UNSTRUCTURED DATA SUMMARIZATION AND INTELLIGENT SEARCH

USING INTELLIGENT DATA RETRIEVAL TO QUERY AND GAIN DEEPER INSIGHTS ABOUT YOUR DATA

BUSINESS PROBLEM

- Within many organizations, there are abundant sources of unstructured data with no real way to search, summarize and intelligently query that data.
- Those sources of data may be product support manuals that customers and support teams utilize, or it may data from a public facing customer support forum.
- Given these vast oceans of unstructured data, how does an organization synthesize that data to provide intelligent insights and provide customer value?

BUSINESS PROBLEM

- For organizations trying to solve this problem, the traditional method might be to simply utilize the built in search functions in their various tools.
- This approach doesn't solve the problem of data sprawl, and many tools lack intelligent search functionality
- Traditional search leaves users with a list of documents in which they must extract and combine the information themselves.

BUSINESS PROBLEM

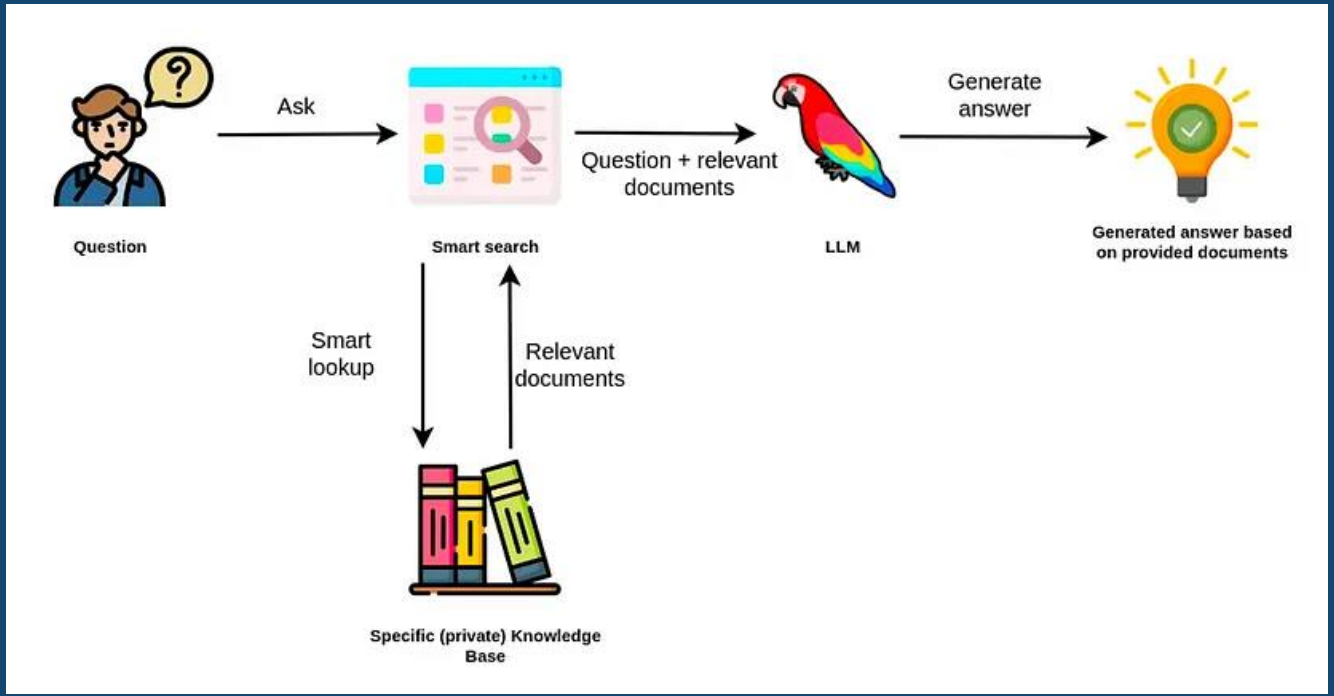
- For organizations that are beginning their AI journey, they may believe that finetuning LLMs on their unstructured data will be a solution to their problems.
- With the changing nature of organizational data, finetuning LLMs is impractical, expensive, and inflexible and doesn't lend itself well to use cases that might need accurate information (LLM hallucinations).
- So, what is the solution?

