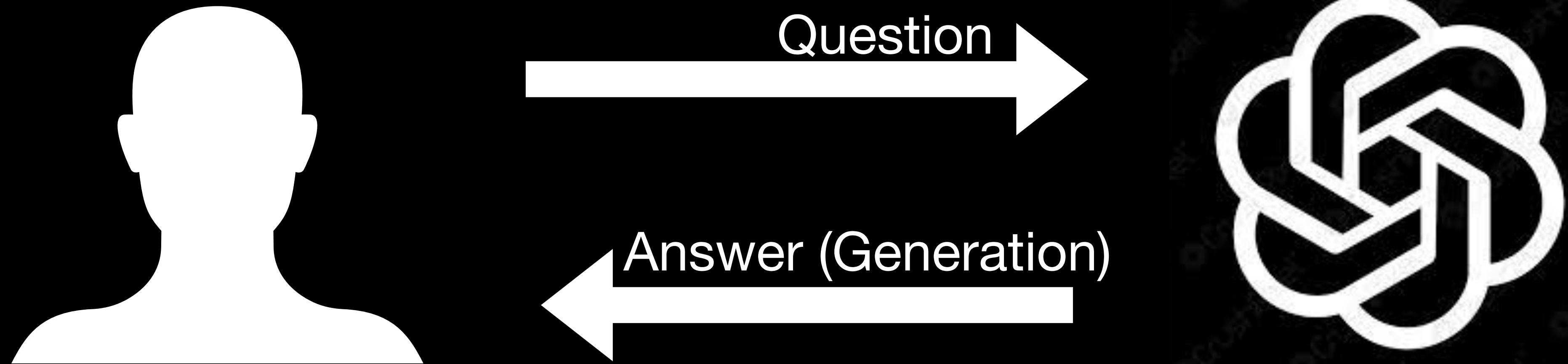


Retrieval-Augmented Generation (RAG) in Healthcare

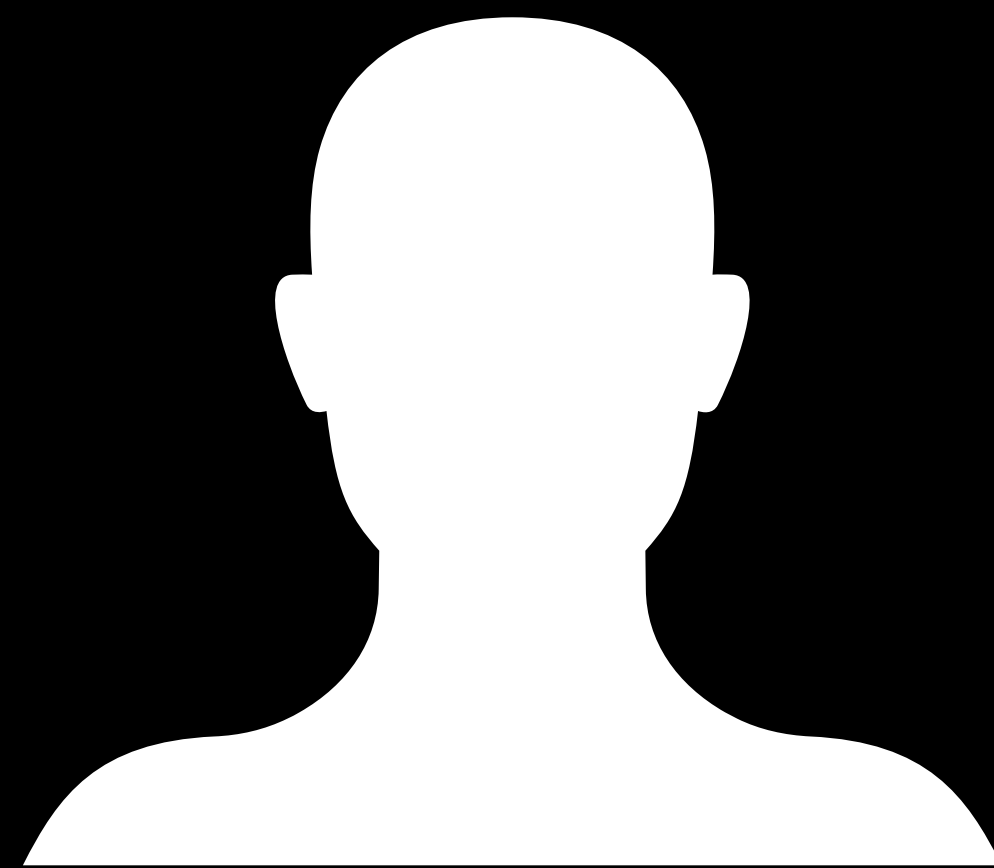
Dr Nagender Aneja

LLMs: Question => Answer



- Built on supervised learning principles.
- Operational Mechanism: Predicting the next word in a sequence.
- Pre-training stage: Next Token Prediction, Foundation Model
- Supervised fine-tuning stage: Instruction Prompt Pairs e.g. (Instruction, Input, Output)
- Alignment stage: Reinforcement learning from human feedback (RLHF)

