, -0.15298396,  0.20722686, -0.15278348,
        0.03610937,  0.10936919,  0.14354476,  0.09363212, -0.00813364,
        0.1714467 ,  0.15730394, -0.02156785,  0.11239511,  0.24912179,
        0.03659537,  0.0892475 , -0.202413  ,  0.11249497, -0.05155509],
      dtype=float32),
 array([-0.00793692,  0.04061861, -0.01272589, -0.0216382 ,  0.05832693,
       -0.20959468,  0.1335544 ,  0.23678838, -0.1236183 , -0.03556946,
       -0.02290245, -0.1449683 , -0.06467045,  0.1215817 , -0.01873406,
        0.01126286, -0.01548086, -0.10774158, -0.09506714, -0.26566857,
        0.04896098, -0.00084279,  0.01825006, -0.05337542, -0.03144738,
       -0.05701503,  0.03505556, -0.05904163, -0.04336127, -0.00061382,
        0.16488056, -0.05326047,  0.09773479, -0.07977082, -0.06476859,
        0.22621374,  0.09030687, -0.11633091, -0.06397521, -0.13930038,
        0.04200654, -0.13733749, -0.06120953,  0.06062739,  0.0891199 ,
        0.02058195, -0.06635275, -0.00753717,  0.01779128,  0.05422449,
        0.10981765, -0.1124575 ,  0.06291748,  0.08702539, -0.03922568,
        0.08803029,  0.03010272,  0.03673965, -0.13025954,  0.10378475,
        0.07686505,  0.06131725,  0.09677017, -0.02306653, -0.05867948,
        0.14983074,  0.11522046,  0.13194208, -0.16845842,  0.130707  ,
       -0.09688476,  0.0888531 ,  0.05799359, -0.03768088,  0.1355294 ,
        0.04766061, -0.1006932 ,  0.02873053, -0.10280664, -0.01355723,
       -0.0322897 ,  0.125441  , -0.13379173,  0.1888993 , -0.1090064 ,
        0.05351871,  0.11118538,  0.09934786,  0.08427987, -0.01916844,
        0.11553331,  0.12629525, -0.00194136,  0.09477089,  0.19319633,
        0.01517526,  0.08618706, -0.15499583,  0.09854038, -0.04697568],
      dtype=float32),
array([ 0.00135616,  0.06625096,  0.02714886, -0.03324671,  0.05406597,
       -0.2076351 ,  0.14450136,  0.27830392, -0.1474757 , -0.05214735,
       -0.02860676, -0.218962  , -0.05803476,  0.11022121, -0.03196976,
        0.0245685 ,  0.0070367 , -0.12605277, -0.11396559, -0.3183468 ,
        0.07659787,  0.01132763,  0.00593386, -0.04407553, -0.05708291,
       -0.05022431,  0.03657781, -0.05108569, -0.0220301 ,  0.00680075,
        0.14817646, -0.03874053,  0.13069744, -0.11300313, -0.10196024,
        0.2306353 ,  0.13352849, -0.12474146, -0.07811124, -0.14196448,
        0.03165774, -0.15317255, -0.04029788,  0.10843351,  0.11978162,
        0.03644174, -0.07184896, -0.00125591,  0.01996329,  0.04686815,
        0.12031849, -0.13361286,  0.07784432,  0.03898075, -0.05535794,
        0.07788541,  0.02375661,  0.06319185, -0.13593689,  0.13807625,
        0.04011758,  0.07736681,  0.10920981, -0.01097703, -0.08413535,
        0.1694132 ,  0.1142689 ,  0.17812304, -0.16391632,  0.13841556,
       -0.08013699,  0.09719803,  0.07872047, -0.04311903,  0.14359443,
        0.06323478, -0.05998136,  0.03068179, -0.10644887,  0.00854869,
       -0.04508544,  0.13762434, -0.12336963,  0.1855616 , -0.11391655,
        0.09752344,  0.1405091 ,  0.12214459,  0.11253129, -0.01929942,
        0.13898279,  0.15566415,  0.01292162,  0.08838749,  0.19901091,
        0.03416261,  0.12509196, -0.13636002,  0.11566975, -0.02010318],
      dtype=float32),
array([-3.76006439e-02,  8.11468363e-02, -1.18198330e-02,  1.22082625e-02,
        5.35595044e-03, -2.21441105e-01,  1.31108329e-01,  3.10447901e-01,
       -2.11071640e-01,  7.52886664e-03, -6.67306557e-02, -1.76628768e-01,
       -4.83631082e-02,  1.88437983e-01, -2.80619003e-02,  3.20329741e-02,
       -2.19840016e-02, -1.36392176e-01, -1.02166705e-01, -3.58890593e-01,
        4.39012572e-02,  4.81801666e-03,  1.11632412e-02, -6.98464885e-02,
       -4.50425185e-02, -4.01994735e-02, -6.03534980e-04, -7.15099052e-02,
       -7.36634061e-02,  2.14629583e-02,  2.10165456e-01, -6.25279024e-02,
        1.19931854e-01, -1.26935437e-01, -8.21741298e-02,  2.74210095e-01,
        9.49538499e-02, -1.17289513e-01, -9.49264839e-02, -1.75545543e-01,
        3.37264240e-02, -2.08480164e-01, -8.98559391e-02,  1.35834515e-01,
        1.21459514e-01,  5.26671447e-02, -7.85357356e-02, -1.38883330e-02,
        3.44770006e-03,  5.95685691e-02,  1.30519092e-01, -1.28386602e-01,
        9.01534930e-02,  7.31256530e-02, -1.94634255e-02,  1.17376871e-01,
        1.67697188e-04,  4.33479100e-02, -1.57258630e-01,  1.38467610e-01,
        8.46170783e-02,  7.77027458e-02,  8.34437460e-02, -2.43678018e-02,
       -8.29226896e-02,  1.89361051e-01,  1.67503580e-01,  2.07188442e-01,
       -1.92358971e-01,  1.90954044e-01, -8.66395757e-02,  8.63512680e-02,
        8.16990361e-02, -2.30716318e-02,  1.48350254e-01,  9.33871120e-02,
       -1.03444301e-01,  3.32759172e-02, -1.03499167e-01,  2.95007881e-02,
       -4.18480560e-02,  1.48850128e-01, -1.25358477e-01,  2.33333096e-01,
       -1.20942295e-01,  1.06142171e-01,  1.28692985e-01,  1.23203449e-01,
        1.00113675e-01, -1.41250789e-02,  1.63177848e-01,  1.50014937e-01,
       -1.95683893e-02,  1.19940504e-01,  2.54336447e-01,  2.12510210e-02,
        1.35626718e-01, -1.89367294e-01,  1.02768317e-01, -7.30541497e-02],
      dtype=float32),
array([-0.02265889,  0.05778723, -0.01179005, -0.00284239,  0.04882376,
       -0.23377152,  0.13176689,  0.3085964 , -0.16563822, -0.00594464,
       -0.06069683, -0.16251391, -0.04316762,  0.19339406, -0.00984798,
        0.00349556, -0.03423214, -0.15440044, -0.15881209, -0.36713216,
        0.0536154 , -0.02835809, -0.01769544, -0.04901092, -0.05845137,
       -0.06207271,  0.01982044, -0.07527879, -0.0646024 ,  0.04246312,
        0.17291646, -0.03934861,  0.15174052, -0.11018316, -0.1073219 ,
        0.25728288,  0.10440008, -0.15727909, -0.0763182 , -0.1741864 ,
        0.02225032, -0.2151592 , -0.09898599,  0.08515406,  0.14156315,
        0.04146844, -0.080073  , -0.02897993,  0.03221606,  0.05149914,
        0.13052906, -0.13760065,  0.0776631 ,  0.07678478, -0.00785946,
        0.10275181, -0.01308092,  0.06987558, -0.16492371,  0.15031946,
        0.07514932,  0.06996216,  0.08479748,  0.00843247, -0.08604642,
        0.19633524,  0.18013164,  0.14563482, -0.19318359,  0.18252854,
       -0.10180707,  0.07443867,  0.04677813, -0.07124216,  0.16272344,
        0.05405647, -0.0953992 ,  0.04183496, -0.10839124, -0.01641163,
       -0.00933946,  0.11554954, -0.13658553,  0.22137882, -0.13358647,
        0.06070266,  0.13194123,  0.11046906,  0.12901476, -0.01263191,
        0.16108233,  0.17001428,  0.02732584,  0.10106397,  0.21696223,
        0.00993073,  0.13067529, -0.19392748,  0.11318995, -0.02350748],
      dtype=float32),
array([-3.12990844e-02,  6.58327192e-02,  2.85430159e-03,  1.10345073e-02,
       -7.04743201e-03, -2.36223593e-01,  1.33402810e-01,  3.03116202e-01,
       -2.05681935e-01,  6.48758421e-03, -8.28733593e-02, -1.69779241e-01,
       -5.81854694e-02,  1.80510432e-01, -4.00698669e-02,  3.47116366e-02,
       -1.62971541e-02, -1.29537463e-01, -9.92213637e-02, -3.68670791e-01,
        4.55319285e-02,  8.06765445e-03, -1.78291200e-04, -6.00495152e-02,
       -5.73267005e-02, -4.28762138e-02, -3.84912407e-03, -6.40033185e-02,
       -7.08072856e-02,  4.36537573e-03,  2.26468816e-01, -4.98397388e-02,
        1.30335823e-01, -1.16139121e-01, -8.42535719e-02,  2.86336660e-01,
        1.00505255e-01, -1.20256521e-01, -9.17292535e-02, -1.76113561e-01,
        2.96843071e-02, -2.00398415e-01, -9.28441510e-02,  1.45912632e-01,
        1.11865871e-01,  5.49624115e-02, -6.89490139e-02, -1.83873083e-02,
       -1.00601949e-02,  6.59109801e-02,  1.25353217e-01, -1.26397550e-01,
        9.62558836e-02,  5.71697466e-02, -2.06405111e-02,  1.16529934e-01,
       -8.17977940e-04,  2.92389747e-02, -1.62125885e-01,  1.34710684e-01,
        6.75722361e-02,  8.40188041e-02,  8.42126012e-02, -1.94504112e-02,
       -1.00880139e-01,  1.89215228e-01,  1.60290688e-01,  2.12331533e-01,
       -2.03707144e-01,  2.01542258e-01, -9.25249755e-02,  9.14819315e-02,
        8.59961137e-02, -2.71495730e-02,  1.61703631e-01,  9.22792554e-02,
       -1.11497119e-01,  5.09562343e-02, -1.00743666e-01,  3.40460427e-02,
       -5.15895225e-02,  1.68939248e-01, -1.28210068e-01,  2.49226272e-01,
       -1.33621320e-01,  1.16187118e-01,  1.42963469e-01,  1.47219375e-01,
        1.09663606e-01,  5.80039807e-03,  1.60661057e-01,  1.45263568e-01,
       -1.83158442e-02,  1.16535008e-01,  2.47885883e-01,  1.26237087e-02,
        1.36337191e-01, -1.75651938e-01,  1.01963326e-01, -7.20273107e-02],
      dtype=float32),
array([-0.01135001,  0.05962004,  0.03272124, -0.01768819,  0.04998965,
       -0.20870571,  0.14685056,  0.2836007 , -0.18323691, -0.03109219,
       -0.03263121, -0.22973996, -0.04514541,  0.15115191, -0.02573079,
        0.00774679, -0.00233633, -0.1304845 , -0.14386515, -0.31636477,
        0.06311441,  0.00088838, -0.02058102, -0.03525062, -0.05741948,
       -0.08361816,  0.04948161, -0.06476792, -0.02724299,  0.00202177,
        0.18204965, -0.03211843,  0.15414716, -0.11622933, -0.12767726,
        0.25150353,  0.13327569, -0.16982682, -0.09414072, -0.16327456,
        0.01518478, -0.16644533, -0.0390787 ,  0.10758833,  0.1320721 ,
        0.01700985, -0.06482255, -0.01300713,  0.01951688,  0.04992113,
        0.12908201, -0.15448081,  0.05776305,  0.0506245 , -0.04097727,
        0.08827188,  0.02562736,  0.04502805, -0.13268901,  0.15970598,
        0.0415643 ,  0.10894849,  0.1138011 , -0.03124123, -0.09126465,
        0.17592564,  0.11290699,  0.18183088, -0.17974737,  0.15896654,
    -0.07602367,  0.10534283,  0.06435065, -0.05782789,  0.17650633,
     0.06925058, -0.07527371,  0.04818593, -0.11926682, -0.00753901,
    -0.03426384,  0.13999003, -0.1504484 ,  0.19165526, -0.14357014,
     0.09853069,  0.1328372 ,  0.13577645,  0.12897931, -0.0323601 ,
     0.11762244,  0.17293827, -0.00854158,  0.07778574,  0.22550419,
     0.03845395,  0.1585336 , -0.14676552,  0.14424598, -0.03719835],
   dtype=float32)]
```

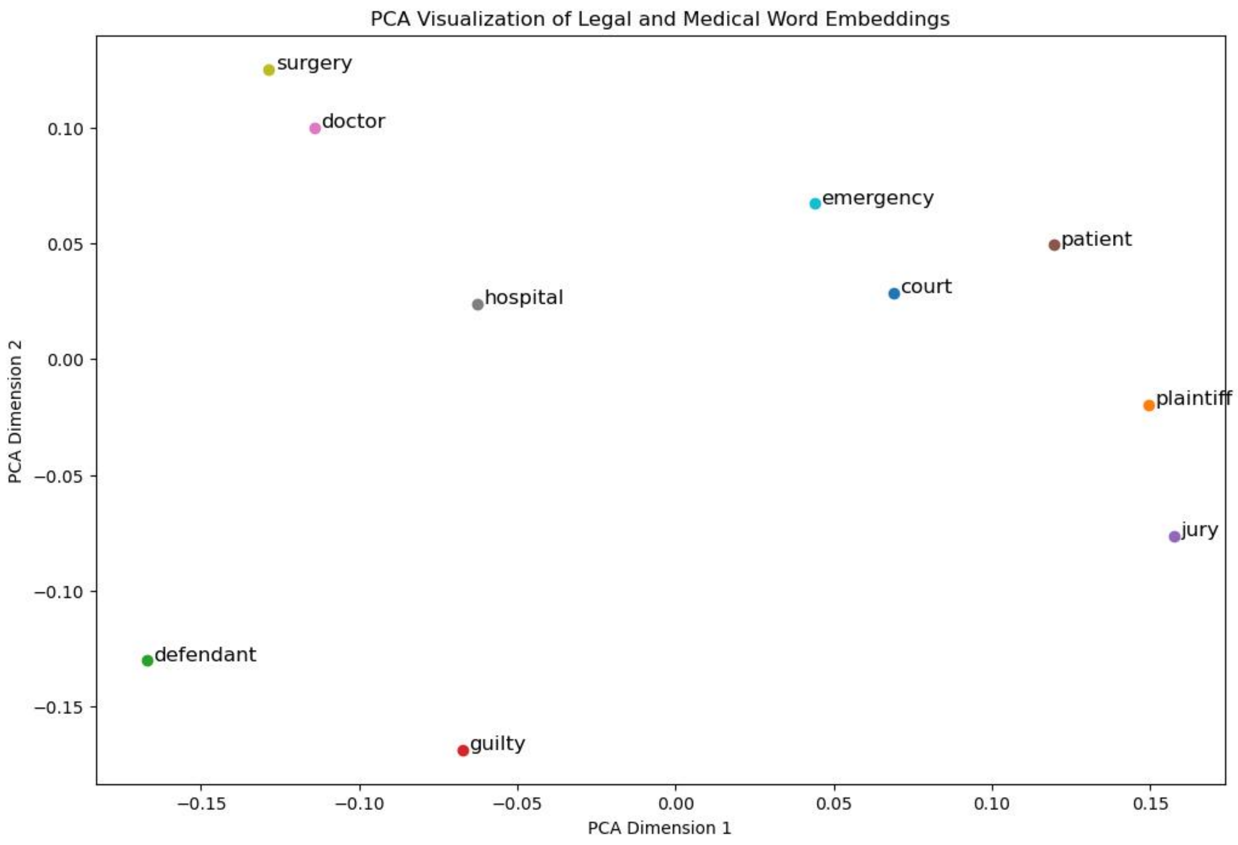
```
pca = PCA(n_components=2)
reduced_vectors = pca.fit_transform(word_vectors)

reduced_vectors
```

Output:

```
array([[ 0.06908131,  0.0287464 ],
       [ 0.14953919, -0.01964696],
       [-0.16693931, -0.13033459],
       [-0.0674262 , -0.16882876],
       [ 0.1574409 , -0.07644414],
       [ 0.11961383,  0.04956438],
       [-0.1142199 ,  0.10011148],
       [-0.06259862,  0.0240761 ],
       [-0.12848931,  0.12535115],
       [ 0.04399811,  0.06740492]
```

```
plt.figure(figsize=(12, 8))
for i, word in enumerate(selected_words):
plt.scatter(reduced_vectors[i, 0], reduced_vectors[i, 1])
plt.text(reduced_vectors[i, 0] + 0.002, reduced_vectors[i, 1], word, fontsize=12)
plt.title("PCA Visualization of Legal and Medical Word Embeddings")
plt.xlabel("PCA Dimension 1")
plt.ylabel("PCA Dimension 2")
plt.show()
```

Output:



Program 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

Theory:

This program demonstrates how word embeddings can be used to improve Generative AI responses by replacing specific words in a prompt with their most semantically similar words. The enriched prompt is then compared with the original prompt in terms of

response detail and relevance.

The program utilizes three main concepts:

1.1 Word Embeddings (GloVe)

Word embeddings convert words into vector representations such that similar words have

closer vector distances in the embedding space.

The model glove-wiki-gigaword-100 provides 100-dimensional word vectors pre-trained

on Wikipedia and Gigaword data.

1.2 Generative AI Model (GPT-2)

GPT-2 (a Transformer-based model) is a Generative AI model capable of autocompleting

text.

It predicts the next word in a sequence based on context.

The model is pre-trained on large internet-based datasets.

It generates text using context from the input prompt.

1.3 Prompt Enrichment with Word2Vec

The program replaces a user-defined keyword with its most semantically similar word

(found using word embeddings).

This helps enhance the prompt and produce a more detailed and meaningful AI-generated

response.

Steps Involved in Execution

Step 1: Import Necessary Libraries

```
pip install transformers –U
```

Output:

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: transformers in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (4.48.3)
Requirement already satisfied: filelock in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from transformers) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from transformers) (0.27.1)
Requirement already satisfied: numpy>=1.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from transformers) (1.24.4)
Requirement already satisfied: packaging>=20.0 in c:\users\anitha d
```

```
b\appdata\roaming\python\python311\site-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers) (0.5.2)
Requirement already satisfied: tqdm>=4.27 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from huggingface-hub<1.0,>=0.24.0->transformers) (2024.12.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from huggingface-hub<1.0,>=0.24.0->transformers) (4.12.2)
Requirement already satisfied: colorama in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->transformers) (2024.12.14)
Note: you may need to restart the kernel to use updated packages.
```

```python
from gensim.scripts.glove2word2vec import glove2word2vec
from gensim.models import KeyedVectors

# Paths to the GloVe file and output Word2Vec file
glove_input_file = "glove.6B/glove.6B.100d.txt"  # Path to GloVe file
word2vec_output_file = "glove.6B/glove.6B.100d.word2vec.txt"  # Output file in Word2Vec format

# Convert GloVe format to Word2Vec format
glove2word2vec(glove_input_file, word2vec_output_file)

# Load the converted Word2Vec model
model = KeyedVectors.load_word2vec_format(word2vec_output_file, binary=False)

# Test the loaded model
print(model.most_similar("king"))
```

output:

```
C:\Users\ANITHA D B\AppData\Local\Temp\ipykernel_13104\2083156905.py:9: DeprecationWarning:
Call to deprecated `glove2word2vec` (KeyedVectors.load_word2vec_format(.., binary=False,
no_header=True) loads GLoVE text vectors.).
  glove2word2vec(glove_input_file, word2vec_output_file)
[('prince', 0.7682328820228577), ('queen', 0.7507690787315369), ('son', 0.7020888328552246), ('brother',
0.6985775232315063), ('monarch', 0.6977890729904175), ('throne', 0.6919989585876465), ('kingdom',
0.6811409592628479), ('father', 0.6802029013633728), ('emperor', 0.6712858080863953), ('ii',
0.6676074266433716)]
```

```
# Define the original medical prompt
original_prompt = "Explain the importance of vaccinations in healthcare."

# Define key terms extracted from the original prompt
key_terms = ["vaccinations", "healthcare"]

# Initialize an empty list to store similar terms
similar_terms = []

# Loop through each key term to find similar words
for term in key_terms:

    # Check if the key term exists in the vocabulary of the 'model' (word embedding model)
    # Assuming 'model.key_to_index' is a way to check for term existence in the model's vocabulary
    if term in model.key_to_index:

        # If the term exists, find the top 3 most similar words using 'model.most_similar(term, topn=3)'
        # and extend the 'similar_terms' list with these words.
        # Assuming 'model.most_similar' returns a list of tuples, where each tuple is (word, similarity_score)

        # We are extracting only the 'word' part using a set comprehension for potential deduplication.
        similar_terms.extend({word for word, _ in model.most_similar(term, topn=3)})

# Enrich the original prompt with the retrieved similar terms
if similar_terms:
    # If similar terms were found, create an enriched prompt by appending
    # "Consider aspects like: " followed by a comma-separated string of similar terms.
    enriched_prompt = f"{original_prompt} Consider aspects like: {', '.join(similar_terms)}."
else:
    # If no similar terms were found, the enriched prompt is the same as the original prompt.
    enriched_prompt = original_prompt

# Output the original and enriched prompts
print("Original Prompt:", original_prompt)
print("Enriched Prompt:", enriched_prompt)
```

output:

```
Original Prompt: Explain the importance of vaccinations in healthcare.
Enriched Prompt: Explain the importance of vaccinations in healthcare. Consider aspects like: vaccination,
inoculations, immunizations, care, health, services.
```

```python
import getpass
import os
GOOGLE_API_KEY= os.environ["GOOGLE_API_KEY"] = getpass.getpass("Enter your Google AI API key: ")
```

```
Enter your Google AI API key: ········
```

```python
from langchain_google_genai import ChatGoogleGenerativeAI

llm = ChatGoogleGenerativeAI(
    model="gemini-2.0-flash-exp",
    temperature=0,
    api_key=GOOGLE_API_KEY,
    max_tokens=256,
    timeout=None,
    max_retries=2,
    # other params...
)
```

```python
llm.invoke("Hi")
```

Output:
```
AIMessage(content='Hi there! How can I help you today?', additional_kwargs={},
response_metadata={'prompt_feedback': {'block_reason': 0, 'safety_ratings': []}, 'finish_reason': 'STOP',
'safety_ratings': []}, id='run-fe0c40aa-6482-42ff-97f2-6e2a8d6ce594-0', usage_metadata={'input_tokens': 1,
'output_tokens': 11, 'total_tokens': 12, 'input_token_details': {'cache_read': 0}})
```

```python
print(llm.invoke(original_prompt).content)
```

Output
```
Vaccinations are a cornerstone of modern healthcare and play a vital role in protecting individuals and
communities from infectious diseases. Their importance can be summarized in several key areas:
```

```
**1. Disease Prevention and Eradication:**

*   **Individual Protection:** Vaccines work by exposing the body to a weakened or inactive form of a
disease-causing agent (virus or bacteria). This triggers the immune system to produce antibodies, which
provide protection against future infections.  If the individual is later exposed to the real disease, their
immune system is primed to fight it off quickly and effectively, often preventing illness or significantly
reducing its severity.
*   **Herd Immunity:** When a large percentage of a population is vaccinated, it creates "herd immunity."
This means that even those who cannot be vaccinated (e.g., infants too young, individuals with certain
medical conditions) are protected because the disease has difficulty spreading. Herd immunity is crucial for
protecting vulnerable populations.
*   **Disease Eradication/Elimination:**  Vaccination campaigns have successfully eradicated diseases like
smallpox and have significantly reduced the incidence of others, such as polio and measles.  Continued
vaccination efforts are essential to maintain these achievements and prevent the resurgence of these diseases.

**2. Reduced Morbidity and Mortality:**
```

Program 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.

Theory:

How It Works:

1. Loads Pre-trained Word Embeddings: Uses the glove-wiki-gigaword-100 model.
2. Retrieves Similar Words: Finds the top 5 similar words for a given seed word.
3. Generates Sentences: Uses sentence templates to create meaningful lines.
4. Constructs a Paragraph: Combines multiple generated sentences into a short creative paragraph.

Try entering different seed words like "adventure", "mystery", or "ocean" and see how the story changes!

Introduction

Word embeddings are a powerful representation of words in a continuous vector space, allowing them to capture semantic relationships between words. This program leverages pre-trained word embeddings to generate creative sentences and construct a paragraph from a given seed word.

What the Program Does?

1. Takes a Seed Word → A user provides a single word (e.g., "adventure").
2. Retrieves Similar Words → Finds top N words that are semantically related.
3. Constructs Sentences → Uses these words to form meaningful sentences.
4. Creates a Short Paragraph → Combines the generated sentences into a creative story.

Key Concepts Used

1. Word Embeddings

Word embeddings transform words into vector representations that capture semantic similarity. This allows words with similar meanings to be close to each other in the vector space.

2. Pre-trained Word Embeddings

The program uses GloVe (Global Vectors for Word Representation), specifically the "glove-wiki-gigaword-100" model, which has been trained on a massive dataset and provides 100-dimensional word vectors.

3. Finding Similar Words

To retrieve similar words, we use:

word_vectors.most_similar(seed_word, topn=5)

This finds the top 5 words closest to the given word in the vector space.

4. Constructing Meaningful Sentences
The program uses sentence templates to insert the retrieved words into coherent sentences.
Example sentence structure:
"The [word] led to an unexpected twist in the story."
If the word is "adventure", the sentence becomes:
"The adventure led to an unexpected twist in the story."
5 Generating a Paragraph
Once multiple sentences are generated, they are combined logically to form a short paragraph.
Example Output:
The journey led to an unexpected twist in the story. The exploration revealed hidden secrets of
This program demonstrates how word embeddings can be used for creative text generation by leveraging semantic relationships between words. It automates the process of brainstorming ideas, making it useful for: Storytelling

- Creative Writing
- Idea Generation for Writers

!pip install sentence_transformers

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: sentence_transformers in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (3.4.0)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from sentence_transformers) (4.48.3)
Requirement already satisfied: tqdm in c:\users\anitha d b\appdata\roaming\python\python311\site-packages
(from sentence_transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from sentence_transformers) (2.5.1)
Requirement already satisfied: scikit-learn in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from sentence_transformers) (1.6.1)
Requirement already satisfied: scipy in c:\users\anitha d b\appdata\roaming\python\python311\site-packages
(from sentence_transformers) (1.10.1)
Requirement already satisfied: huggingface-hub>=0.20.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from sentence_transformers) (0.27.1)
Requirement already satisfied: Pillow in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from sentence_transformers) (11.1.0)
Requirement already satisfied: filelock in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from huggingface-hub>=0.20.0->sentence_transformers) (3.17.0)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.20.0-
>sentence_transformers) (2024.12.0)
Requirement already satisfied: packaging>=20.9 in c:\users\anitha d
```