SOLUTION

- **Vector Databases** provide a way to store metadata and embedding information about unstructured data.
- Utilizing **Vector Search** (Semantic Search), organizations can quickly query their data to find relevant insightful content.
- By combining Language Models (LLMs) with the results of Vector semantic search, organizations can easily summarize, query, and gain deeper insights about their data.
- This approach enables more complex data retrieval that is not limited to keywords and synonyms and can deliver a true **multi-modal search experience.**

SOLUTION

- This process of combining Vector Search and LLMs is known as retrieval augmented generation (RAG)



BENEFITS

- Multimodal Search: Utilizing RAG pipelines with audio/video transcription as well as OCR technologies can help organizations gain deeper insights that traditional methods may miss.
- Contextual Understanding: RAG doesn't just retrieve documents; it understands the context of the query and generates a response that directly addresses the user's needs. It can combine information from multiple sources to create a coherent, context-aware answer. This can greatly increase customer satisfaction and help drive down support costs.

BENEFITS

- Real-Time and Up-to-Date Information: As organizational data grows and changes, RAG can retrieve and generate responses based on the most recent information available. This is crucial for support teams when new products releases and they need to find the latest product information.
- Improved User Experience: RAG provides a more intuitive and seamless user experience by allowing users to interact with the system in a conversational manner. It reduces the effort needed to extract useful information, making the retrieval process more efficient and user-friendly.