LLMs: Question + Information => Answer



Question + Information

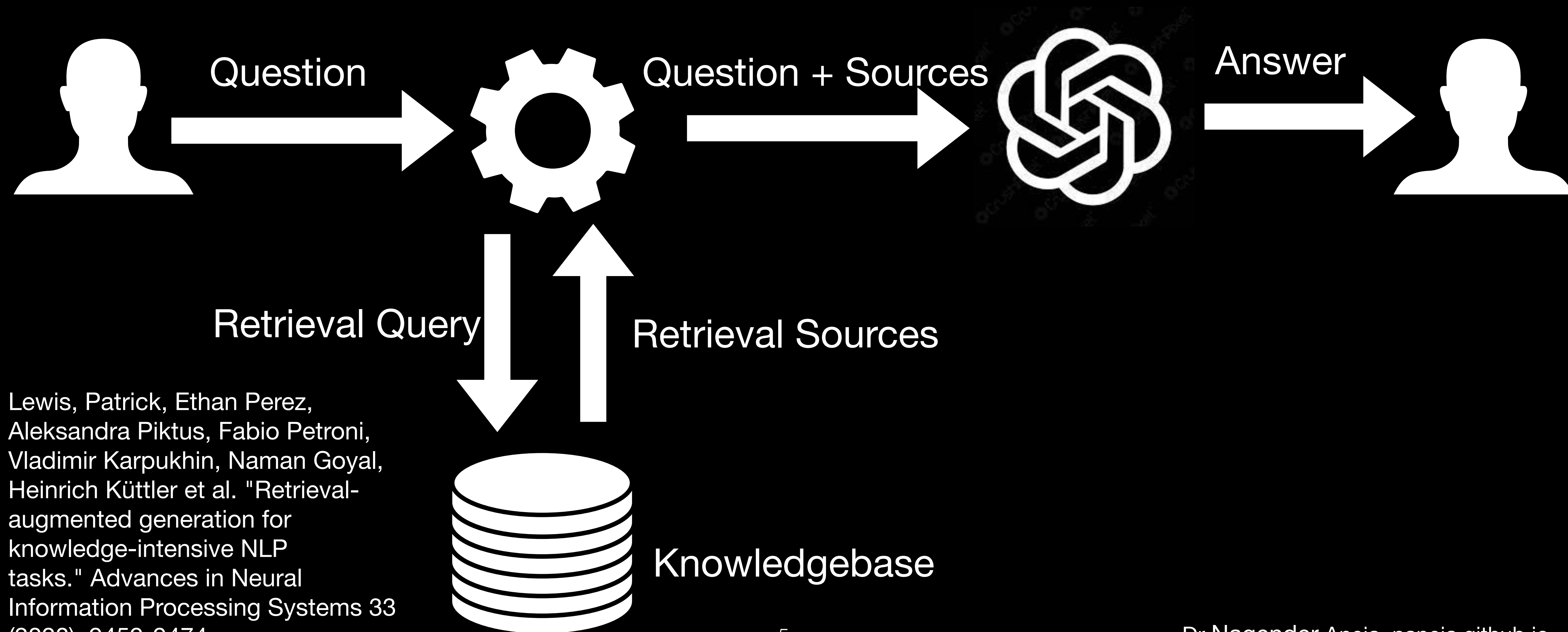


Answer (Generation)

What if the Information is large

- Increase in Latency
- Increase in Cost if paid API is being used
- Decline in Accuracy when lot of information is irrelevant

Retrieval-Augmented Generation



Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler et al. "Retrieval-augmented generation for knowledge-intensive NLP tasks." Advances in Neural Information Processing Systems 33 (2020): 9459-9474.