```
b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.20.0->sentence_transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.20.0->sentence_transformers) (6.0.2)
Requirement already satisfied: requests in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.20.0->sentence_transformers) (2.32.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.20.0->sentence_transformers) (4.12.2)
Requirement already satisfied: networkx in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from torch>=1.11.0->sentence_transformers) (3.4.2)
Requirement already satisfied: jinja2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from torch>=1.11.0->sentence_transformers) (3.1.5)
Requirement already satisfied: sympy==1.13.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from torch>=1.11.0->sentence_transformers) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from sympy==1.13.1->torch>=1.11.0->sentence_transformers) (1.3.0)
Requirement already satisfied: colorama in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tqdm->sentence_transformers) (0.4.6)
Requirement already satisfied: numpy>=1.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers<5.0.0,>=4.41.0->sentence_transformers) (1.24.4)
Requirement already satisfied: regex!=2019.12.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers<5.0.0,>=4.41.0->sentence_transformers) (2024.11.6)
Requirement already satisfied: tokenizers<0.22,>=0.21 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers<5.0.0,>=4.41.0->sentence_transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers<5.0.0,>=4.41.0->sentence_transformers) (0.5.2)
Requirement already satisfied: joblib>=1.2.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from scikit-learn->sentence_transformers) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from scikit-learn->sentence_transformers) (3.5.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from jinja2->torch>=1.11.0->sentence_transformers) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.20.0->sentence_transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.20.0->sentence_transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.20.0->sentence_transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.20.0->sentence_transformers) (2024.12.14)
```

!pip install langchain-huggingface

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: langchain-huggingface in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (0.1.2)
Requirement already satisfied: huggingface-hub>=0.23.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-huggingface) (0.27.1)
Requirement already satisfied: langchain-core<0.4.0,>=0.3.15 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-huggingface) (0.3.32)
Requirement already satisfied: sentence-transformers>=2.6.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-huggingface) (3.4.0)
Requirement already satisfied: tokenizers>=0.19.1 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-huggingface) (0.21.0)
Requirement already satisfied: transformers>=4.39.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-huggingface) (4.48.3)
Requirement already satisfied: filelock in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from huggingface-hub>=0.23.0->langchain-huggingface) (3.17.0)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.23.0->langchain-
huggingface) (2024.12.0)
Requirement already satisfied: packaging>=20.9 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.23.0->langchain-
huggingface) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from huggingface-hub>=0.23.0->langchain-huggingface) (6.0.2)
Requirement already satisfied: requests in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from huggingface-hub>=0.23.0->langchain-huggingface) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from huggingface-hub>=0.23.0->langchain-huggingface) (4.67.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from huggingface-hub>=0.23.0->langchain-
huggingface) (4.12.2)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-core<0.4.0,>=0.3.15->langchain-
huggingface) (1.33)
Requirement already satisfied: langsmith<0.4,>=0.1.125 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-core<0.4.0,>=0.3.15->langchain-
huggingface) (0.3.2)
Requirement already satisfied: pydantic<3.0.0,>=2.5.2 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-core<0.4.0,>=0.3.15->langchain-
huggingface) (2.10.6)
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from langchain-core<0.4.0,>=0.3.15->langchain-
huggingface) (9.0.0)
Requirement already satisfied: torch>=1.11.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from sentence-transformers>=2.6.0->langchain-huggingface) (2.5.1)
Requirement already satisfied: scikit-learn in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from sentence-transformers>=2.6.0->langchain-huggingface) (1.6.1)
Requirement already satisfied: scipy in c:\users\anitha d b\appdata\roaming\python\python311\site-packages
(from sentence-transformers>=2.6.0->langchain-huggingface) (1.10.1)
Requirement already satisfied: Pillow in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from sentence-transformers>=2.6.0->langchain-huggingface) (11.1.0)
```

```
Requirement already satisfied: numpy>=1.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers>=4.39.0->langchain-huggingface) (1.24.4)
Requirement already satisfied: regex!=2019.12.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers>=4.39.0->langchain-huggingface) (2024.11.6)
Requirement already satisfied: safetensors>=0.4.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from transformers>=4.39.0->langchain-huggingface) (0.5.2)
Requirement already satisfied: jsonpointer>=1.9 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from jsonpatch<2.0,>=1.33->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (3.0.0)
Requirement already satisfied: httpx<1,>=0.23.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (0.28.1)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (3.10.15)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (1.0.0)
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (0.23.0)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from pydantic<3.0.0,>=2.5.2->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (0.7.0)
Requirement already satisfied: pydantic-core==2.27.2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from pydantic<3.0.0,>=2.5.2->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (2.27.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.23.0->langchain-huggingface) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.23.0->langchain-huggingface) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.23.0->langchain-huggingface) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests->huggingface-hub>=0.23.0->langchain-huggingface) (2024.12.14)
Requirement already satisfied: networkx in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from torch>=1.11.0->sentence-transformers>=2.6.0->langchain-huggingface) (3.4.2)
Requirement already satisfied: jinja2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from torch>=1.11.0->sentence-transformers>=2.6.0->langchain-huggingface) (3.1.5)
Requirement already satisfied: sympy==1.13.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from torch>=1.11.0->sentence-transformers>=2.6.0->langchain-huggingface) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from sympy==1.13.1->torch>=1.11.0->sentence-transformers>=2.6.0->langchain-huggingface) (1.3.0)
Requirement already satisfied: colorama in c:\users\anitha d b\appdata\roaming\python\python311\site-
```

```
packages (from tqdm>=4.42.1->huggingface-hub>=0.23.0->langchain-huggingface) (0.4.6)
Requirement already satisfied: joblib>=1.2.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from scikit-learn->sentence-transformers>=2.6.0->langchain-huggingface) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from scikit-learn->sentence-transformers>=2.6.0->langchain-huggingface) (3.5.0)
Requirement already satisfied: anyio in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (4.8.0)
Requirement already satisfied: httpcore==1.* in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from httpcore==1.*->httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (0.14.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from jinja2->torch>=1.11.0->sentence-transformers>=2.6.0->langchain-huggingface) (3.0.2)
Requirement already satisfied: sniffio>=1.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from anyio->httpx<1,>=0.23.0->langsmith<0.4,>=0.1.125->langchain-core<0.4.0,>=0.3.15->langchain-huggingface) (1.3.1)
```

!pip install tf-keras –user

```
Requirement already satisfied: tf-keras in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (2.18.0)
Requirement already satisfied: tensorflow<2.19,>=2.18 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tf-keras) (2.18.0)
Requirement already satisfied: tensorflow-intel==2.18.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow<2.19,>=2.18->tf-keras) (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (25.1.24)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (3.4.0)
Requirement already satisfied: packaging in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (24.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (5.29.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (2.32.3)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-packages (from tensorflow-
intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (68.2.2)
Requirement already satisfied: six>=1.12.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (1.70.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.8.0)
Collecting numpy<2.1.0,>=1.26.0 (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras)
  Downloading numpy-2.0.2-cp311-cp311-win_amd64.whl.metadata (59 kB)
Requirement already satisfied: h5py>=3.11.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.12.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (0.31.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (0.45.1)
Requirement already satisfied: rich in c:\users\anitha d b\appdata\roaming\python\python311\site-packages
(from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (13.9.4)
Requirement already satisfied: namex in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (0.0.8)
Requirement already satisfied: optree in c:\users\anitha d b\appdata\roaming\python\python311\site-
packages (from keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (0.14.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\anitha d
b\appdata\roaming\python\python311\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0-
>tensorflow<2.19,>=2.18->tf-keras) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from requests<3,>=2.21.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (2024.12.14)
Requirement already satisfied: markdown>=2.6.8 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\anitha d b\appdata\roaming\python\python311\site-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow-intel==2.18.0->tensorflow<2.19,>=2.18->tf-keras) (0.1.2)
Downloading numpy-2.0.2-cp311-cp311-win_amd64.whl (15.9 MB)
   ---------------------------------------- 0.0/15.9 MB ? eta -:--:--
    --------------------------------------- 0.3/15.9 MB ? eta -:--:--
   - -------------------------------------- 0.5/15.9 MB 2.1 MB/s eta 0:00:08
   -- ------------------------------------- 1.0/15.9 MB 2.2 MB/s eta 0:00:07
   --- ------------------------------------ 1.6/15.9 MB 2.2 MB/s eta 0:00:07
   ---- ----------------------------------- 1.8/15.9 MB 2.0 MB/s eta 0:00:07
   ----- ---------------------------------- 2.4/15.9 MB 2.0 MB/s eta 0:00:07
   ------ --------------------------------- 2.6/15.9 MB 2.0 MB/s eta 0:00:07
   ------- -------------------------------- 3.1/15.9 MB 2.0 MB/s eta 0:00:07
   --------- ------------------------------ 3.7/15.9 MB 2.0 MB/s eta 0:00:07
   ---------- ----------------------------- 4.2/15.9 MB 2.0 MB/s eta 0:00:06
   ----------- ---------------------------- 4.5/15.9 MB 2.0 MB/s eta 0:00:06
   ------------ --------------------------- 5.0/15.9 MB 2.0 MB/s eta 0:00:06
   ------------ --------------------------- 5.0/15.9 MB 2.0 MB/s eta 0:00:06
   ------------- -------------------------- 5.5/15.9 MB 1.9 MB/s eta 0:00:06
   ------------- -------------------------- 5.5/15.9 MB 1.9 MB/s eta 0:00:06
   ------------- -------------------------- 5.5/15.9 MB 1.9 MB/s eta 0:00:06
   --------------- ------------------------ 6.0/15.9 MB 1.7 MB/s eta 0:00:06
   --------------- ------------------------ 6.3/15.9 MB 1.7 MB/s eta 0:00:06
   ---------------- ----------------------- 6.6/15.9 MB 1.6 MB/s eta 0:00:06
```

```
  ----------------- ---------------------- 6.8/15.9 MB 1.6 MB/s eta 0:00:06
  ----------------- ---------------------- 7.1/15.9 MB 1.6 MB/s eta 0:00:06
  ----------------- ---------------------- 7.1/15.9 MB 1.6 MB/s eta 0:00:06
  ------------------ --------------------- 7.3/15.9 MB 1.5 MB/s eta 0:00:06
  ------------------- -------------------- 7.6/15.9 MB 1.5 MB/s eta 0:00:06
  -------------------- ------------------- 8.1/15.9 MB 1.5 MB/s eta 0:00:06
  -------------------- ------------------- 8.1/15.9 MB 1.5 MB/s eta 0:00:06
  -------------------- ------------------- 8.1/15.9 MB 1.5 MB/s eta 0:00:06
  --------------------- ------------------ 8.4/15.9 MB 1.4 MB/s eta 0:00:06
  --------------------- ------------------ 8.7/15.9 MB 1.4 MB/s eta 0:00:06
  ---------------------- ----------------- 8.9/15.9 MB 1.4 MB/s eta 0:00:06
  ----------------------- ---------------- 9.2/15.9 MB 1.4 MB/s eta 0:00:05
  ----------------------- ---------------- 9.4/15.9 MB 1.4 MB/s eta 0:00:05
  ------------------------ --------------- 9.7/15.9 MB 1.4 MB/s eta 0:00:05
  ------------------------ --------------- 9.7/15.9 MB 1.4 MB/s eta 0:00:05
  ------------------------- -------------- 10.0/15.9 MB 1.3 MB/s eta 0:00:05
  ------------------------- -------------- 10.2/15.9 MB 1.3 MB/s eta 0:00:05
  -------------------------- ------------- 10.7/15.9 MB 1.3 MB/s eta 0:00:04
  --------------------------- ------------ 11.0/15.9 MB 1.4 MB/s eta 0:00:04
  ---------------------------- ----------- 11.3/15.9 MB 1.3 MB/s eta 0:00:04
  ---------------------------- ----------- 11.5/15.9 MB 1.3 MB/s eta 0:00:04
  ----------------------------- ---------- 11.8/15.9 MB 1.4 MB/s eta 0:00:04
  ------------------------------ --------- 12.1/15.9 MB 1.3 MB/s eta 0:00:03
  ------------------------------ --------- 12.3/15.9 MB 1.3 MB/s eta 0:00:03
  ------------------------------- -------- 12.6/15.9 MB 1.3 MB/s eta 0:00:03
  -------------------------------- ------- 13.1/15.9 MB 1.3 MB/s eta 0:00:03
  --------------------------------- ------ 13.4/15.9 MB 1.4 MB/s eta 0:00:02
  ---------------------------------- ----- 13.9/15.9 MB 1.4 MB/s eta 0:00:02
  ---------------------------------- ----- 13.9/15.9 MB 1.4 MB/s eta 0:00:02
  ------------------------------------ --- 14.4/15.9 MB 1.4 MB/s eta 0:00:02
  ------------------------------------ --- 14.7/15.9 MB 1.4 MB/s eta 0:00:01
  ------------------------------------- -- 14.9/15.9 MB 1.4 MB/s eta 0:00:01
  -------------------------------------- - 15.5/15.9 MB 1.4 MB/s eta 0:00:01
  ---------------------------------------  15.7/15.9 MB 1.4 MB/s eta 0:00:01
  ---------------------------------------- 15.9/15.9 MB 1.4 MB/s eta 0:00:00
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.24.4
    Uninstalling numpy-1.24.4:
      Successfully uninstalled numpy-1.24.4
Successfully installed numpy-2.0.2
  WARNING: The scripts f2py.exe and numpy-config.exe are installed in 'C:\Users\ANITHA D
B\AppData\Roaming\Python\Python311\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-
location.
  WARNING: Failed to remove contents in a temporary directory 'C:\Users\ANITHA D
B\AppData\Roaming\Python\Python311\site-packages\~-mpy'.
  You can safely remove it manually.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed.
This behaviour is the source of the following dependency conflicts.
langchain 0.3.16 requires numpy<2,>=1.22.4; python_version < "3.12", but you have numpy 2.0.2 which is
```

```
incompatible.
langchain-community 0.3.16 requires numpy<2,>=1.22.4; python_version < "3.12", but you have numpy 2.0.2 which is incompatible.
pyfume 0.3.4 requires numpy==1.24.4, but you have numpy 2.0.2 which is incompatible.
scipy 1.10.1 requires numpy<1.27.0,>=1.19.5, but you have numpy 2.0.2 which is incompatible.
astropy 5.3.4 requires numpy<2,>=1.21, but you have numpy 2.0.2 which is incompatible.
contourpy 1.2.0 requires numpy<2.0,>=1.20, but you have numpy 2.0.2 which is incompatible.
matplotlib 3.8.0 requires numpy<2,>=1.21, but you have numpy 2.0.2 which is incompatible.
numba 0.59.0 requires numpy<1.27,>=1.22, but you have numpy 2.0.2 which is incompatible.
pywavelets 1.5.0 requires numpy<2.0,>=1.22.4, but you have numpy 2.0.2 which is incompatible.
streamlit 1.30.0 requires numpy<2,>=1.19.3, but you have numpy 2.0.2 which is incompatible.
streamlit 1.30.0 requires packaging<24,>=16.8, but you have packaging 24.2 which is incompatible.
streamlit 1.30.0 requires pillow<11,>=7.1.0, but you have pillow 11.1.0 which is incompatible.
streamlit 1.30.0 requires protobuf<5,>=3.20, but you have protobuf 5.29.3 which is incompatible.
streamlit 1.30.0 requires tenacity<9,>=8.1.0, but you have tenacity 9.0.0 which is incompatible.
```

!pip install numpy==1.24.4 –user

```
Collecting numpy==1.24.4  WARNING: The script f2py.exe is installed in 'C:\Users\ANITHA D B\AppData\Roaming\Python\Python311\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
tensorflow-intel 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.24.4 which is incompatible.
streamlit 1.30.0 requires packaging<24,>=16.8, but you have packaging 24.2 which is incompatible.
streamlit 1.30.0 requires pillow<11,>=7.1.0, but you have pillow 11.1.0 which is incompatible.
streamlit 1.30.0 requires protobuf<5,>=3.20, but you have protobuf 5.29.3 which is incompatible.
streamlit 1.30.0 requires tenacity<9,>=8.1.0, but you have tenacity 9.0.0 which is incompatible.

  Using cached numpy-1.24.4-cp311-cp311-win_amd64.whl.metadata (5.6 kB)
Using cached numpy-1.24.4-cp311-cp311-win_amd64.whl (14.8 MB)
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
Successfully installed numpy-1.24.4
```

```python
from sentence_transformers import SentenceTransformer, util

# Load a pretrained SentenceTransformer model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Define an expanded finance-related corpus
corpus = [
    "The stock market saw significant gains today, driven by strong earnings reports.",
    "Investing in diversified portfolios helps mitigate risk and maximize returns.",
    "The Federal Reserve's decision to raise interest rates could impact market liquidity.",
    "Cryptocurrency has become an increasingly popular asset class among investors.",
    "Financial analysts predict that the global economy will face challenges in the coming years.",
    "Bonds are considered a safer investment option compared to stocks.",
    "Banks are adopting blockchain technology to improve the efficiency of financial transactions.",
    "The economic impact of the pandemic has been severe, but recovery is underway.",
    "Inflation rates have been rising steadily, leading to higher costs for consumers.",
    "Corporate governance and transparency are crucial for investor confidence.",
    "The real estate market is experiencing a boom as demand outstrips supply in many areas.",
    "Investors should be aware of market volatility and adjust their strategies accordingly.",
    "Diversification is a key principle in reducing risk in investment portfolios.",
    "Hedge funds use complex strategies to generate high returns, often with higher risks.",
    "Stock buybacks are often seen as a sign of confidence by corporate executives."
]

# Encode the corpus into embeddings
corpus_embeddings = model.encode(corpus, convert_to_tensor=True)
corpus_embeddings
```

```
WARNING:tensorflow:From C:\Users\ANITHA D B\AppData\Roaming\Python\Python311\site-packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