TECHNOLOGY OVERVIEW

TECHNOLOGIES

Hugging Face

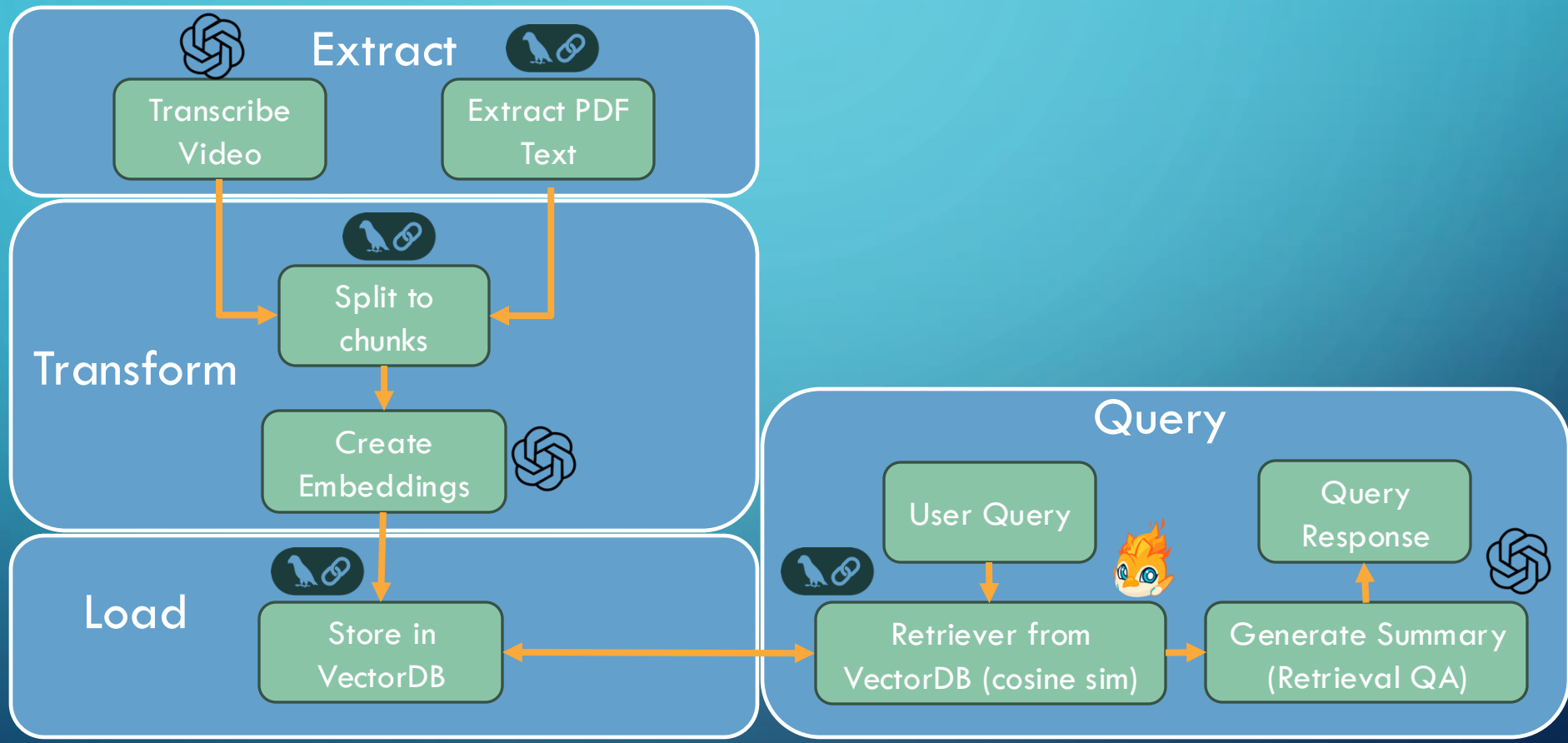
PostgreSQL

LangChain

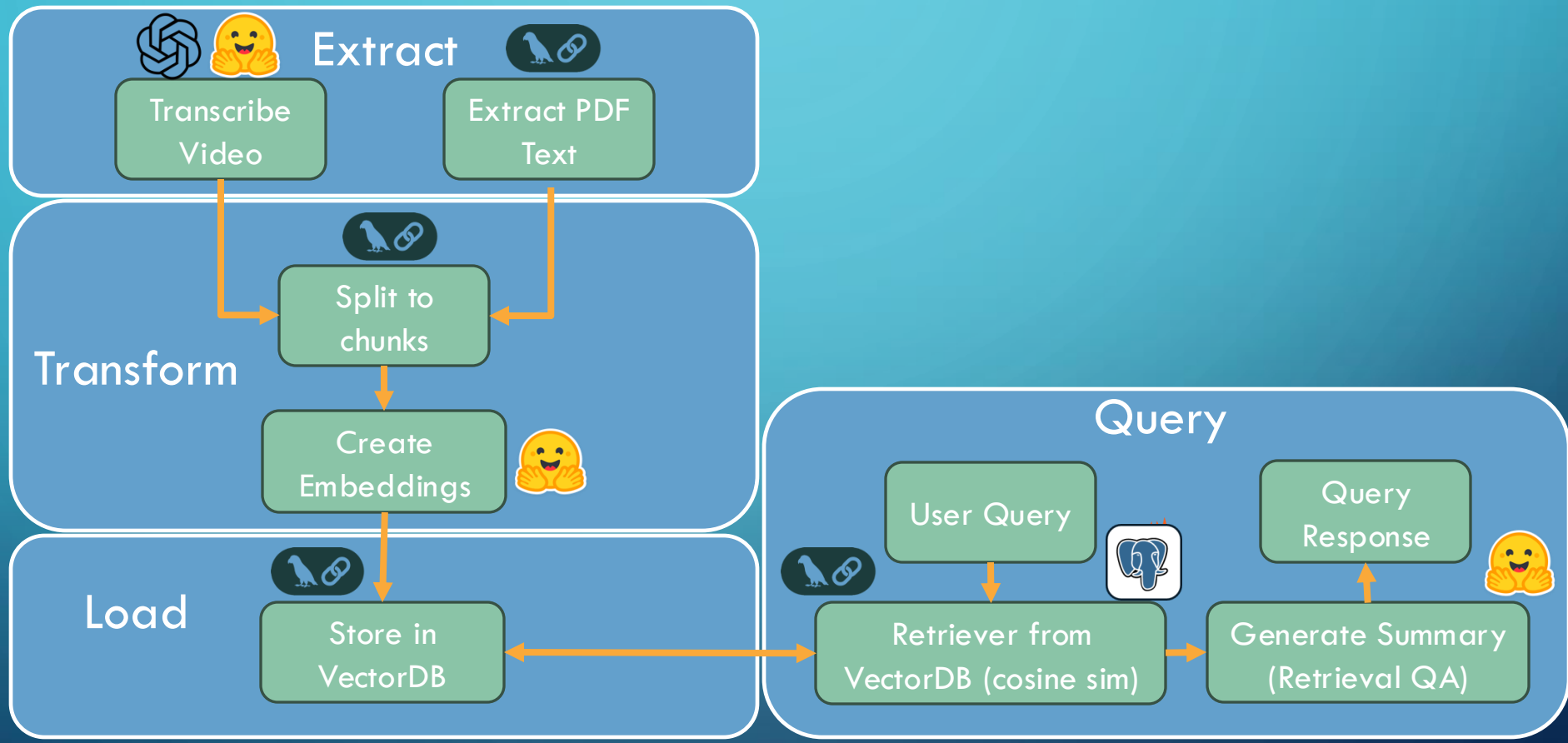
TECHNOLOGIES

(ON-PREM)