How RAG Works

- Retrieves relevant data from external sources before generating answer.
- Incorporates external knowledge (e.g., private docs, PDFs, databases) into prompts.
- Key Benefits:
 - Increased Accuracy, Source Citation, Dynamic Updates

When to use Long Context Models OR RAG

- When to Use Long Context Models:
 - Smaller Datasets, Complex Analysis, Tolerable Delays, Low Volume
- When to Use RAG:
 - Large Datasets, Fast Responses, Cost-Effectiveness, Transparency

Question: Which model will you use

- Real-time customer support using dynamic knowledge bases -> Long context Model or RAG
 - Ans: RAG
- Deep analysis of a single policy document -> Long Context Model or RAG
 - Ans: Long Context Models

RAG Implementation with Frameworks

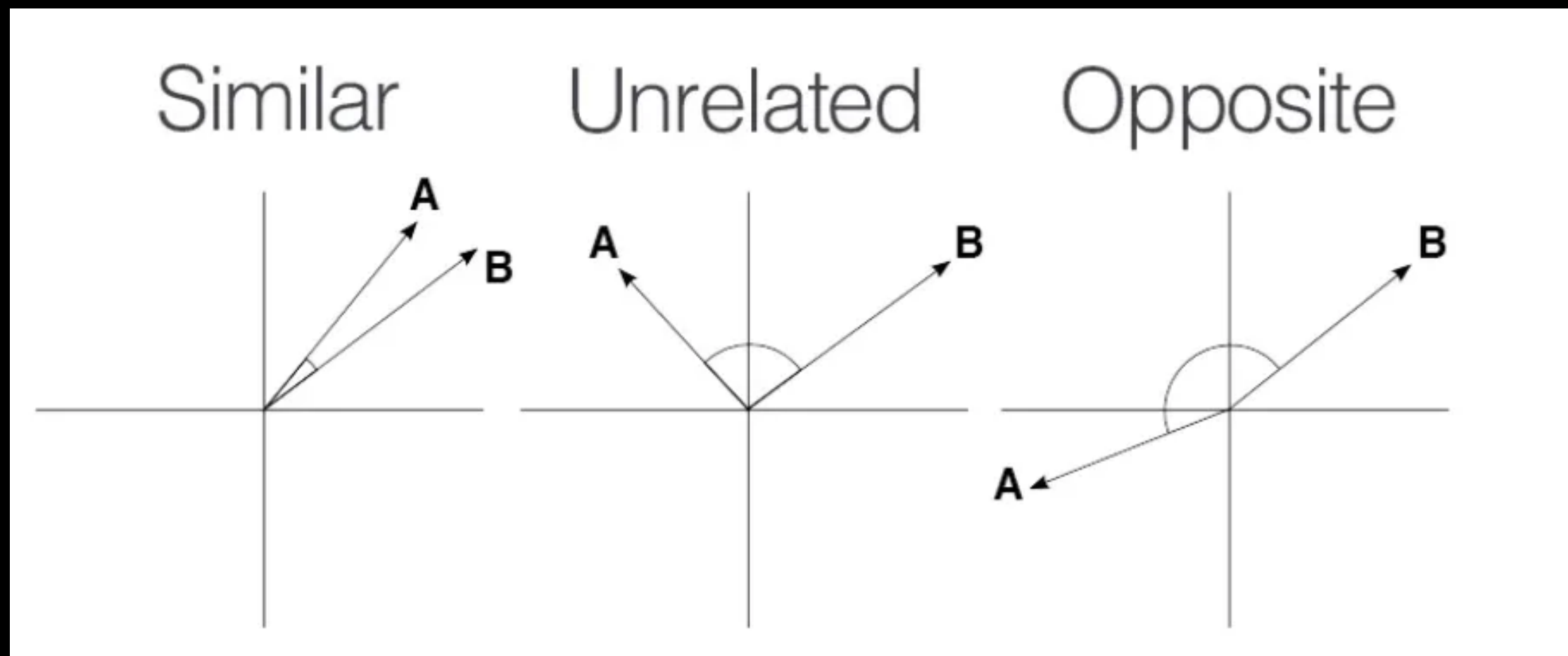
- LangChain and LlamaIndex
 - Offer user-friendly classes for implementing a retriever on your data source, with the first step being index creation for your data

Embeddings

- For Textual Data
 - GPT series and Llama can create vector embeddings for words, sentences, or paragraphs within their internal layers
- For Image Data
 - CNNs can produce embeddings for image data by extracting the information at a specific network layer
- Audio data
 - image embedding techniques to visual representations of audio frequencies, such as spectrograms

0.2 0.8 0.1 0.5 0.9 0.6 0.2 0.3 0.4 0.7

Cosine Similarity



$$\text{Cosine Similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}, \quad -1 \leq \cos(\theta) \leq 1$$

$$\text{Where: } \begin{cases} \cos(\theta) = 1 & \text{if the vectors are identical (angle } \theta = 0^\circ) \\ \cos(\theta) = 0 & \text{if the vectors are orthogonal (angle } \theta = 90^\circ) \\ \cos(\theta) = -1 & \text{if the vectors are opposite (angle } \theta = 180^\circ) \end{cases}$$

The Role of Embeddings in Similarity Search

- Embedding is the key to similarity search tasks
- Applications: de-duplication, recommendation systems, anomaly detection, and reverse image searching