```
modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]
```

```
C:\Users\ANITHA D B\AppData\Roaming\Python\Python311\site-packages\huggingface_hub\file_download.py:140: UserWarning: `huggingface_hub` cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\ANITHA D B\.cache\huggingface\hub\models--sentence-transformers--all-MiniLM-L6-v2. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
```

```
modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]
```

```
C:\Users\ANITHA D B\AppData\Roaming\Python\Python311\site-packages\huggingface_hub\file_download.py:140: UserWarning: `huggingface_hub` cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\ANITHA D B\.cache\huggingface\hub\models--sentence-transformers--all-MiniLM-L6-v2. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
```

```
# Function to generate a story using contextual embeddings
def generate_paragraph(seed_word, corpus, corpus_embeddings, model, top_n=5):

    # Encode the seed word as a sentence
    seed_sentence = f"Tell me more about {seed_word} in finance."
    seed_embedding = model.encode(seed_sentence, convert_to_tensor=True)

    # Find the most similar sentences in the corpus to the seed sentence
    similarities = util.pytorch_cos_sim(seed_embedding, corpus_embeddings)[0]
    top_results = similarities.topk(top_n)
    print('top_results:',top_results)

    # Construct a more coherent story using the most similar sentences
    story = f"The topic of '{seed_word}' is crucial in the finance industry. "

    for idx in top_results.indices:
        similar_sentence = corpus[idx]
        story += f"{similar_sentence} "

    story += f"These concepts highlight the importance of {seed_word} in understanding financial
markets and investment strategies."
    return story
# Example usage
seed_word = "bonds"
story = generate_paragraph(seed_word, corpus, corpus_embeddings, model, top_n=5)
print(story)
```

```
top_results: torch.return_types.topk(
values=tensor([0.6597, 0.4536, 0.4218, 0.4031, 0.3689]),
indices=tensor([ 5,  3,  2, 13,  1]))
The topic of 'bonds' is crucial in the finance industry. Bonds are considered a safer investment option
compared to stocks. Cryptocurrency has become an increasingly popular asset class among investors. The
Federal Reserve's decision to raise interest rates could impact market liquidity. Hedge funds use complex
strategies to generate high returns, often with higher risks. Investing in diversified portfolios helps mitigate
risk and maximize returns. These concepts highlight the importance of bonds in understanding financial
markets and investment strategies.
```

```
#!pip install langchain-huggingface
```

Program 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.

Theory:
This program analyzes sentiment in text using a pre-trained Hugging Face model. It is particularly useful in real-world applications such as customer feedback analysis, social media monitoring, and product review evaluation.

Key Concepts Involved

1 Sentiment Analysis

Sentiment analysis is a Natural Language Processing (NLP) technique used to determine the emotional tone of a given text. It classifies text as positive, negative, or neutral.
Example:

"This product is amazing!" → Positive ✓
"I'm very disappointed with the service." → Negative ✗
"It was an average experience." → Neutral

Transformers and Pre-trained Models

This program leverages Transformers, a powerful deep-learning-based NLP framework by Hugging Face. Specifically, it uses a pre-trained model for sentiment analysis, which means:

No need for manual training.
Leverages large-scale datasets used in training.
Fast and efficient inference.

3 Pipeline in Hugging Face

The pipeline("sentiment-analysis") function loads a pre-trained model and applies it directly to text input. The underlying model is typically DistilBERT, BERT, or RoBERTa, which are transformer-based architectures designed for NLP tasks.

Consider a customer feedback analysis system where user reviews are processed to determine sentiment. This could be useful for businesses to track customer satisfaction.
Following are the review statements:

"The product quality is amazing! I'm very satisfied.",
"I had a terrible experience with customer service.",
"The delivery was quick, but the packaging was damaged.",
"Absolutely love this! Best purchase I've made.",
"Not worth the money, very disappointed."

```
# Approach 1: Using Transformers Pipeline
#%pip install --upgrade --quiet huggingface_hub
#%pip install --upgrade langchain
        from transformers import pipeline

# Load the sentiment analysis pipeline
        sentiment_analyzer = pipeline("sentiment-analysis")
```

```
# Example sentences for analysis
    sentences = [
        "The product quality is amazing! I'm very satisfied.",
        "I had a terrible experience with customer service.",
        "The delivery was quick, but the packaging was damaged.",
        "Absolutely love this! Best purchase I've made.",
        "Not worth the money, very disappointed."
    ]

# Analyze sentiment for each sentence
        results = sentiment_analyzer(sentences)

# Print the results
  for sentence, result in zip(sentences, results):
  print(f"Sentence: {sentence}\nSentiment: {result['label']}, Confidence: {result['score']:.2f}\n")
```

Output:

```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english and revision
714eb0f (https://huggingface.co/distilbert/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
C:\Users\Dell\anaconda3\lib\site-packages\huggingface_hub\file_download.py:140: UserWarning:
`huggingface_hub` cache-system uses symlinks by default to efficiently store duplicated files but your
machine does not support them in C:\Users\Dell\.cache\huggingface\hub\models--distilbert--distilbert-base-
uncased-finetuned-sst-2-english. Caching files will still work but in a degraded version that might require
more space on your disk. This warning can be disabled by setting the
`HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details, see
https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an
administrator. In order to activate developer mode, see this article: https://docs.microsoft.com/en-
us/windows/apps/get-started/enable-your-device-for-development
  warnings.warn(message)
WARNING:tensorflow:From C:\Users\Dell\AppData\Roaming\Python\Python39\site-
packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated.
Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
model.safetensors:   0%|          | 0.00/268M [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]
Sentence: The product quality is amazing! I'm very satisfied.
Sentiment: POSITIVE, Confidence: 1.00

Sentence: I had a terrible experience with customer service.
Sentiment: NEGATIVE, Confidence: 1.00

Sentence: The delivery was quick, but the packaging was damaged.
Sentiment: NEGATIVE, Confidence: 1.00

Sentence: Absolutely love this! Best purchase I've made.
Sentiment: POSITIVE, Confidence: 1.00
```

```
Sentence: Not worth the money, very disappointed.
Sentiment: NEGATIVE, Confidence: 1.00
Results
```

Output:

```
[{'label': 'POSITIVE', 'score': 0.9998825788497925},
 {'label': 'NEGATIVE', 'score': 0.9993104934692383},
 {'label': 'NEGATIVE', 'score': 0.9997345805168152},
 {'label': 'POSITIVE', 'score': 0.9998751878738403},
 {'label': 'NEGATIVE', 'score': 0.9998034834861755}]
```

### Approach 2: Using API calls

```
from langchain_huggingface import HuggingFaceEndpoint

# get a token: https://huggingface.co/docs/api-inference/quicktour#get-your-api-token

from getpass import getpass

HUGGINGFACEHUB_API_TOKEN = getpass()
........
import os

os.environ["HUGGINGFACEHUB_API_TOKEN"] = HUGGINGFACEHUB_API_TOKEN
from langchain.chains import LLMChain
from langchain_core.prompts import PromptTemplate
text = ["The product quality is amazing! I'm very satisfied.",
   "I had a terrible experience with customer service.",
   "The delivery was quick, but the packaging was damaged.",
   "Absolutely love this! Best purchase I've made.",
   "Not worth the money, very disappointed."]

template = """Perform the sentiment analysis for the following:{text}.

Answer: Following is the sentiment for the given text:"""

prompt = PromptTemplate.from_template(template)
repo_id = "meta-llama/Llama-3.2-3B-Instruct" #"mistralai/Mistral-7B-Instruct-v0.2"

llm = HuggingFaceEndpoint(
   repo_id=repo_id,
   max_length=256,
   temperature=0.5,
   huggingfacehub_api_token=HUGGINGFACEHUB_API_TOKEN,
)
llm_chain = prompt | llm
print(llm_chain.invoke({"text": text}))
```

```
WARNING! max_length is not default parameter.
                max_length was transferred to model_kwargs.
                Please make sure that max_length is what you intended.
 ['positive', 'negative', 'negative', 'positive', 'negative'].
```

Explanation:
1. The first sentence is positive because it mentions that the product quality is amazing and the speaker is very satisfied.
2. The second sentence is negative because it mentions a terrible experience with customer service.
3. The third sentence is negative because the delivery was quick, but the packaging was damaged.
4. The fourth sentence is positive because the speaker loves the product and thinks it's the best purchase they've made.
5. The fifth sentence is negative because the speaker is very disappointed with the product.

Program 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.

Theory:

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.
Summarize long texts using a pre-trained Hugging Face model. It utilizes the summarization pipeline to take a passage as input and generate a concise summary.
Here's a Python program that runs in Jupyter Notebook to summarize long texts using a pre-trained Hugging Face model. It utilizes the summarization pipeline to take a passage as input and generate a concise summary.

🎯 Real-World Applications

✅ News Article Summarization – Quickly summarize lengthy news reports.

✅ Research Paper Summarization – Extract key insights from academic papers.

✅ Customer Support Summarization – Condense long conversations into key takeaways.

✅ Legal Document Summarization – Summarize lengthy contracts and legal papers.

The summarizer function in Hugging Face's transformers library is part of the pipeline API, which simplifies the process of using pre-trained NLP models. It allows users to generate summaries of long text passages.

Text for Summarization : AI In Education

Artificial Intelligence (AI) is transforming education by introducing adaptive learning techniques, automating administrative processes, and enabling intelligent tutoring systems. AI-driven learning platforms analyze vast amounts of student data, including learning habits, strengths, and weaknesses, to personalize educational experiences. This customization allows students to progress at their own pace, ensuring that they receive content suited to their proficiency level.

Additionally, AI chatbots and virtual assistants are becoming common in academic institutions, providing real-time support to students. These tools answer frequently asked questions, guide students through complex topics, and help with scheduling and reminders. Educators also benefit from AI-powered grading systems that assess assignments, quizzes, and exams, significantly reducing workload and providing instant feedback.

Moreover, AI enhances accessibility in education by offering language translation services, speech-to-text conversion, and assistive technologies for students with disabilities. By breaking language barriers and supporting diverse learning needs, AI makes education more inclusive.

However, challenges remain in implementing AI in education. Data privacy concerns arise as student information is collected and analyzed, requiring robust security measures. There is also the risk of AI biases, where algorithmic decisions may favor certain groups over others due to biased training data. Additionally, educators must undergo proper training to integrate AI effectively into their teaching methods.

To fully harness AI's potential in education, institutions must adopt ethical AI frameworks, ensure transparency in algorithmic decision-making, and continuously update their technological infrastructure. Collaboration between educators, policymakers, and AI developers is crucial in shaping the future of education and ensuring that AI serves as an enabler rather than a disruptor.

```
from transformers import pipeline

# Load the summarization pipeline
summarizer = pipeline("summarization")

# Expanded input passage
text = """
Artificial Intelligence (AI) is transforming education by introducing adaptive learning techniques, automating administrative processes, and enabling intelligent tutoring systems.
AI-driven learning platforms analyze vast amounts of student data, including learning habits, strengths, and weaknesses, to personalize educational experiences.
This customization allows students to progress at their own pace, ensuring that they receive content suited to their proficiency level.

Additionally, AI chatbots and virtual assistants are becoming common in academic institutions, providing real-time support to students.
These tools answer frequently asked questions, guide students through complex topics, and help with scheduling and reminders.
Educators also benefit from AI-powered grading systems that assess assignments, quizzes, and exams, significantly reducing workload and providing instant feedback.

Moreover, AI enhances accessibility in education by offering language translation services, speech-to-text conversion, and assistive technologies for students with disabilities.
By breaking language barriers and supporting diverse learning needs, AI makes education more inclusive.

However, challenges remain in implementing AI in education. Data privacy concerns arise as student information is collected and analyzed, requiring robust security measures.
There is also the risk of AI biases, where algorithmic decisions may favor certain groups over others due to biased training data.
Additionally, educators must undergo proper training to integrate AI effectively into their teaching methods.

To fully harness AI’s potential in education, institutions must adopt ethical AI frameworks, ensure transparency in algorithmic decision-making, and continuously update their technological infrastructure.
Collaboration between educators, policymakers, and AI developers is crucial in shaping the future of education and ensuring that AI serves as an enabler rather than a disruptor.
"""

# Generate the summary with longer output
summary = summarizer(long_text, max_length=100, min_length=50, do_sample=False)

# Print the summarized text
print("Summarized Text:\n", summary[0]['summary_text'])
```

Output:

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e
(https://huggingface.co/sshleifer/distilbart-cnn-12-6).
Using a pipeline without specifying a model name and revision in production is not recommended.
Device set to use cpu
Summarized Text:
 Artificial Intelligence (AI) is transforming education by introducing adaptive learning techniques,
automating administrative processes, and enabling intelligent tutoring systems . AI chatbots and
virtual assistants are becoming common in academic institutions, providing real-time support to
students . Data privacy concerns arise as student information is collected and analyzed, requiring
robust security measures .
```

The Transformative Role of Artificial Intelligence in Modern Education

Artificial Intelligence (AI) has emerged as a cornerstone of innovation in education, fundamentally reshaping how knowledge is delivered, personalized, and assessed. As institutions increasingly integrate AI into their pedagogical frameworks, the impact extends beyond automation to the creation of intelligent learning environments that foster engagement, accessibility, and efficiency.

One of the most profound contributions of AI to education is **adaptive learning**, a paradigm that leverages data-driven insights to customize educational content for individual students. Unlike traditional one-size-fits-all approaches, AI-powered platforms analyze student performance, learning patterns, and cognitive preferences to adjust the difficulty level, pace, and mode of instruction in real-time. This ensures that students who struggle with certain concepts receive targeted reinforcement, while advanced learners can progress without unnecessary repetition.

Intelligent tutoring systems (ITS) represent another significant advancement, providing students with **personalized, AI-driven guidance** outside of traditional classroom settings. These systems, built on natural language processing and machine learning, simulate human tutors by offering step-by-step explanations, identifying gaps in understanding, and adapting instructional methods accordingly. AI tutors are particularly valuable in disciplines such as mathematics, science, and language learning, where real-time feedback and iterative problem-solving are crucial to mastery.