SUMMARIZATION AND SEARCH PIPELINE



SUMMARIZATION AND SEARCH PIPELINE (ON-PREM)



HITACHI VANTARA DATA SOURCES

- Video: Smart Cities: How Digital Solutions are Transforming Urban Life
- Video: Smart City Safety: Leveraging Technology to Protect Communities
- Video: Optimizing City Movement with People Flow Analysis
- Video: Smart Parking for Smart Cities: The Future of Urban Parking Solutions
- Video: Smart Transportation: The Key to Sustainable and Efficient Cities
- Video: Smart Traffic Management: Keeping Cities Moving in the 21st Century
- PDF: Smart City Operational Intelligence for Smarter Communities
- PDF: On the Future of Cities. An Industry Roundtable by Hitachi
- PDF: Communities Thrive with Hitachi Smart Spaces and Video Intelligence

DEMO

RESULTS: QUERY/SEARCH

”Summarize the mentions of Public Safety according to Hitachi Vantara”

```
- Hitachi Smart Spaces and Video Intelligence provide tools for public safety officials to protect citizens.
- The technology enables faster response to incidents and better preparation for threats.
- It improves overall situational awareness for public safety officials.
- Video analytics transform security cameras into city sensors for smarter operational intelligence and security.
- The Hitachi Visualization Suite helps public safety officials find necessary assets in real time by correlating disparat
e data and video systems geospatially.
- Automated security solutions protect people, property, and critical infrastructures.
- The solutions help create safer, smarter, and more efficient communities and public spaces.
- Privacy protection ensures compliance with regulations while enhancing public safety.
```

”Summarize how Hitachi Vantara utilizes Lidar in their smart city solutions?”

```
- **Privacy Protection**: Hitachi Vantara uses 3D LiDAR Sensors to ensure privacy protection for citizens, minimizing liab
ility for organizations by preventing unnecessary intrusions into personal privacy.
- **Technology**: LiDAR measures the time of flight (ToF) of laser points to build three-dimensional, real-time informatio
n about the physical world.
- **Applications**: Widely used in autonomous vehicles and geographical mapping from drones, and now adapted for smart spa
ces such as retail, airports, event spaces, facilities, and healthcare campuses.
- **Cost Efficiency**: Innovations have dramatically reduced costs, making LiDAR more accessible for various applications.
- **Data Insights**: Combined with machine learning techniques, LiDAR data provides valuable insights for enhancing operat
ions, safety, and customer experience.
- **Granular Resolution**: Hitachi 3D LiDAR Sensor offers granular resolution and close-range data, which can be stitched
together from multiple devices to provide full coverage.
```

DISCUSSION

DISCUSSION AND QUESTIONS

- LLMs by themselves provide no organizational value in terms of customer support and satisfaction. They need to be fed with the appropriate data that will benefit support teams and customers.
- The solution to this is not finetuning a foundational model but rather feeding it the correct data via a RAG pipeline. RAG provides the flexibility organizations need to address their growing and changing data while address customer support needs.
- The work horse behind RAG is the Vector DB and its ability to perform semantic searching of textual data. The LLMs help stitch the vector search results into an intelligent and coherent answer/response.
- VectorDB (with a little help from the LLMs) are the future to building next generation multimodal search applications that will help lower support costs and help customers quickly find the answers to their problems.