Vector Databases

keep and manage embeddings

Vector Databases: Powering Semantic Search

- Specialized databases for storing and managing embeddings
- Key Features:
 - Efficient Storage: Handles large volumes of complex data as embeddings
 - Semantic Search: Compares query embeddings with stored embeddings
- Why Use Vector Databases?
 - Fast and relevant retrieval of similar items. Understands the intent behind queries, independent of wording

Revolutionizing Medicine with RAG

Med-PaLM

- A large language model from Google Research, designed for the medical domain
- <https://sites.research.google/med-palm/>
- Med-PaLM 1: Preprinted in 2022; published in Nature (July 2023)
 - First AI to exceed the 60% pass mark on U.S. Medical Licensing Exam (USMLE)-style questions
- Med-PaLM 2: Launched in March 2023
 - Sets new standards for expert-level medical reasoning.

Medical evaluation challenge

- MACCROBAT dataset
- <https://www.kaggle.com/datasets/okolojeremiah/maccrobat>
- A dataset of detailed patient medical reports from PubMed Central, enriched with annotated information
- The dataset aims to advance healthcare analytics and improve patient care and medical research

Building a Medical RAG Pipeline from Scratch

An application that can sift through patient data and answer questions



15939911.txt and 15939911.ann

- CASE: A 28-year-old previously healthy man presented with a 6-week history of palpitations. The symptoms occurred during rest, 2–3 times per week, lasted up to 30 minutes at a time and were associated with dyspnea. Except for a grade 2/6 holosystolic tricuspid regurgitation murmur (best heard at the left sternal border with inspiratory accentuation), physical examination yielded unremarkable findings.
- An electrocardiogram (ECG) revealed normal sinus rhythm and a Wolff– Parkinson– White pre-excitation pattern (Fig.1: Top), produced by a right-sided accessory pathway.
- Transthoracic echocardiography demonstrated the presence of Ebstein's anomaly of the tricuspid valve, with apical displacement of the valve and formation of an “atrialized” right ventricle (a functional unit between the right atrium and the inlet [inflow] portion of the right ventricle) (Fig.2).
- The anterior tricuspid valve leaflet was elongated (Fig.2C, arrow), whereas the septal leaflet was rudimentary (Fig.2C, arrowhead).
- Contrast echocardiography using saline revealed a patent foramen ovale with right-to-left shunting and bubbles in the left atrium (Fig.2D).
- The patient underwent an electrophysiologic study with mapping of the accessory pathway, followed by radiofrequency ablation (interruption of the pathway using the heat generated by electromagnetic waves at the tip of an ablation catheter).
- His post-ablation ECG showed a prolonged PR interval and an odd “second” QRS complex in leads III, aVF and V2–V4 (Fig.1Bottom), a consequence of abnormal impulse conduction in the “atrialized” right ventricle.
- The patient reported no recurrence of palpitations at follow-up 6 months after the ablation.

21067996.txt and 21067996.ann

- A 41-year-old man with weight loss, nausea, vomiting, and abdominal pain was referred for an abdominal CT scan to rule out intraabdominal pathology.
- A dedicated CT of the abdomen was performed at our institution according to our standard venous phase protocol.
- Following ingestion of positive oral contrast material (barium sulfate suspension 2.1% w/v, Mallinckrodt) and injection of 125ml of iopromide, 300mgI/mL (Ultravist, Bayer HealthCare), helical CT was obtained with a 65-s delays using a 64-slice GE Lightspeed VCT scanner (GE Medical Systems; Milwaukee, WI).
- Axial sections of the abdomen showed a large (18×22×22 cm) heterogeneous retroperitoneal mass probably arising from the left adrenal gland containing multiple areas of macroscopic fat (Fig.1).
- The patient had no documented clinical or laboratory evidence of adrenal hormonal excess.
- On the basis of the clinical history and CT findings, the patient underwent surgical resection of the left adrenal mass, which was found on pathologic examination to represent a low-grade adrenocortical carcinoma.

Medical RAG Pipeline: imports

```
import os
import glob
import ast
import pandas as pd
import numpy as np
import pprint

from tqdm.notebook import tqdm

import openai
from sklearn.metrics.pairwise import cosine_similarity
```

Medical RAG Pipeline

```
if os.getenv('OPENAI_API_KEY'):  
    print("API key retrieved successfully.")  
else:  
    print("""API key not found""")  
  
model_embedding = "text-embedding-3-small" # 1536  
#model_embedding = "text-embedding-3-large" # 3072
```

Generate Embeddings

```
#Generate Embeddings
client = openai.OpenAI()

# text to embedding vector using OpenAI Embedding Model
def text_to_embedding(text, model=model_embedding):
    try:
        res = client.embeddings.create(input=[text], model=model)
        return res.data[0].embedding
    except:
        return
```