Beyond individualized learning, AI enhances **collaborative education** by fostering interactive, technology-driven experiences. Virtual reality (VR) and augmented reality (AR) applications, powered by AI algorithms, create **immersive simulations** that enable students to explore historical events, conduct virtual science experiments, and engage in role-based learning. These innovations bridge the gap between theoretical knowledge and practical application, making complex concepts more tangible and accessible.

AI also plays a critical role in **automating administrative functions**, thereby allowing educators to allocate more time to teaching and mentorship. Automated grading systems can evaluate assignments, quizzes, and even subjective responses with increasing accuracy, while AI-driven scheduling tools streamline academic operations. Additionally, AI chatbots and virtual assistants handle routine queries from students, reducing response times and improving administrative efficiency.

One of the most significant yet underexplored benefits of AI in education is its potential to **enhance accessibility and inclusivity**. Speech-to-text and text-to-speech technologies enable students with disabilities to engage with learning materials more effectively. AI-driven translation services remove language barriers, allowing students from diverse linguistic backgrounds to access high-quality educational content. Moreover, AI-powered predictive analytics can identify students at risk of falling behind, enabling early interventions to prevent academic disengagement.

Despite these advantages, AI's integration into education is not without challenges. **Ethical concerns surrounding data privacy, bias in AI algorithms, and the digital divide must be addressed** to ensure equitable access to AI-driven education. Institutions must adopt **transparent AI governance policies**, emphasizing accountability and inclusivity in algorithmic decision-making. Additionally, educators must be equipped with the necessary training to effectively implement AI tools within their instructional practices, ensuring that technology serves as an enabler rather than a disruptor.

As AI continues to evolve, its role in education will extend beyond content delivery to **fostering critical thinking, creativity, and problem-solving skills**. The future of education lies not in replacing human educators but in **augmenting their capabilities**, enabling a more **engaging, efficient, and personalized** learning experience for students worldwide. By striking a balance between technological innovation and ethical responsibility, AI has the potential to **democratize education and bridge learning gaps on a global scale**.

```
from langchain_huggingface import HuggingFaceEndpoint

# get a token: https://huggingface.co/docs/api-inference/quicktour#get-your-api-token

from getpass import getpass

HUGGINGFACEHUB_API_TOKEN = getpass()
········
import os

os.environ["HUGGINGFACEHUB_API_TOKEN"] = HUGGINGFACEHUB_API_TOKEN
```

text = f"""Artificial Intelligence (AI) has emerged as a cornerstone of innovation in education, fundamentally reshaping how knowledge is delivered, personalized, and assessed. As institutions increasingly integrate AI into their pedagogical frameworks, the impact extends beyond automation to the creation of intelligent learning environments that foster engagement, accessibility, and efficiency.

One of the most profound contributions of AI to education is adaptive learning, a paradigm that leverages data-driven insights to customize educational content for individual students. Unlike traditional one-size-fits-all approaches, AI-powered platforms analyze student performance, learning patterns, and cognitive preferences to adjust the difficulty level, pace, and mode of instruction in real-time. This ensures that students who struggle with certain concepts receive targeted reinforcement, while advanced learners can progress without unnecessary repetition.

Intelligent tutoring systems (ITS) represent another significant advancement, providing students with personalized, AI-driven guidance outside of traditional classroom settings. These systems, built on natural language processing and machine learning, simulate human tutors by offering step-by-step explanations, identifying gaps in understanding, and adapting instructional methods accordingly. AI tutors are particularly valuable in disciplines such as mathematics, science, and language learning, where real-time feedback and iterative problem-solving are crucial to mastery.

Beyond individualized learning, AI enhances collaborative education by fostering interactive, technology-driven experiences. Virtual reality (VR) and augmented reality (AR) applications, powered by AI algorithms, create immersive simulations that enable students to explore historical events, conduct virtual science experiments, and engage in role-based learning. These innovations bridge the gap between theoretical knowledge and practical application, making complex concepts more tangible and accessible.

AI also plays a critical role in automating administrative functions, thereby allowing educators to allocate more time to teaching and mentorship. Automated grading systems can evaluate assignments, quizzes, and even subjective responses with increasing accuracy, while AI-driven scheduling tools streamline academic operations. Additionally, AI chatbots and virtual assistants handle routine queries from students, reducing response times and improving administrative efficiency.

One of the most significant yet underexplored benefits of AI in education is its potential to enhance accessibility and inclusivity. Speech-to-text and text-to-speech technologies enable students with disabilities to engage with learning materials more effectively. AI-driven translation services remove language barriers, allowing students from diverse linguistic backgrounds to access high-quality educational content. Moreover, AI-powered predictive analytics can identify students at risk of falling behind, enabling early interventions to prevent academic disengagement.

Despite these advantages, AI's integration into education is not without challenges. Ethical concerns surrounding data privacy, bias in AI algorithms, and the digital divide must be addressed to ensure equitable access to AI-driven education. Institutions must adopt transparent AI governance policies, emphasizing accountability and inclusivity in algorithmic decision-making. Additionally, educators must be equipped with the necessary training to effectively implement AI tools within their instructional practices, ensuring that technology serves as an enabler rather than a disruptor.

As AI continues to evolve, its role in education will extend beyond content delivery to fostering critical thinking, creativity, and problem-solving skills. The future of education lies not in replacing human educators but in augmenting their capabilities, enabling a more engaging, efficient, and personalized learning experience for students worldwide. By striking a balance between technological innovation and ethical responsibility, AI has the potential to democratize education and bridge learning gaps on a global scale.

```
"""

import requests

API_URL = "https://api-inference.huggingface.co/models/facebook/bart-large-cnn"
headers = {"Authorization": "Bearer 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"}

def query(payload):
   response = requests.post(API_URL, headers=headers, json=payload)
   return response.json()
output= query("inputs": text)  # Remove the curly braces
```

Output:

```
[{'summary_text': 'Artificial Intelligence (AI) has emerged as a cornerstone of innovation in education. As in
stitutions increasingly integrate AI into their pedagogical frameworks, the impact extends beyond automatio
n to the creation of intelligent learning environments. The future of education lies not in replacing human ed
ucators but in augmenting their capabilities, enabling a more engaging, efficient, and personalized learning e
xperience.'}]
```

Input cell:

```
output[0]['summary_text']
```

Output:

```
'Artificial Intelligence (AI) has emerged as a cornerstone of innovation in education. As institutions increasi
ngly integrate AI into their pedagogical frameworks, the impact extends beyond automation to the creation o
f intelligent learning environments. The future of education lies not in replacing human educators but in aug
menting their capabilities, enabling a more engaging, efficient, and personalized learning experience.'
```

Program 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.

Theory:

What is Cohere?
Cohere is an AI-powered Natural Language Processing (NLP) platform that provides large language models (LLMs) for text generation, classification, summarization, search, and embedding. It is similar to OpenAI's GPT models but focuses on enterprise-level AI applications, offering scalable APIs for developers.

Key Features of Cohere
1. Text Generation – Generate human-like text for chatbots, creative writing, content generation, and more.
2. Text Summarization – Extract key points from long documents or articles.
3. Text Classification – Automatically categorize text into different labels.
4. Semantic Search – Improve search accuracy by understanding the meaning of queries.
5. Embeddings API – Convert words, sentences, or documents into numerical representations for machine learning applications.
6. Custom Models – Fine-tune models based on specific business needs.

Cohere vs Other LLMs (Like OpenAI)

Feature	Cohere	OpenAI (GPT)
Focus	Enterprise NLP	General NLP
API Speed	Fast & Optimized	Powerful but may be slower
Model Customization	Supports fine-tuning	Limited fine-tuning
Security & Privacy	More control for businesses	API-based access
Best For	Companies integrating AI	Chatbots & general AI

How Does Cohere Work?
Cohere provides an API-based service where users can send text input to a pre-trained model and receive AI-generated responses.
1. Sign up on Cohere's platform
2. Get an API Key from the Cohere dashboard
3. Use the API for various NLP tasks (text generation, summarization, classification, embeddings, etc.)

Cohere Models
Cohere provides different LLMs for various tasks:
Command – Best for instruction-following and structured responses.
Generate – Used for creative writing and text expansion.
Embed – Converts text into vector embeddings for semantic search.

Use Cases of Cohere
✅ Chatbots & Virtual Assistants
✅ Automated Content Writing (blog articles, marketing copy)
✅ Text Summarization (news articles, legal documents)
✅ Customer Support Automation (analyze customer queries)

✅ Semantic Search & Information Retrieval

Final Thoughts

Cohere is a powerful NLP tool for businesses and developers looking for custom AI solutions. It provides an API-first approach, making it easy to integrate AI into real-world applications like chatbots, summarization, and document analysis.

● If you're working on an NLP-based project, Cohere is a great alternative to OpenAI's GPT!

This program demonstrates how to integrate LangChain with Cohere's large language models (LLMs) to perform text summarization, key takeaway extraction, and sentiment analysis on a document loaded from Google Drive. Here's a breakdown of the key concepts and technologies used:

1. LangChain Framework

LangChain is an open-source framework designed to simplify the development of applications using Large Language Models (LLMs). It enables easy integration with different LLM providers like Cohere, OpenAI, and others, and helps in chaining multiple AI-driven components like prompt templates, memory, and agents for building complex applications.

PromptTemplate: LangChain provides a utility to create structured prompts. The PromptTemplate class allows you to define dynamic prompts where parts of the text can be replaced with actual inputs (like the document text in this case).

2. Cohere Language Models

Cohere provides powerful NLP APIs for tasks like text generation, summarization, and classification using their Command model family. This program uses Cohere's Command model to process and analyze the text document.

API Integration: The program requires an API Key for accessing Cohere's services. The key is securely entered using getpass.getpass() to prevent exposing sensitive information.

Model Used: command: This is a general-purpose model optimized for tasks like summarization, question answering, and content generation.

3. Google Colab and Google Drive Integration

The program is designed to run in Google Colab, which provides a cloud-based environment for running Python code with free access to GPUs.

Authentication:

o auth.authenticate_user(): Authenticates your Google account to allow access to your Google Drive files.

o drive.mount('/content/drive'): Mounts Google Drive so you can directly access and manipulate files from Colab.

File Loading:

The program reads a .txt file from Google Drive using Python's built-in file handling (open() function).

4. Prompt Engineering

Prompt engineering is the practice of designing effective prompts to guide an LLM to generate desired outputs. The program builds a multi-task prompt that asks the model to:

1. Summarize the document: Generate a concise summary.
2. List key takeaways: Highlight 3 important points.
3. Perform sentiment analysis: Identify whether the document's tone is positive, negative, or neutral.

#### Step 1: Run the following command to install the necessary libraries:

Output:

!pip install langchain langchain-cohere langchain-community

```
Requirement already satisfied: langchain in c:\users\admin\anaconda3\lib\site-packages (0.3.17)
Collecting langchain-cohere
  Downloading langchain_cohere-0.4.2-py3-none-any.whl (42 kB)
Requirement already satisfied: langchain-community in c:\users\admin\anaconda3\lib\site-packages (0.3.16)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in c:\users\admin\anaconda3\lib\site-packages (from
langchain) (1.4.32)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in c:\users\admin\anaconda3\lib\site-packages
(from langchain) (4.0.1)
Requirement already satisfied: langsmith<0.4,>=0.1.17 in c:\users\admin\anaconda3\lib\site-packages (from
langchain) (0.3.4)
Requirement already satisfied: numpy<2,>=1.22.4 in c:\users\admin\appdata\roaming\python\python39\site-
packages (from langchain) (1.24.4)
Requirement already satisfied: langchain-text-splitters<0.4.0,>=0.3.3 in c:\users\admin\anaconda3\lib\site-
packages (from langchain) (0.3.5)
Requirement already satisfied: PyYAML>=5.3 in c:\users\admin\anaconda3\lib\site-packages (from
langchain) (6.0)
Requirement already satisfied: langchain-core<0.4.0,>=0.3.33 in c:\users\admin\anaconda3\lib\site-packages
(from langchain) (0.3.33)
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in c:\users\admin\anaconda3\lib\site-packages (from
langchain) (2.10.6)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in c:\users\admin\anaconda3\lib\site-packages (from
langchain) (3.11.11)
Requirement already satisfied: requests<3,>=2 in c:\users\admin\anaconda3\lib\site-packages (from
langchain) (2.27.1)
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in c:\users\admin\anaconda3\lib\site-packages
(from langchain) (9.0.0)
Collecting types-pyyaml<7.0.0.0,>=6.0.12.20240917
  Downloading types_PyYAML-6.0.12.20241230-py3-none-any.whl (20 kB)
Collecting cohere<6.0,>=5.12.0
  Downloading cohere-5.13.12-py3-none-any.whl (252 kB)
Requirement already satisfied: httpx-sse<0.5.0,>=0.4.0 in c:\users\admin\anaconda3\lib\site-packages (from
langchain-community) (0.4.0)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in c:\users\admin\anaconda3\lib\site-packages
(from langchain-community) (0.6.7)
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in c:\users\admin\anaconda3\lib\site-
packages (from langchain-community) (2.7.1)
Requirement already satisfied: attrs>=17.3.0 in c:\users\admin\anaconda3\lib\site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (21.4.0)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\admin\anaconda3\lib\site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (5.1.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in c:\users\admin\anaconda3\lib\site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (1.18.3)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\admin\anaconda3\lib\site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (1.2.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in c:\users\admin\anaconda3\lib\site-packages
```

```
(from aiohttp<4.0.0,>=3.8.3->langchain) (2.4.4)
Requirement already satisfied: propcache>=0.2.0 in c:\users\admin\anaconda3\lib\site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (0.2.1)
Requirement already satisfied: aiosignal>=1.1.2 in c:\users\admin\anaconda3\lib\site-packages (from
aiohttp<4.0.0,>=3.8.3->langchain) (1.2.0)
Requirement already satisfied: typing-extensions>=3.6.5 in c:\users\admin\anaconda3\lib\site-packages
(from async-timeout<5.0.0,>=4.0.0->langchain) (4.12.2)
Requirement already satisfied: httpx>=0.21.2 in c:\users\admin\anaconda3\lib\site-packages (from
cohere<6.0,>=5.12.0->langchain-cohere) (0.28.1)
Requirement already satisfied: pydantic-core<3.0.0,>=2.18.2 in c:\users\admin\anaconda3\lib\site-packages
(from cohere<6.0,>=5.12.0->langchain-cohere) (2.27.2)
Collecting fastavro<2.0.0,>=1.9.4
  Downloading fastavro-1.10.0-cp39-cp39-win_amd64.whl (546 kB)
Requirement already satisfied: tokenizers<1,>=0.15 in c:\users\admin\anaconda3\lib\site-packages (from
cohere<6.0,>=5.12.0->langchain-cohere) (0.21.0)
Collecting types-requests<3.0.0,>=2.0.0
  Downloading types_requests-2.32.0.20241016-py3-none-any.whl (15 kB)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in c:\users\admin\anaconda3\lib\site-packages
(from dataclasses-json<0.7,>=0.5.7->langchain-community) (3.26.0)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in c:\users\admin\anaconda3\lib\site-packages
(from dataclasses-json<0.7,>=0.5.7->langchain-community) (0.9.0)
Requirement already satisfied: anyio in c:\users\admin\anaconda3\lib\site-packages (from httpx>=0.21.2-
>cohere<6.0,>=5.12.0->langchain-cohere) (3.5.0)
Requirement already satisfied: certifi in c:\users\admin\anaconda3\lib\site-packages (from httpx>=0.21.2-
>cohere<6.0,>=5.12.0->langchain-cohere) (2021.10.8)
Requirement already satisfied: idna in c:\users\admin\anaconda3\lib\site-packages (from httpx>=0.21.2-
>cohere<6.0,>=5.12.0->langchain-cohere) (3.3)
Requirement already satisfied: httpcore==1.* in c:\users\admin\anaconda3\lib\site-packages (from
httpx>=0.21.2->cohere<6.0,>=5.12.0->langchain-cohere) (1.0.7)
Requirement already satisfied: h11<0.15,>=0.13 in c:\users\admin\anaconda3\lib\site-packages (from
httpcore==1.*->httpx>=0.21.2->cohere<6.0,>=5.12.0->langchain-cohere) (0.14.0)
Requirement already satisfied: packaging<25,>=23.2 in c:\users\admin\anaconda3\lib\site-packages (from
langchain-core<0.4.0,>=0.3.33->langchain) (24.2)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in c:\users\admin\anaconda3\lib\site-packages (from
langchain-core<0.4.0,>=0.3.33->langchain) (1.33)
Requirement already satisfied: jsonpointer>=1.9 in c:\users\admin\anaconda3\lib\site-packages (from
jsonpatch<2.0,>=1.33->langchain-core<0.4.0,>=0.3.33->langchain) (3.0.0)
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in c:\users\admin\anaconda3\lib\site-packages
(from langsmith<0.4,>=0.1.17->langchain) (0.23.0)
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in c:\users\admin\anaconda3\lib\site-
packages (from langsmith<0.4,>=0.1.17->langchain) (1.0.0)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in c:\users\admin\anaconda3\lib\site-packages (from
langsmith<0.4,>=0.1.17->langchain) (3.10.15)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\admin\anaconda3\lib\site-packages (from
pydantic<3.0.0,>=2.7.4->langchain) (0.7.0)
Requirement already satisfied: python-dotenv>=0.21.0 in c:\users\admin\anaconda3\lib\site-packages (from
pydantic-settings<3.0.0,>=2.4.0->langchain-community) (1.0.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\admin\anaconda3\lib\site-packages
(from requests<3,>=2->langchain) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\admin\anaconda3\lib\site-packages (from
requests<3,>=2->langchain) (1.26.9)
```

```
Requirement already satisfied: greenlet!=0.4.17 in c:\users\admin\anaconda3\lib\site-packages (from
SQLAlchemy<3,>=1.4->langchain) (1.1.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in c:\users\admin\anaconda3\lib\site-packages
(from tokenizers<1,>=0.15->cohere<6.0,>=5.12.0->langchain-cohere) (0.28.1)
Requirement already satisfied: tqdm>=4.42.1 in c:\users\admin\anaconda3\lib\site-packages (from
huggingface-hub<1.0,>=0.16.4->tokenizers<1,>=0.15->cohere<6.0,>=5.12.0->langchain-cohere) (4.64.0)
Requirement already satisfied: filelock in c:\users\admin\anaconda3\lib\site-packages (from huggingface-
hub<1.0,>=0.16.4->tokenizers<1,>=0.15->cohere<6.0,>=5.12.0->langchain-cohere) (3.6.0)
Requirement already satisfied: fsspec>=2023.5.0 in c:\users\admin\anaconda3\lib\site-packages (from
huggingface-hub<1.0,>=0.16.4->tokenizers<1,>=0.15->cohere<6.0,>=5.12.0->langchain-cohere) (2025.2.0)
Requirement already satisfied: colorama in c:\users\admin\anaconda3\lib\site-packages (from
tqdm>=4.42.1->huggingface-hub<1.0,>=0.16.4->tokenizers<1,>=0.15->cohere<6.0,>=5.12.0->langchain-
cohere) (0.4.4)
  Downloading types_requests-2.32.0.20240914-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240907-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240905-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240712-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240622-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240602-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240523-py3-none-any.whl (15 kB)
  Downloading types_requests-2.32.0.20240521-py3-none-any.whl (15 kB)
  Downloading types_requests-2.31.0.20240406-py3-none-any.whl (15 kB)
  Downloading types_requests-2.31.0.20240403-py3-none-any.whl (15 kB)
  Downloading types_requests-2.31.0.20240402-py3-none-any.whl (15 kB)
  Downloading types_requests-2.31.0.20240311-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.20240310-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.20240218-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.20240125-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.20240106-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.20231231-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.10-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.9-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.8-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.7-py3-none-any.whl (14 kB)
  Downloading types_requests-2.31.0.6-py3-none-any.whl (14 kB)
Collecting types-urllib3
  Downloading types_urllib3-1.26.25.14-py3-none-any.whl (15 kB)
Requirement already satisfied: mypy-extensions>=0.3.0 in c:\users\admin\anaconda3\lib\site-packages (from
typing-inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain-community) (0.4.3)
Requirement already satisfied: sniffio>=1.1 in c:\users\admin\anaconda3\lib\site-packages (from anyio-
>httpx>=0.21.2->cohere<6.0,>=5.12.0->langchain-cohere) (1.2.0)
Installing collected packages: types-urllib3, types-requests, fastavro, types-pyyaml, cohere, langchain-cohere
Successfully installed cohere-5.13.12 fastavro-1.10.0 langchain-cohere-0.4.2 types-pyyaml-6.0.12.20241230
types-requests-2.31.0.6 types-urllib3-1.26.25.14
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
```

```
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
```

Input cell:

```
!pip install gdown
```

Output:

```
Collecting gdown
  Downloading gdown-5.2.0-py3-none-any.whl (18 kB)
Requirement already satisfied: requests[socks] in c:\users\admin\anaconda3\lib\site-packages (from gdown)
(2.27.1)
Requirement already satisfied: tqdm in c:\users\admin\anaconda3\lib\site-packages (from gdown) (4.64.0)
Requirement already satisfied: beautifulsoup4 in c:\users\admin\anaconda3\lib\site-packages (from gdown)
(4.11.1)
Requirement already satisfied: filelock in c:\users\admin\anaconda3\lib\site-packages (from gdown) (3.6.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\admin\anaconda3\lib\site-packages (from
beautifulsoup4->gdown) (2.3.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\admin\anaconda3\lib\site-packages
(from requests[socks]->gdown) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\admin\anaconda3\lib\site-packages (from
requests[socks]->gdown) (1.26.9)
Requirement already satisfied: idna<4,>=2.5 in c:\users\admin\anaconda3\lib\site-packages (from
requests[socks]->gdown) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\admin\anaconda3\lib\site-packages (from
requests[socks]->gdown) (2021.10.8)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in c:\users\admin\anaconda3\lib\site-packages (from
requests[socks]->gdown) (1.7.1)
Requirement already satisfied: colorama in c:\users\admin\anaconda3\lib\site-packages (from tqdm->gdown)
(0.4.4)
Installing collected packages: gdown
Successfully installed gdown-5.2.0
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
```

Input cell:

```
#### Step 2: Get the Cohere API Key
   - Go to Cohere’s website.
   - Sign in or create an account.
   - Navigate to the API Keys section in your account settings.
   - Copy the API key.
```

#### Step 3: Set the Cohere API Key in the Environment and Create an instance of the Cohere LLM

Input Cell:

```
import getpass
import os

if not os.environ.get("COHERE_API_KEY"):
   os.environ["COHERE_API_KEY"] = getpass.getpass("Enter API key for Cohere: ")

from langchain_cohere import ChatCohere

model = ChatCohere(model="command-r7b-12-2024")
```

#### Test the Cohere Model

```
from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_template("Tell me a quote on the {topic}")
chain = prompt | model


chain.invoke({"topic": "AI"}).content
```

Ouput:

```
'Here\'s a quote on artificial intelligence:\n\n"Artificial intelligence is the future. It\'s not just about computers or robots. It\'s about how we can use the power of data and algorithms to make our lives better, to solve problems, and to create new opportunities." - Sundar Pichai, CEO of Google.\n\nThis quote emphasizes the broad impact of AI beyond just technology, highlighting its potential to improve lives, solve complex problems, and drive innovation.'
```

Step 4: Create a text file and upload it to Google Drive.
Load the text file from the Google Drive

```
import gdown
#https://drive.google.com/file/d/1eeoLCTs-BS2Wl2RcX_tL83h_kCZIZxDA/view?usp=sharing
# Google Drive file ID (Extract from the URL)
file_id = "1eeoLCTs-BS2Wl2RcX_tL83h_kCZIZxDA"
file_path = "ai_agents_info.txt"

# Download the file
gdown.download(f"https://drive.google.com/uc?export=download&id={file_id}", file_path,
quiet=False)



# Read the file
with open(file_path, "r", encoding="utf-8") as file:
   document_text = file.read()

print(document_text)
```

Output cell:

```
AI agents are autonomous systems capable of perceiving their environment,
making decisions, and executing actions to achieve specific goals. They can be classified into several types:

1. Reactive Agents: These agents do not store past experiences and make decisions solely based on the current situation.
   Examples include chess-playing programs that evaluate only the present board state.

2. Deliberative Agents: These agents build models of the world and use planning to achieve their goals.
   They use reasoning mechanisms to determine the best course of action.

3. Learning Agents: These agents improve their performance over time using machine learning techniques.
   Reinforcement learning-based robots are an example of learning agents.

4. Multi-Agent Systems (MAS): A system where multiple AI agents interact, collaborate, or compete to complete tasks.
   Applications include swarm robotics and distributed AI.

5. Utility-Based Agents: These agents maximize a utility function, ensuring optimal decision-making.
   They are widely used in economics and game theory.

AI agents are applied in various domains, including healthcare, finance, robotics, and natural language processing.
Their ability to adapt and learn from data makes them crucial in modern AI applications.
```

Step 5: Creating a Prompt Template and Inferencing
#### Example use cases:
1. Extract the desired information from the given text
2. Extract PII information.
3. Extract the Key Words from a technical document.

Input cell:

```
from langchain_core.prompts import ChatPromptTemplate

prompt = ChatPromptTemplate.from_template("Extract and list the types of AI agents as bullet points from the following text:{document_text}")
chain = prompt | model
```

Input Cell:

```
print(chain.invoke({"document_text": document_text}).content)
```

Output cell:

```
Here are the types of AI agents listed from the text:

- Reactive Agents
- Deliberative Agents
- Learning Agents
- Multi-Agent Systems (MAS)
- Utility-Based Agents
```

Program 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.

Theory:

This Python program integrates various libraries and concepts to extract institutionrelated data from Wikipedia and structure it using Pydantic. Provides an interactive interface using ipy widgets. Here's the breakdown of the key components and concepts:

1. Libraries and Their Roles

wikipedia-api:

o A Python library used to interact with Wikipedia and fetch pages.

o Provides access to the Wikipedia REST API to extract article text and metadata.

pydantic:

o A data validation and settings management library using Python type annotations.

o Base Model is used to define a schema for institution details, ensuring structured data.

ipywidgets:

o Used to create interactive widgets (text boxes, buttons) in Jupyter Notebooks.

o Provides a user-friendly interface for input and output.

IPython.display:

o Enables rich output display in Jupyter Notebooks, allowing dynamic updates of results.

Approach 1: Using Cohere and LangChain

Input Cell:

```
# Install the langchain-cohere library (command to be run in the terminal, not Python code)
# pip install -U langchain-cohere

# Import necessary modules from langchain and pydantic
from langchain.prompts import PromptTemplate # For creating prompt templates
from langchain.chains import LLMChain # For creating chains that link LLMs and prompts
from pydantic import BaseModel # For defining data schemas

# Define Pydantic schema for the desired output
class InstitutionDetails(BaseModel):
    """
    Pydantic model to structure the output data for institution details.
    """
    founder: str # Founder of the institution (string)
    founded: str # Year/date when the institution was founded (string)
    branches: int # Number of current branches (integer)
    employees: int # Number of employees working in the institution (integer)
    summary: str # A 4-line brief summary of the institution (string)
```

input cell:

```
# Define the prompt template for GPT-3
prompt_template = """
Given the name of an institution, extract the following details from Wikipedia:
1. Founder of the institution
2. When it was founded
3. Current branches of the institution
4. How many employees work in it
5. A 4-line brief summary of the institution

Institution: {institution_name}
"""

import getpass
import os

# Check if the COHERE_API_KEY environment variable is already set
if not os.environ.get("COHERE_API_KEY"):
   # If not set, prompt the user to enter their Cohere API key and set it as an environment variable
   os.environ["COHERE_API_KEY"] = getpass.getpass("Enter API key for Cohere: ")

# Import the ChatCohere class from the langchain_cohere library
from langchain_cohere import ChatCohere

# Initialize the ChatCohere model with a specific model version (command-r7b-12-2024)
model = ChatCohere(model="command-r7b-12-2024")

# Setup Langchain with the prompt and model

# Create a PromptTemplate object, specifying input variables and the template
prompt = PromptTemplate(input_variables=["institution_name"], template=prompt_template)

# Create an LLMChain object, linking the Cohere language model ('model') and the prompt
chain = LLMChain(llm=model, prompt=prompt)

# Function to fetch institution details using GPT-3
def fetch_institution_details(institution_name: str):
   """
   Fetches institution details using the Langchain chain and GPT-3 model.

   Args:
      institution_name (str): The name of the institution to fetch details for.

   Returns:
      str: The result from the LLMChain run, containing institution details.
   """
   # Run the LLMChain with the institution name as input and get the result
   result = chain.run(institution_name=institution_name)
   return result
```

```
# Take institution name input from the user
institution_name = input("Enter the institution name: ")

# Call the function to fetch institution details, passing the user input
institution_details = fetch_institution_details(institution_name)

# Print the fetched institution details
print(institution_details)
```

Output cell:

```
Enter the institution name: VTU belagavi
C:\Users\admin\AppData\Local\Temp\ipykernel_13572\2651505773.py:21: LangChainDeprecationWarning:
The method `Chain.run` was deprecated in langchain 0.1.0 and will be removed in 1.0. Use :meth:`~invoke`
instead.
  result = chain.run(institution_name=institution_name)
## VTU Belagavi Details

**1. Founder:** The Visvesvaraya Technological University (VTU) Belagavi was established by the
Government of Karnataka in 1997. It is named after Dr. Visvesvaraya, a renowned Indian engineer and
statesman.

**2. Founding Date:** The university was founded on July 20, 1997.

**3. Current Branches:** VTU Belagavi has several affiliated colleges and institutes across Karnataka. As
of 2023, it has over 150 affiliated engineering colleges, offering undergraduate and postgraduate programs
in various engineering disciplines.

**4. Number of Employees:** The exact number of employees within VTU Belagavi itself is not publicly
available. However, the affiliated colleges and institutes employ a significant number of faculty members,
administrative staff, and support staff.

**5. Brief Summary:** VTU Belagavi is a public technical university in Karnataka, India. It was established
to promote technical education and research in the state. The university offers a wide range of undergraduate
and postgraduate engineering programs through its affiliated colleges. VTU Belagavi plays a crucial role in
shaping the technical workforce in Karnataka and contributes to the development of the state's engineering
sector.
```