THANK YOU

APPENDIX: CODE SNIPPETS

APPENDIX: CODE SNIPPETS

Prompt Template

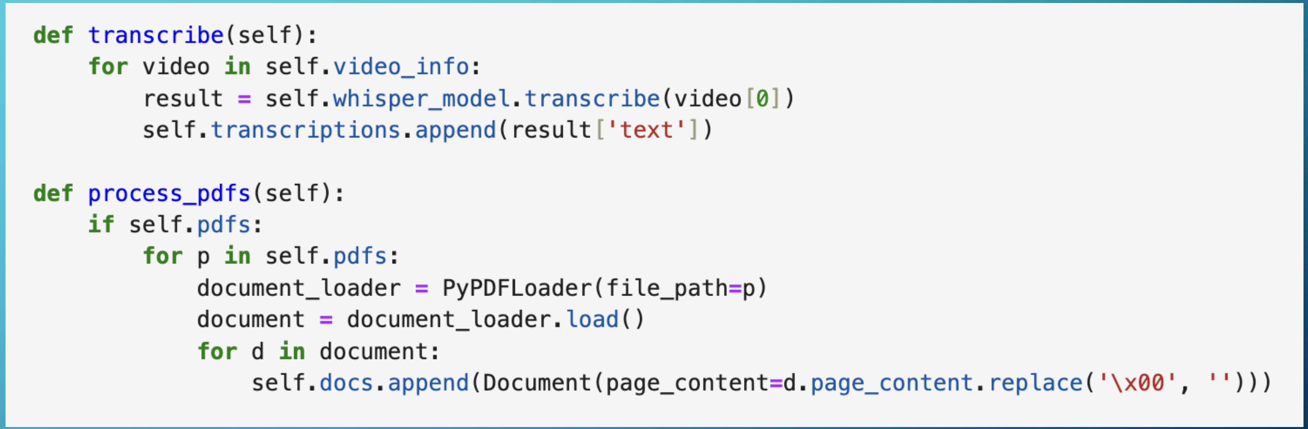
- Tell the LLM to create a summary based on the the semantic search results from the Vector DB.

```
"""Use the following pieces of transcripts from a video to answer the
   question in bullet points and summarized. If you don't know the answer,
   just say that you don't know, don't try to make up an answer.

   {context}

   Question: {question}
   Summarized answer in bullter points:"""
```

APPENDIX: CODE SNIPPETS

Text Extraction

- Transcribe the videos
- Extract text from PDFs



```
def transcribe(self):
    for video in self.video_info:
        result = self.whisper_model.transcribe(video[0])
        self.transcriptions.append(result['text'])

def process_pdfs(self):
    if self.pdfs:
        for p in self.pdfs:
            document_loader = PyPDFLoader(file_path=p)
            document = document_loader.load()
            for d in document:
                self.docs.append(Document(page_content=d.page_content.replace('\x00', '')))
```

APPENDIX: CODE SNIPPETS

Process Text

- Split the text and convert to an Langchain document object

```python
def process_text(self):
    if len(self.transcriptions) > 0:
        text = "\n".join(self.transcriptions)
        text_splitter = RecursiveCharacterTextSplitter(
            chunk_size=1000, chunk_overlap=0, separators=[" ", ",", "\n"]
        )
        texts = text_splitter.split_text(text)
        self.docs = [Document(page_content=t) for t in texts]
    else:
       print("Error: No transcriptions. Transcribe a video first.")
```

APPENDIX: CODE SNIPPETS

Vector Embeddings

- Define vector embedding method
- Define Vector DB store
- Send embeddings to Vector DB store

```
self.embeddings = OpenAIEmbeddings(model=embed_model)
```

```
self.dataset_path = f"hub://{self.activeloop_org_id}/{self.activeloop_dataset_name}"
self.db = DeepLake(dataset_path=self.dataset_path, embedding=self.embeddings, overwrite=True)
```

```
def to_vectordb(self):
    if self.docs:
        self.db.add_documents(self.docs)
    else:
        print("Error: No docs defined")
```

APPENDIX: CODE SNIPPETS

One Shot Summarization

- Send document chucks to the LLM to be summarized
- Use map reduce method of summarization

```python
def summarize(self):
    if self.docs:
        chain = load_summarize_chain(self.llm, chain_type="map_reduce")
        output_summary = chain.invoke(self.docs)
        return output_summary
    else:
        print("Error: No docs defined")
```

APPENDIX: CODE SNIPPETS

Intelligently Query Vector DB and Summarize

- Query Vector DB using cosine similarity and return top 4 chunks
- Use LLM to craft the summary.

```python
def query(self, q="Summarize the mentions of smart cities according to Hitachi Vantara"):
    retriever = self.db.as_retriever()
    retriever.search_kwargs['distance_metric'] = self.distance_metric
    retriever.search_kwargs['k'] = self.k
    PROMPT = PromptTemplate(template=self.prompt_template, input_variables=["context", "question"])
    chain_type_kwargs = {"prompt": PROMPT}
    qa = RetrievalQA.from_chain_type(llm=self.llm,
                                     chain_type="stuff",
                                     retriever=retriever,
                                     chain_type_kwargs=chain_type_kwargs)
    return qa.invoke(q)
```