Sample Question and Answers

```
# Define the user question, and convert it to embedding.  
question = "What are the symptoms for a patient with  
repeated hemoptysis."
```

```
answer_1 = "Cloud computing optimizes data storage and  
processing but has no role in treating hemoptysis."
```

```
answer_2 = "There can be thoracic pain that intensify while  
breathing, as well as a strong dry cough that may persist."
```

Get Embedding

```
question_emb = text_to_embedding(question)
answer_1_emb = text_to_embedding(answer_1)
answer_2_emb = text_to_embedding(answer_2)

print(len(question_emb), len(answer_1_emb), len(answer_2_emb))
#3072 3072 3072
```

Cosine Similarity

```
print("Question -> Answer 1: ",  
      cosine_similarity([question_emb], [answer_1_emb]))  
print("Question -> Answer 2: ",  
      cosine_similarity([question_emb], [answer_2_emb]))
```

3072 3072 3072

Question -> Answer 1: [[0.37250219]]

Question -> Answer 2: [[0.43499012]]

MACCROBAT

```
project_name = "kaggleMACCROBAT"  
  
csv_file = f"./data/{project_name}.csv"  
embeddings_file = f"./data/{project_name}_embedding.csv"
```

Combine text files to CSV

```
def create_csv(csv_file):
    if(os.path.exists(csv_file)):
        return
    files = glob.glob(f"./data/kaggleMACCROBAT/*.txt")
    rows = []
    file_names = []
    for file in tqdm(files):
        file_name = os.path.basename(file)
        with open(file, 'r', encoding='utf-8') as f:
            content = f.read()
            rows.append(content)
            file_names.append(file_name)
    df = pd.DataFrame({
        "file": file_names,
        "content": rows
    })
    df.to_csv(csv_file, index=False)
    print(df.shape)
create_csv(csv_file)
```

	A	B
1	file	content
2	19860925.txt	Our 24-year-old non-smoking male patient presented with repeated hemoptysis in May 2008 with 4 days of concomitant right thoracic pain which intensified while breathing. During holidays in his home country, this Cuban patient suffered from a cold with fever and a strong cough. The strong dry cough persisted after recovery from the cold.
3	26228535.txt	A 25-year-old female patient had noticed left-sided visual loss and amenorrhea 2 years prior to being seen, with right hemiparesis developing 6 months beforehand. Aggravated symptoms (drooping at corner of right lip and bilateral temporal hemianopia) finally prompted hospitalization. At that time, a massive (>5 cm) tumor of intra- and suprasellar location, excluding mesencephalon
4	27773410.txt	A 69-year-old male diabetic patient was admitted with acute left ventricular (LV) failure and was treated with intensive decongestive therapy. He had severe peripheral edema and bilateral basal crepitations. Initial workup revealed normal levels of troponine (<0.01 ng/ml) and CPK-MB (4.6 ng/ml), thereby ruling out acute coronary syndrome.
5	28103924.txt	Our patient was a 7-year-old Italian boy born after an uneventful gestation of normal duration. At the age of 16 months, he presented with a clinically evident enlarged abdomen and was referred for oncological examination. Initial tests revealed anemia, thrombocytopenia, and splenomegaly. A bone marrow biopsy revealed the presence of foam cells, which led to suspicion of lysosomal storage
6	27064109.txt	A 53-year-old man came to our hospital with signs and symptoms of acute heart failure after a 2-week history of progressive breathlessness. He had a history of recurrent skin abscesses and atopic dermatitis and regularly visited a dermatologist in our hospital. On arriving at our hospital, his extremities were warm and dry.
7	20146086.txt	A 54-year-old woman with stage IV NSCLC was treated with carboplatin and paclitaxel without disease response. Molecular analysis of tumor tissue was unavailable at that time. However, her demographic profile (Asian, minimal smoking history, non-small cell histology) predicted her disease would harbor EGFR TKI sensitive cells [2].

Dataset

```
df = pd.read_csv(csv_file)
print(df.shape)
df.head(3)
```

(200, 2)

	file	content
0	19860925.txt	Our 24-year-old non-smoking male patient prese...
1	26228535.txt	A 25-year-old female patient had noticed left-...
2	27773410.txt	A 69-year-old male diabetic patient was admitt...