Approach 2 Using WikiPediaAPIWrapper

Input cell:

```
%pip install --upgrade --quiet  Wikipedia
```

Output cell:

```
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -rpcio (c:\users\admin\anaconda3\lib\site-packages)
```

Input cell:
```
from langchain_community.tools import WikipediaQueryRun
from langchain_community.utilities import WikipediaAPIWrapper
from pydantic import BaseModel, Field
import re

# Step 1: Define the Pydantic schema
class InstitutionDetails(BaseModel):
    founder: str = Field(..., description="Founder of the institution")
    founded_year: str = Field(..., description="Year the institution was founded")
    branches: list[str] = Field(..., description="Current branches in the institution")
    employees: str = Field(..., description="Number of employees in the institution")
    summary: str = Field(..., description="A brief 4-line summary of the institution")

# Step 2: Create a custom output parser
def parse_wikipedia_content(content: str) -> InstitutionDetails:
    founder_match = re.search(r"Founded by\s*([\w\s,]+)", content)
    founded_year_match = re.search(r"Established in\s*(\d{4})", content)
    branches_match = re.findall(r"(\b[A-Z][a-zA-Z\s]+ Campus\b)", content)
    employees_match = re.search(r"(\d{3,6})\s*employees", content)

    summary_sentences = content.split(". ")[:4]  # Extract first 4 sentences

    return InstitutionDetails(
        founder=founder_match.group(1) if founder_match else "Not Found",
        founded_year=founded_year_match.group(1) if founded_year_match else "Not Found",
        branches=branches_match if branches_match else ["Not Found"],
        employees=employees_match.group(1) if employees_match else "Not Found",
        summary=". ".join(summary_sentences)
    )

# Step 3: Fetch details from Wikipedia
wiki = WikipediaQueryRun(api_wrapper=WikipediaAPIWrapper())
institution_name = "Infosys"
wiki_content = wiki.run(institution_name)

# Step 4: Parse and display results
institution_details = parse_wikipedia_content(wiki_content)
print(institution_details.model_dump_json(indent=4))
```

Output cell:
```
{
    "founder": "Not Found",
    "founded_year": "Not Found",
    "branches": [
        "Not Found"
    ],
    "employees": "Not Found",
    "summary": "Page: Infosys\nSummary: Infosys Limited is an Indian multinational technology company that offers business consulting, information technology, and outsourcing services. Founded in 1981, the company is headquartered in Bengaluru.\nOn 24 August 2021, Infosys became the fourth Indian company to achieve a market capitalization of US$100 billion. As of 2024, Infosys is the second-largest Indian Big Tech company by revenue and market capitalization.\nInfosys has also attracted controversies due to allegations of visa and tax fraud in the United States and for creating malfunctioning government websites.\n\nPage: Infosys Foundation\nSummary: Infosys Foundation is a non-profit organisation based in Karnataka, India, established in 1996 by Infosys to support the underprivileged sections of society.  It supports programs in the areas of education, rural development, healthcare, arts and culture, and destitute care   in remote regions of   India"
}
```

Program 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.

Theory:

This Python program creates an interactive chatbot focused on answering questions related to the Indian Penal Code (IPC). It utilizes AI language models, Wikipedia data extraction, and data validation techniques to provide accurate, structured legal information. Let's break down the key concepts and components used in this program.

1. Required Libraries and Their Roles

LangChain:
A framework for developing applications powered by language models. It provides utilities for building prompts, managing LLM (Large Language Model) chains, and integrating different AI models seamlessly.

Cohere:
A platform offering state-of-the-art NLP models for text generation, classification, and more. Here, it powers the AI responses using its "command" model, which is optimized for complex text-to-text tasks like answering legal queries.

Pydantic:
A data validation and parsing library that enforces data models using Python type hints. It ensures that the chatbot’s responses are structured, reliable, and adhere to defined schemas.

Wikipedia-API:
A Python library that interacts with Wikipedia to fetch information. It enables the chatbot to pull the Indian Penal Code content directly from Wikipedia.

IPython Widgets (ipywidgets):
A library that allows for the creation of interactive widgets in Jupyter Notebooks, making it possible to build user-friendly interfaces.

```
from langchain.text_splitter import RecursiveCharacterTextSplitter,
SentenceTransformersTokenTextSplitter  # Import text splitters from langchain

import numpy as np  # Import NumPy for numerical operations

from pypdf import PdfReader  # Import PdfReader to read PDF files

from tqdm import tqdm  # Import tqdm for progress bar visualization
def word_wrap(string, n_chars=72):
    """
    Wraps a long string to a specified number of characters per line.

    Args:
        string (str): The string to wrap.
        n_chars (int, optional): The maximum number of characters per line. Defaults to 72.
```

```
Returns:
    str: The word-wrapped string.
"""
if len(string) < n_chars:
    return string  # Return the string directly if it's shorter than the character limit
else:
    # Find the last space before the character limit and insert a newline
    return string[:n_chars].rsplit(' ', 1)[0] + '\n' + \
        word_wrap(string[len(string[:n_chars].rsplit(' ', 1)[0]) + 1:], n_chars)

from pypdf import PdfReader # Import PdfReader from the pypdf library

reader = PdfReader("BNS(IPC).pdf") # Create a PdfReader object to read the PDF file "BNS (IPC).pdf"
pdf_texts = [p.extract_text().strip() for p in reader.pages] # Extract text from each page of the PDF,
strip whitespace, and store in a list

# Filter out empty strings from the list of extracted texts
pdf_texts = [text for text in pdf_texts if text]

print(word_wrap(pdf_texts[0])) # Print the word-wrapped version of the first text element extracted
from the PDF
```

Ouput Cell:

```
THE BHARA TIYA NY AYA SANHITA, 2023
NO. 45 OF2023
[25th December
,2023.]
An Act to consolidate and amend the provisions relating to
offences and for
matters connected therewithor incidentalthereto.
BEit
enacted by Parliament in the Seventy-fourth Year of the Republic of
India as
follows:—
CHAPTERI
PRELIMINARY
1.(1) This Act may be called
the Bharatiya Nyaya Sanhita, 2023.
(2) It shall come into force on such
date as the Central Government may , bynotification
in the Official
Gazette, appoint, and different dates maybe appointed for different
provisions
of this Sanhita.Short
title,
commencement
and
application.vlk/kkj.k
EXTRAORDINARY
Hkkx II
—[k.M 1
PART II — Section 1
izkf/kdkj ls izdkf'kr
PUBLISHED BY
AUTHORITY
lañ 53] ubZ fnYyh] lkseokj] fnlEcj 25] 2023@ ikS"k 4] 1945
¼'kd½
No. 53] NEW DELHI, MONDAY, DECEMBER 25, 2023/PAUSHA 4, 1945
(SAKA)
bl Hkkx esa fHkUu i`"B la[;k nh tkrh gS ftlls fd ;g vyx ladyu ds
:i esa j[kk tk ldsA
Separate paging is given to this Part in order that
it may be filed as a separate compilation.xxxGIDHxxx
xxxGIDExxx
jftLVªh
lañ Mhñ ,yñ —(,u)04@0007@2003 —23 REGISTERED NO.
DL—(N)04/0007/2003—23
MINISTRY OF LAW AND JUSTICE
(Legislative
Department)
New Delhi, the25th December,2023/Pausha4,1945 (Saka)
The
following Act of Parliament received the assent of the President on
the
25th December, 2023 and is hereby published for general
information:—
सी.जी.-डी.एल.-अ.-25122023-250883
CG-DL-E-25122023-250883
```

Input Cell:

```
# [5]: from langchain.text_splitter import RecursiveCharacterTextSplitter,
SentenceTransformersTokenTextSplitter # Import text splitters from langchain

character_splitter = RecursiveCharacterTextSplitter(# Initialize RecursiveCharacterTextSplitter
   separators=["\n\n", "\n", ". ", " ", ""], # Define separators to split text by priority (double newline,
newline, sentence end, space, character)
   chunk_size=1000, # Define the maximum chunk size in characters
   chunk_overlap=20 # Define the overlap between adjacent chunks in characters
)

character_split_texts = character_splitter.split_text('\n\n'.join(pdf_texts)) # Split the PDF texts into
chunks using the defined character splitter, joining the pdf texts with double newlines first

print(word_wrap(character_split_texts[10])) # Print the word-wrapped version of the 11th chunk
(index 10)
print(f"\nTotal chunks: {len(character_split_texts)}") # Print the total number of chunks created
```

Output Cell:

```
Illustration.
A Magistrate exercising jurisdiction in respect of a
charge on which he has
power to sentence to fine or imprisonment, with
or without appeal, is a Judge;
(17) “life” means the life of a human
being, unless the contrary appears from the
context;
(18) “local law”
means a law applicable only to a particular part of India;
(19) “man”
means male human being of any age;
(20) “month” and “year”.－－Wherever
the word “month” or the word “year” is
used, it is to be understood
that the month or the year is to be reckoned according to
the Gregorian
calendar;
(21) “movable property” includes property of every
description, except land
and things attached to the earth or
permanently fastened to anything which is attached
to the earth;
(22)
 “number”.—Unless the contrary appears from the context, words
importing
the singular number include the plural number, and words
importing the plural number
include the singular number;

Total chunks: 480
```

Input Cell:

```
from getpass import getpass # Import getpass for secure password input

#from Langchain import HuggingFaceHub # commented out line, likely an old import statement

from langchain_community.llms import HuggingFaceHub # Import HuggingFaceHub from
langchain_community for language models
import os # Import os module for environment variables

inference_api_key = getpass() # Prompt user to enter their Hugging Face API key securely
#place your huggingface API key after running this cell

os.environ["HUGGINGFACEHUB_API_TOKEN"] = inference_api_key # Set the Hugging Face API
key as an environment variable

inference_api_key # Display the API key (for debugging or confirmation, but generally not recommended
to print API keys)
```

```
from langchain_community.embeddings import HuggingFaceInferenceAPIEmbeddings # Import
HuggingFaceInferenceAPIEmbeddings for embeddings

embedding_function = HuggingFaceInferenceAPIEmbeddings( # Initialize
HuggingFaceInferenceAPIEmbeddings
                    api_key=inference_api_key, # Pass the API key for authentication
                    model_name="sentence-transformers/all-MiniLM-l6-v2" # Specify the model to use
for embeddings (all-MiniLM-l6-v2)
)

from langchain_community.vectorstores import FAISS # Import FAISS for vector storage
db = FAISS.from_texts(character_split_texts, embedding_function) # Create a FAISS vector database
from the split texts and embedding function

print(db.index.ntotal) # Print the total number of vectors in the FAISS index
……..
480
```

Input Cell

```
query = "What does BNS Section 72 talks about ?" # Define the query string to search for in the vector
database

retrieved_documents = db.similarity_search(query) # Perform a similarity search in the vector database
'db' using the defined query
                                # and store the retrieved documents in the 'retrieved_documents' variable
```

Input cell :

```
retrieved_documents
```

Output Cell:

```
[Document(id='3841859c-1d6e-4d2b-ba08-572014f3724b', metadata={}, page_content='victim.\n71.Whoev
er has been previously convicted of an offence punishable under\nsection 64 or section 65 or section 66 or se
ction 70 and is subsequently convicted of an\noffence punishable under any of the said sections shall be puni
shed with imprisonment for\nlife which shall mean imprisonment for the remainder of that person’s natural
 life, or with\ndeath.\n72.(1) Whoever prints or publishes the name or any matter which may make known\nt
he identity of any person against whom an offence under section 64 or section 65 or\nsection 66 or section 6
7 or section 68 or section 69 or section 70 or section 71 is alleged or\nfound to have been committed (hereaft
er in this section referred to as the victim) shall be\npunished with imprisonment of either description for a t
erm which may extend to two years\nand shall also be liable to fine.\n(2) Nothing in sub-section (1) extends
to any printing or publication of the name or'),
 Document(id='228db1fb-9d82-4e5d-9f39-3c38bf6be90a', metadata={}, page_content='it is his duty\nto pre
vent.Sec. 1]THE GAZETTE OF INDIA EXTRAORDINARY21____________________________________
_________________________________________________________________________________________
_________________________________________________________________________________________
_________________________________________________________________________________________
_________________________________________________________________'),
 Document(id='6ce20620-9149-411d-8ffd-a2dfff0ba5cd', metadata={}, page_content='Explanation.—For th
e purposes of this sub-section, “recognised welfare institution\nor organisation” means a social welfare in
stitution or organisation recognised in this behalf\nby the Central Government or the State Government.\n73.
Whoever prints or publishes any matter in relation to any proceeding before a\nCourt with respect to an offe
nce referred to in section 72 without the previous permission of\nsuch Court shall be punished with imprison
ment of either description for a term which may\nextend to two years and shall also be liable to fine.\nExpla
nation.—The printing or publication of the judgment of any High Court or the\nSupreme Court does not am
ount to an offence within the meaning of this section.\nOf criminal force and assault against woman\n74.Wh
oever assaults or uses criminal force to any woman, intending to outrage or\nknowing it to be likely that he
will thereby outrage her modesty, shall be punished with'),
 Document(id='16d81999-6f71-4d9c-b4fd-7b274ce17a8d', metadata={}, page_content='THE BHARA TIY
A NY AYA SANHITA, 2023\nNO. 45 OF2023\n[25th December ,2023.]\nAn Act to consolidate and amen
d the provisions relating to offences and for\nmatters connected therewithor incidentalthereto.\nBEit enacted
 by Parliament in the Seventy-fourth Year of the Republic of India as\nfollows: – – \nCHAPTERI\nPRELI
MINARY\n1.(1) This Act may be called the Bharatiya Nyaya Sanhita, 2023.\n(2) It shall come into force on
 such date as the Central Government may , bynotification\nin the Official Gazette, appoint, and different dat
es maybe appointed for different provisions\nof this Sanhita.Short title,\ncommencement\nand\napplication.
vlk/kkj.k\nEXTRAORDINARY\nHkkx II —[k.M 1\nPART II — Section 1\nizkf/kdkj ls izdkf\'kr\nPUBLIS
HED BY AUTHORITY\nlañ 53] ubZ fnYyh] lkseokj] fnlEcj 25] 2023@ ikS"k 4] 1945 ¼\'kd½\nNo. 53] NE
W DELHI, MONDAY, DECEMBER 25, 2023/PAUSHA 4, 1945 (SAKA)\nbl Hkkx esa fHkUu i`"B la[;k n
h tkrh gS ftlls fd ;g vyx ladyu ds :i esa j[kk tk ldsA')]
```