Split one row into chunks of 1024 chars

```
# Split the input text into chunks of specified
size (1024 characters)
def split_into_chunks(text, chunk_size=1024):
    chunks = []
    for i in range(0, len(text), chunk_size):
        chunks.append(text[i:i+chunk_size])
    return chunks
```

Split all rows

```
def create_chunks(df):  
    chunks = []  
    for content in df["content"]:  
        content = content.replace("\n", " ") # Remove newlines  
        chunks.extend(split_into_chunks(content))  
    return chunks
```

Create chunks from the dataset

```
df = pd.read_csv(csv_file)
chunks = create_chunks(df)
print(len(chunks))
```

651

Generate embedding

```
# Generate embedding
def file_to_embedding(csv_file, embeddings_file):
    if(os.path.exists(embeddings_file)):
        df = pd.read_csv(embeddings_file)
        return df

    df = pd.read_csv(csv_file)
    chunks = create_chunks(df)

    embeddings = []
    for chunk in tqdm(chunks, total=len(chunks), desc="Processing chunks"):
        embedding = text_to_embedding(text=chunk)
        embeddings.append(embedding)
    df_updated = pd.DataFrame({
        "content": chunks,
        "embedding": embeddings
    })

    df_updated.to_csv(embeddings_file, index=False)
    return df_updated
```

Embedding for all Chunks

```
df = file_to_embedding(csv_file=csv_file, embeddings_file=embeddings_file)
df.head(5)
```

	chunk	embedding
0	Our 24-year-old non-smoking male patient prese...	[0.012524573132395744, 0.03273537755012512, -0...
1	s were detected in the cytological examination...	[0.002973469439893961, 0.03482300415635109, -0...
2	A 25-year-old female patient had noticed left-...	[-0.015337462536990643, 0.03896137326955795, 0...
3	ot regress, despite a 4-week course of the dop...	[-0.004401150159537792, 0.02939952351152897, -...
4	mposed of polygonal and spindle cells with rou...	[-0.009954688139259815, 0.019837914034724236, ...

Get Context for the Query

convert_embedding to np.array

```
def convert_embedding(x):  
    # check whether the embedding is stored as a string  
    if isinstance(x, str):  
        # convert the string representation to a python list  
        x = ast.literal_eval(x)  
        # convert the list to a numpy array  
        return np.array(x)  
    # convert an existing list or similar object to a numpy array  
    else:  
        return np.array(x)  
  
# apply the conversion to every embedding  
df['embedding'] = df['embedding'].apply(convert_embedding)
```

Get Relevant Indices

```
def get_relevant_indices(df, question, chunks_to_retrieve=3):  
  
    question_emb = text_to_embedding(question)  
  
    # apply the conversion to every embedding  
    df['embedding'] = df['embedding'].apply(convert_embedding)  
    embeddings_list = np.array(df['embedding'].tolist())  
  
    cosine_similarities = cosine_similarity(  
        [question_emb], embeddings_list)  
  
    # sort similarity scores from highest to lowest  
    indices = np.argsort(cosine_similarities[0])[::-1]  
  
    # select the indices of the top n most relevant chunks  
    indices = indices[:chunks_to_retrieve]  
  
    return indices
```

Get Relevant Indices

```
question = "What are symptoms for a patient with  
repeated hemoptysis."
```

```
df = file_to_embedding(csv_file=csv_file,  
embeddings_file=embeddings_file)
```

```
indices = get_relevant_indices(df=df,  
question=question)
```

```
print(indices)
```

```
[553 199 588]
```

Show Resources

```
def show_resources(df, indices):  
    for idx, item_index in enumerate(indices):  
        print(f"> Chunk {idx+1} (Index: {item_index})")  
        item = df.content[item_index]  
        print(f"\"\"\"{item[:80]}\n...\n{item[-80:]}\"\"")  
        print("----")
```

Show resources

```
question = "What are symptoms for a patient  
with repeated hemoptysis."
```

```
df = file_to_embedding(csv_file=csv_file,  
embeddings_file=embeddings_file)
```

```
indices = get_relevant_indices(df=df,  
question=question)
```

```
show_resources(df, indices)
```

Show resources: output

> Chunk 1 (Index: 553)

An 82-year-old woman was transferred to our hospital because of dyspnea and mass
...
she was then extubated. After that, only a small amount of bloody sputum was cou

> Chunk 2 (Index: 199)

A 60-year-old woman patient was admitted to our hospital on Feb. 18, 2016 becaus
...
Tortuous blood vessels were observed running in the submucosa of the trachea (F

> Chunk 3 (Index: 588)

The patient was a 48-year-old man who had previously been hospitalized due to he
...
ve-mentioned examinations, DAH due to some sort of vasculitis was suspected. Thu

Get Context

```
def get_context(df, indices):  
    context = " ".join(df.content[indices])  
    return context
```

Get Context for the question

```
question = "What are symptoms for a patient with  
repeated hemoptysis."  
  
df = file_to_embedding(csv_file=csv_file,  
embeddings_file=embeddings_file)  
  
indices = get_relevant_indices(df=df, question=question)  
  
#show_resources(df, indices)  
context = get_context(df, indices)  
  
pprint.pprint(context)
```