Input Cell:

!pip install gradio

Output Cell

```
Collecting gradio
  Downloading gradio-4.44.1-py3-none-any.whl (18.1 MB)
     ---------------------------------------- 18.1/18.1 MB 3.6 MB/s eta 0:00:00
Collecting tomlkit==0.12.0
  Downloading tomlkit-0.12.0-py3-none-any.whl (37 kB)
Requirement already satisfied: typer<1.0,>=0.12 in c:\users\dell\anaconda3\lib\site-packages (from gradio)
(0.12.3)
Collecting aiofiles<24.0,>=22.0
  Downloading aiofiles-23.2.1-py3-none-any.whl (15 kB)
Requirement already satisfied: markupsafe~=2.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio)
(2.0.1)
Collecting urllib3~=2.0
  Using cached urllib3-2.3.0-py3-none-any.whl (128 kB)
Requirement already satisfied: jinja2<4.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (2.11.3)
Requirement already satisfied: fastapi<1.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (0.111.
1)
Requirement already satisfied: importlib-resources<7.0,>=1.3 in c:\users\dell\anaconda3\lib\site-packages (f
rom gradio) (6.4.0)
Requirement already satisfied: anyio<5.0,>=3.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (3.
5.0)
Collecting ffmpy
  Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Collecting gradio-client==1.3.0
  Downloading gradio_client-1.3.0-py3-none-any.whl (318 kB)
     ------------------------------------ 318.7/318.7 kB 20.6 MB/s eta 0:00:00
Requirement already satisfied: uvicorn>=0.14.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (0.
30.3)
```

```
Requirement already satisfied: typing-extensions~=4.0 in c:\users\dell\anaconda3\lib\site-packages (from gr
adio) (4.12.2)
Requirement already satisfied: httpx>=0.24.1 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (0.2
7.0)
Requirement already satisfied: python-multipart>=0.0.9 in c:\users\dell\anaconda3\lib\site-packages (from g
radio) (0.0.9)
Requirement already satisfied: pyyaml<7.0,>=5.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio)
(6.0)
Requirement already satisfied: orjson~=3.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (3.10.
6)
Collecting semantic-version~=2.0
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: huggingface-hub>=0.19.3 in c:\users\dell\anaconda3\lib\site-packages (from
gradio) (0.27.1)
Collecting ruff>=0.2.2
  Downloading ruff-0.9.5-py3-none-win_amd64.whl (10.9 MB)
     ---------------------------------------- 10.9/10.9 MB 3.6 MB/s eta 0:00:00
Requirement already satisfied: numpy<3.0,>=1.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio)
(1.24.4)
Requirement already satisfied: matplotlib~=3.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (3.
5.2)
Requirement already satisfied: pandas<3.0,>=1.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio)
(1.4.4)
Requirement already satisfied: pillow<11.0,>=8.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio)
(9.2.0)
Requirement already satisfied: packaging in c:\users\dell\anaconda3\lib\site-packages (from gradio) (24.1)
Requirement already satisfied: pydantic>=2.0 in c:\users\dell\anaconda3\lib\site-packages (from gradio) (2.9.
2)
Requirement already satisfied: fsspec in c:\users\dell\anaconda3\lib\site-packages (from gradio-client==1.3.
0->gradio) (2024.6.1)
Requirement already satisfied: websockets<13.0,>=10.0 in c:\users\dell\anaconda3\lib\site-packages (from g
radio-client==1.3.0->gradio) (12.0)
Requirement already satisfied: sniffio>=1.1 in c:\users\dell\anaconda3\lib\site-packages (from anyio<5.0,>=
3.0->gradio) (1.2.0)
Requirement already satisfied: idna>=2.8 in c:\users\dell\anaconda3\lib\site-packages (from anyio<5.0,>=3.
0->gradio) (3.3)
Requirement already satisfied: fastapi-cli>=0.0.2 in c:\users\dell\anaconda3\lib\site-packages (from fastapi<
1.0->gradio) (0.0.4)
Requirement already satisfied: starlette<0.38.0,>=0.37.2 in c:\users\dell\anaconda3\lib\site-packages (from f
astapi<1.0->gradio) (0.37.2)
Requirement already satisfied: email_validator>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from fas
tapi<1.0->gradio) (2.2.0)
Requirement already satisfied: httpcore==1.* in c:\users\dell\anaconda3\lib\site-packages (from httpx>=0.24.
1->gradio) (1.0.5)
Requirement already satisfied: certifi in c:\users\dell\anaconda3\lib\site-packages (from httpx>=0.24.1->gra
dio) (2022.9.14)
Requirement already satisfied: h11<0.15,>=0.13 in c:\users\dell\anaconda3\lib\site-packages (from httpcore
==1.*->httpx>=0.24.1->gradio) (0.14.0)
Requirement already satisfied: requests in c:\users\dell\anaconda3\lib\site-packages (from huggingface-hub>
=0.19.3->gradio) (2.28.1)
```

```
Requirement already satisfied: tqdm>=4.42.1 in c:\users\dell\anaconda3\lib\site-packages (from huggingface
-hub>=0.19.3->gradio) (4.66.4)
Requirement already satisfied: filelock in c:\users\dell\anaconda3\lib\site-packages (from huggingface-hub>
=0.19.3->gradio) (3.6.0)
Requirement already satisfied: zipp>=3.1.0 in c:\users\dell\anaconda3\lib\site-packages (from importlib-reso
urces<7.0,>=1.3->gradio) (3.21.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\dell\anaconda3\lib\site-packages (from matplotl
ib~=3.0->gradio) (1.4.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\dell\anaconda3\lib\site-packages (from matplotl
ib~=3.0->gradio) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\users\dell\anaconda3\lib\site-packages (from matplotlib~=
3.0->gradio) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\dell\anaconda3\lib\site-packages (from matplotli
b~=3.0->gradio) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\dell\anaconda3\lib\site-packages (from matp
lotlib~=3.0->gradio) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\anaconda3\lib\site-packages (from pandas<3.
0,>=1.0->gradio) (2022.1)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\dell\anaconda3\lib\site-packages (from py
dantic>=2.0->gradio) (0.7.0)
Requirement already satisfied: pydantic-core==2.23.4 in c:\users\dell\anaconda3\lib\site-packages (from pyd
antic>=2.0->gradio) (2.23.4)
Requirement already satisfied: click>=8.0.0 in c:\users\dell\anaconda3\lib\site-packages (from typer<1.0,>=
0.12->gradio) (8.0.4)
Requirement already satisfied: rich>=10.11.0 in c:\users\dell\anaconda3\lib\site-packages (from typer<1.0,>
=0.12->gradio) (13.7.1)
Requirement already satisfied: shellingham>=1.3.0 in c:\users\dell\anaconda3\lib\site-packages (from typer<
1.0,>=0.12->gradio) (1.5.4)
Requirement already satisfied: colorama in c:\users\dell\anaconda3\lib\site-packages (from click>=8.0.0->ty
per<1.0,>=0.12->gradio) (0.4.5)
Requirement already satisfied: dnspython>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages (from email_v
alidator>=2.0.0->fastapi<1.0->gradio) (2.3.0)
Requirement already satisfied: six>=1.5 in c:\users\dell\anaconda3\lib\site-packages (from python-dateutil>
=2.7->matplotlib~=3.0->gradio) (1.17.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\dell\anaconda3\lib\site-packages (from r
ich>=10.11.0->typer<1.0,>=0.12->gradio) (2.19.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\dell\anaconda3\lib\site-packages (from ric
h>=10.11.0->typer<1.0,>=0.12->gradio) (3.0.0)
Requirement already satisfied: python-dotenv>=0.13 in c:\users\dell\anaconda3\lib\site-packages (from uvic
orn>=0.14.0->gradio) (1.0.0)
Requirement already satisfied: watchfiles>=0.13 in c:\users\dell\anaconda3\lib\site-packages (from uvicorn>
=0.14.0->gradio) (0.22.0)
Requirement already satisfied: httptools>=0.5.0 in c:\users\dell\anaconda3\lib\site-packages (from uvicorn>
=0.14.0->gradio) (0.6.1)
Collecting requests
  Using cached requests-2.32.3-py3-none-any.whl (64 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\dell\anaconda3\lib\site-packages (from
requests->huggingface-hub>=0.19.3->gradio) (2.0.4)
Requirement already satisfied: mdurl~=0.1 in c:\users\dell\anaconda3\lib\site-packages (from markdown-it-
py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio) (0.1.2)
```

```
Installing collected packages: pydub, urllib3, tomlkit, semantic-version, ruff, ffmpy, aiofiles, requests, gradi
o-client, gradio
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.26.20
    Uninstalling urllib3-1.26.20:
      Successfully uninstalled urllib3-1.26.20
  Attempting uninstall: tomlkit
    Found existing installation: tomlkit 0.11.1
    Uninstalling tomlkit-0.11.1:
      Successfully uninstalled tomlkit-0.11.1
  Attempting uninstall: requests
    Found existing installation: requests 2.28.1
    Uninstalling requests-2.28.1:
      Successfully uninstalled requests-2.28.1
Successfully installed aiofiles-23.2.1 ffmpy-0.5.0 gradio-4.44.1 gradio-client-1.3.0 pydub-0.25.1 requests-2.
32.3 ruff-0.9.5 semantic-version-2.10.0 tomlkit-0.12.0 urllib3-2.3.0
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. T
his behaviour is the source of the following dependency conflicts.
anaconda-project 0.11.1 requires ruamel-yaml, which is not installed.
tensorflow-intel 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 1.24.4 which is incompatible.
langchainplus-sdk 0.0.17 requires pydantic<2,>=1, but you have pydantic 2.9.2 which is incompatible.
conda-repo-cli 1.0.20 requires clyent==1.2.1, but you have clyent 1.2.2 which is incompatible.
conda-repo-cli 1.0.20 requires nbformat==5.4.0, but you have nbformat 5.5.0 which is incompatible.
conda-repo-cli 1.0.20 requires requests==2.28.1, but you have requests 2.32.3 which is incompatible.
botocore 1.34.145 requires urllib3<1.27,>=1.25.4; python_version < "3.10", but you have urllib3 2.3.0 whic
h is incompatible.
```

Input cell:

```
# Bharathiya Nyay Sanhita - IPC Chatbot

import cohere # Import the Cohere library for language model interaction
import gradio as gr # Import Gradio for creating a user interface

from langchain_community.vectorstores import FAISS # Import FAISS for vector storage

# Initialize the Cohere client
co = cohere.Client('eA50e27b88O7ycXim4jLv5BxMJurbufz9e1AMJNz') # Replace 'YOUR API' with
your actual Cohere API key

# Initialize the FAISS vector store
db = FAISS.from_texts(character_split_texts, embedding_function) # Create a FAISS index from the
text chunks and embedding function

print("Total indexed documents in FAISS:", db.index.ntotal) # Print the number of documents indexed
in the FAISS vector store
```

Total indexed documents in FAISS: 480

Input Cell:

```
def rag(query, retrieved_documents, model="command"):
    """
    Performs Retrieval-Augmented Generation (RAG) to answer a query based on retrieved documents
using Cohere's chat model.

    Args:
        query (str): The user's question or query.
        retrieved_documents (list): A list of documents retrieved from a vector database that are relevant
to the query.
        model (str, optional): The Cohere chat model to use. Defaults to "command".

    Returns:
        str: The generated answer from the Cohere chat model.
    """
    # Extract the page content from each retrieved document and join them into a single string with double
newlines as separators.
    information = "\n\n".join([docs.page_content for docs in retrieved_documents])

    # Define the messages to be sent to the Cohere chat model.
    # This includes a system message to set the context and a user message with the query and retrieved
information.
    messages = [
        {
            "role": "system",
            "content": """You are a helpful expert in Bharatiya Nyay Sanhita (BNS).
Your users are asking questions about information contained in Bharatiya Nyay Sanhita document.
You will be shown the user's question, and the relevant information from given document.
Note that if asked for section get BNS section number.
Answer the user's question using only this information.""" # System message to guide the model's
behavior as a BNS expert.
        },
        {
            "role": "user",
            "content": f"Question: {query}. \n Information: {information}" # User message containing the
user's query and the retrieved information.
        }
    ]

    # Call the Cohere chat API to get a response based on the provided messages.
    response = co.chat( # Using the 'co' Cohere client (assumed to be initialized elsewhere)
        model=model, # Specify the model to use
        message=query, # Pass the user's query as the main message
        documents=messages # Pass the list of messages including system and user prompts
    )

    return response.text # Return the text of the generated response from the Cohere chat model.
```

```
# Define the chatbot function
def chatbot(query):
    """
    Chatbot function to answer user queries based on retrieved documents using RAG.

    Args:
        query (str): The user's question or query.

    Returns:
        tuple: A tuple containing the response from the RAG model and the source text from retrieved
documents.
              Returns an error message and empty source text if an exception occurs.
    """
    try:
        # Query FAISS to get retrieved documents
        retrieved_documents = db.similarity_search(query, k=5) # Perform similarity search in FAISS
database 'db' using the query and retrieve top 5 documents

        # Debug: print retrieved documents
        # print("Retrieved Documents:", retrieved_documents) # Commented out debug print statement

        # Call the RAG function
        response = rag(query, retrieved_documents) # Call the rag function (defined previously) to generate
a response using the query and retrieved documents

        source_text = "\n\n".join([doc.page_content for doc in retrieved_documents]) # Combine the
page content of retrieved documents into a single string separated by double newlines
        return response, source_text # Return the generated response and the combined source text

    except Exception as e: # Catch any exceptions that might occur during the process
        # Debug: print exception details
        print("Error:", e) # Print the error message to the console

        return str(e), "" # Return the error message as a string and an empty string for source text in case of
error
```

```
# Set up the Gradio interface
iface = gr.Interface( # Create a Gradio Interface object

    fn=chatbot, # Pass the user query to the chatbot function. 'chatbot' is assumed to be a function defined
elsewhere in the code.
    inputs=gr.Textbox(lines=2, placeholder="Ask a Bharatiya Nyay Sanhita (BNS /IPC) question..."), #
Define the input component as a textbox with 2 lines and a placeholder text.

    outputs=[ # Define the output components as a list
        gr.Textbox(label="Response", lines=4), # First output is a textbox labeled "Response" with 4 lines.
        gr.Textbox(label="Source Text", lines=10) # Second output is a textbox labeled "Source Text" with
10 lines.
    ],
    title="Bharatiya Nyay Sanhita", # Set the title of the Gradio interface.
    description="Ask any Bharathiya Nyay Sanhita related question, and I will provide answers based
on the relevant information." # Set the description of the Gradio interface.
)

# Launch the Gradio interface
iface.launch() # Launch the Gradio interface to start the chatbot application.
```

```
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.

Error: 0
```