Get Context for the question: output

```
('An 82-year-old woman was transferred to our hospital because of dyspnea and '
'massive hemoptysis. The patient had a history of bronchial asthma that was '
'well-controlled with bronchodilator medications. She had no history of '
'tuberculosis, nontuberculous mycobacterial infection, or smoking. The '
'physical examination revealed diffuse bilateral crackles. She suffered from '
'severe hypoxemia (pH 7.362, PCO2 35.1 mmHg, PO2 61.0 mmHg, HCO3- 20.1 mmHg, '
'BE -5.5 mmHg, SpO2 90.5%, under 10 L O2/min, reservoir mask). After the '
'tracheal intubation, 100 mL of bright-red blood was aspirated. A chest '
'radiograph showed bilateral infiltrates (Fig.1). A chest CT further '
'demonstrated multiple consolidations and ground glass opacity and focal '
'bronchiectasis in right segment 4 (S4) (Fig.2). There were no '
'space-occupying lesions. Four days after admission, her respiratory '
'condition was improved. Since there was no active hemorrhaging from the '
'tracheal tube, she was then extubated. After that, only a small amount of '
'bloody sputum was cou A 60-year-old woman patient was admitted to our '
'hospital on Feb. 18, 2016 because of frequent episodes of hemoptysis for 2 '
'weeks. In the previous decade, the patient had frequent but less severe '
'episodes of hemoptysis, which typically ensued following an upper '
'respiratory tract infection. Ear, nose, and throat examination and '
'laryngoscope failed to identify an apparent source of bleeding. CT chest '
'scan revealed no abnormality. The most recent episode occurred after an '
'upper respiratory tract infection 2 weeks ago. The patient coughed up as '
'much as about 500\ u200amL of fresh blood in an episode of hemoptysis, which '
'was alleviated by anti-tussive therapy. The patient denied a history of '
'smoking and illicit drug use. Diagnostic workup revealed no evidence of '
'. . .
'disease, with the exception of his younger brother who had been diagnosed '
'with hepatitis C. Furthermore, he had not received any drugs that had the '
'potential to cause DAH. Following the above-mentioned examinations, DAH due '
'to some sort of vasculitis was suspected. Thu')
```

Get Prompts

```
def get_prompts(question, context):
    system_prompt = """\
You are an assistant and expert in answering questions \
from a chunks of content. \
If you cannot answer from given chunk, \
say that you cannot answer this question. \
"""

    user_prompt = f"""\
Read the following informations that might contain \
the context you require to answer the question. \
You can use the informations starting from the \
<START_OF_CONTEXT> tag and end with the <END_OF_CONTEXT> tag. \

Here is the content:
    <START_OF_CONTEXT>
    {context}
    <END_OF_CONTEXT>

Provide an informative and accurate answer to the \
following question based on the available context.

Question: {question}
Answer: \
"""

    return system_prompt, user_prompt
```

Create Messages from Prompts

```
def create_messages(system_prompt, user_prompt):  
    return [  
        {"role": "system", "content": system_prompt},  
        {"role": "user", "content": user_prompt},  
    ]
```

Get Response from GPT-4o based on the prompts

```
def generate_chat_response(system_prompt, user_prompt,
model="gpt-4o-mini", temperature=0):

    client = openai.OpenAI()

    messages = create_messages(system_prompt, user_prompt)

    #pprint.pprint(messages)
    completion = client.chat.completions.create(model=model,
                                                messages=messages,
temperature=temperature)

    return completion.choices[0].message.content
```

Ask Question using RAG

```
question = "What are symptoms for a patient with repeated hemoptysis."

df = file_to_embedding(csv_file=csv_file ,embeddings_file=embeddings_file)

indices = get_relevant_indices(df, question=question)
#show_resources(df, indices)

context = get_context(df, indices)
#pprint.pprint(context)

system_prompt, user_prompt = get_prompts(question=question,
context=context)

answer = generate_chat_response(system_prompt=system_prompt,
user_prompt=user_prompt)

pprint.pprint(answer)
```

RAG: output

('The symptoms for a patient with repeated hemoptysis, as indicated in the ' provided context, include episodes of coughing up blood, which can vary in ' volume. For example, one patient coughed up as much as about 500 mL of fresh ' blood during an episode. Additionally, physical examination may reveal signs ' such as rhonchi and reduced breath sounds. Other symptoms may include ' dyspnea (difficulty breathing) and signs of respiratory distress, as seen in ' the case of the 82-year-old woman who experienced severe hypoxemia and ' required oxygen support.')

Ask Question using RAG

```
question = "Is the bleeding possible in case of hemoptysis."

df = file_to_embedding(csv_file=csv_file , embeddings_file=embeddings_file)

indices = get_relevant_indices(df, question=question)
#show_resources(df, indices)

context = get_context(df, indices)
#pprint.pprint(context)

system_prompt, user_prompt = get_prompts(question=question,
context=context)

answer = generate_chat_response(system_prompt=system_prompt,
user_prompt=user_prompt)

pprint.pprint(answer)
```

RAG: output

('Yes, bleeding is possible in the case of hemoptysis, as hemoptysis is ' 'defined as the coughing up of blood from the respiratory tract. The provided ' 'context describes two cases of patients experiencing hemoptysis, with one ' 'patient having massive hemoptysis and the other having frequent episodes of ' 'hemoptysis. In both cases, the presence of blood indicates that bleeding is ' 'occurring within the respiratory system.')

Ask Question using RAG

```
question = "What is Artificial Intelligence?"

df = file_to_embedding(csv_file=csv_file ,embeddings_file=embeddings_file)

indices = get_relevant_indices(df, question=question)
#show_resources(df, indices)

context = get_context(df, indices)
#pprint.pprint(context)

system_prompt, user_prompt = get_prompts(question=question, context=context)

answer = generate_chat_response(system_prompt=system_prompt,
user_prompt=user_prompt)

pprint.pprint(answer)
```

'I cannot answer this question.'

What is important

- Prompt Engineering
- AI is not like Google or Bing Search Engine
- AI needs more and relevant context to get better answers and reasoning

Challenges of RAG Systems

Challenges of RAG Systems

- Chunking, dealing with multimodality, updating documents, compliance, and evaluating the RAG performance.

Challenges of RAG Systems

- Small chunks might miss the connection: continuously test and refine chunking methods based on application needs.
- The multimodal nature of documents (text and image) and their representation in the same latent space can be difficult.
- Implementing dynamic updating mechanisms for vectors.
- Compliance is another crucial aspect.

Challenges of RAG Systems

- An important aspect to consider in some retrieval systems is the balance between latency and quality.
- Implementing a combination of different methods, such as keyword and embedding retrieval complemented with reranking, can deliver precise results at the cost of a bigger latency.

Retrieval step in RAG system critical for the overall effectiveness

LlamaIndex framework

Hybrid Search Versus Embedding Search

- The embeddings search may not always be the most effective method for.
- Keyword lookup + Embedding search for context
- This approach balances the specificity of keyword searches and the contextual understanding of embeddings.

Enhancing Retrieval with LlamaIndex Framework

- Combines keyword and embedding search for precise query results
- Reranking prioritizes results based on recency and relevance.
- Summary-based indexing highlights key information.
- Enhancing chunks with metadata adds context and improves retrieval accuracy.
- Organizing data with metadata filters is also advantageous for structured retrieval, ensuring that relevant chunks are efficiently retrieved.
- Establishing node relationships between chunks assists specific retrieval algorithms.

Fine-tuning the embedding model

- Better embedding model =>
- GPT-4 while using more economical options like GPT-3.5

Monitoring and Evaluating RAG

- accuracy, semantic similarity, and reliability
- https://docs.llamaindex.ai/en/stable/module_guides/evaluating/

Key Metrics for Evaluating AI-Generated Answers

- Correctness
 - Assesses alignment with a predefined reference answer
 - Requires labeled data to verify accuracy
- Faithfulness
 - Ensures the answer accurately reflects retrieved-context
 - Avoids distortions or fabrications of source material

Key Metrics for Evaluating AI-Generated Answers

- Context Relevancy
 - Measures the relevance of retrieved-context to the query
 - Aims to retrieve and utilize information closely tied to the user's request
- Guideline Adherence
 - Checks compliance with predefined criteria (stylistic, factual, ethical)
 - Ensures responses meet established norms and effectively address the query

Key Metrics for Evaluating AI-Generated Answers

- Embedding Semantic Similarity
 - Uses similarity scores between embeddings of generated and reference answers.
 - Helps evaluate semantic closeness to reference content.

Best Practices for Building Reliable RAG Systems

Best Practice: Incorporate a Vector Database

- Efficiently handles multiple queries in real-time
- Reduces latency by quickly retrieving relevant embeddings
- Ideal for high-demand applications like real-time chatbots and dynamic search engines

Best Practice: Use Guardrails for Responsible AI

- Content Filters: Restrict the model from discussing sensitive topics like politics or answering biased questions
- Language Customization: Instruct the model to adopt a specific tone or style (e.g., formal, conversational)

Best Practice: Privacy Protection Measures

- Sanitize Input: Remove sensitive or personal information before processing.
- Retrieval Guardrails: Ensure only authorized, non-confidential data is retrieved.
- Minimize Hallucinations: Implement mechanisms to verify the accuracy and reliability of responses.

Thank you
naneja@vt.edu