RETRIEVAL-AUGMENTED GENERATION

MODULE 1

Introduction to RAG and LLMs

WHAT IS RAG ?

Core Concepts

Principles

Advantages of RAG over Traditional LLM

Aspect	Traditional LLMs	RAG
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]

Use Cases of RAG

1. Enhanced Content Creation
2. Customer Feedback Analysis
3. Market Intelligence
4. Personalized Recommendations
5. Dialogue Systems and Chatbots

Large Language Models (LLMs) Overview

Definition and Importance in NLP

Key capabilities

Evolution of LLMs

GPT (Generative Pre-trained Transformer)

- Introduced: 2018 by OpenAI
- Key Innovation: First large-scale transformer-based language model, capable of generating human-like text and performing various NLP tasks without task-specific training.
- Impact: Laid the foundation for subsequent advancements in LLMs, setting a new standard for AI-generated content.

GPT-2

- [illegible]
- [illegible]
- [illegible]

GPT-3:

- [illegible]
- [illegible]
- [illegible]

ChatGPT (and GPT-4)

- [illegible]
- [illegible]
- [illegible]

GPT-4o

- [illegible]
- [illegible]
- [illegible]

GPT-4o

- [illegible]
- [illegible]
- [illegible]

GPT-4o Mini

- [illegible]
- [illegible]
- [illegible]

BERT (Bidirectional Encoder Representations from Transformers)

- Introduced 2018 by Google.
- Key features: [illegible]
- Impact: Significant improvements in NLP tasks, setting new benchmarks for understanding and processing

T5 [Text-To-Text Transfer Transformer]

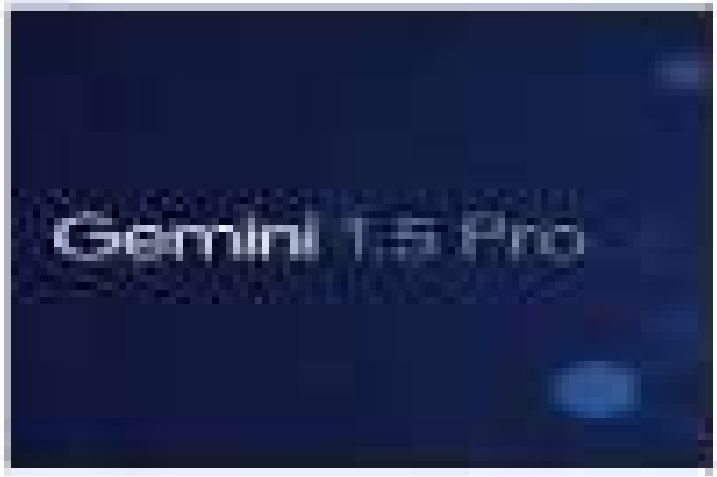
- [illegible]
- [illegible]
- [illegible]

Claude 3.5

- [illegible] state-of-the-art [illegible]
- [illegible] outperforms [illegible] Claude 3 Opus
- [illegible] coding, [illegible] [illegible]

Gemini 1.5

- [illegible]
- [illegible]
- [illegible]



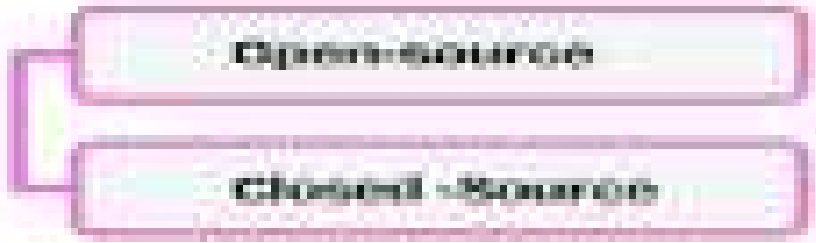
Gemini 1.5 Pro

- [illegible]
- [illegible]
- [illegible]

Current State-of-the-Art

- **[illegible]** [illegible]
- **[illegible]** [illegible]
- **[illegible]** [illegible]

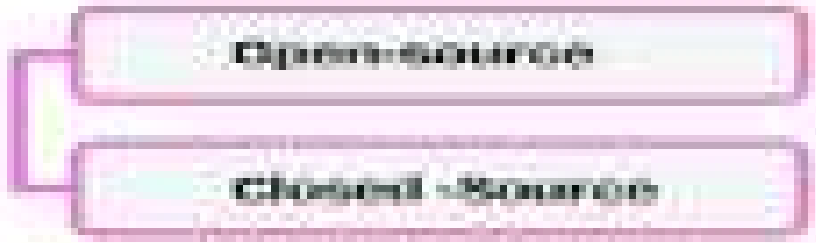
Types of LLMs



Open-source model :- LLama 3.1 by Meta

Open-source model :- Gemma by Google

Types of LLMs



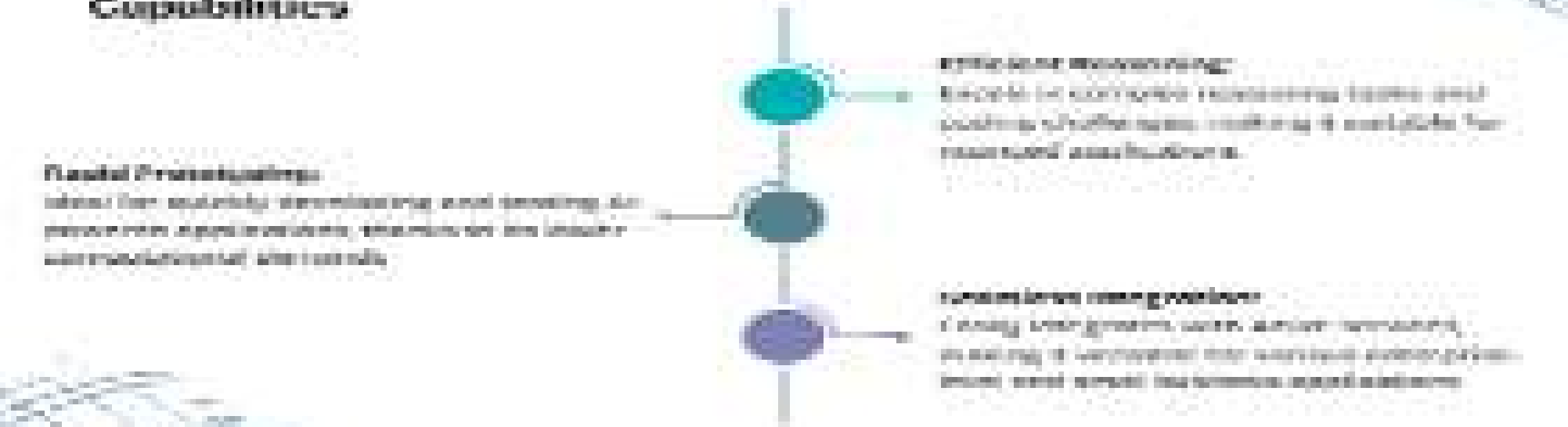
Types of LLMs

Closed -source LLMs

Claude by Anthropic

Closed -Source GPT-4.0 mini Model by OpenAI

Capabilities

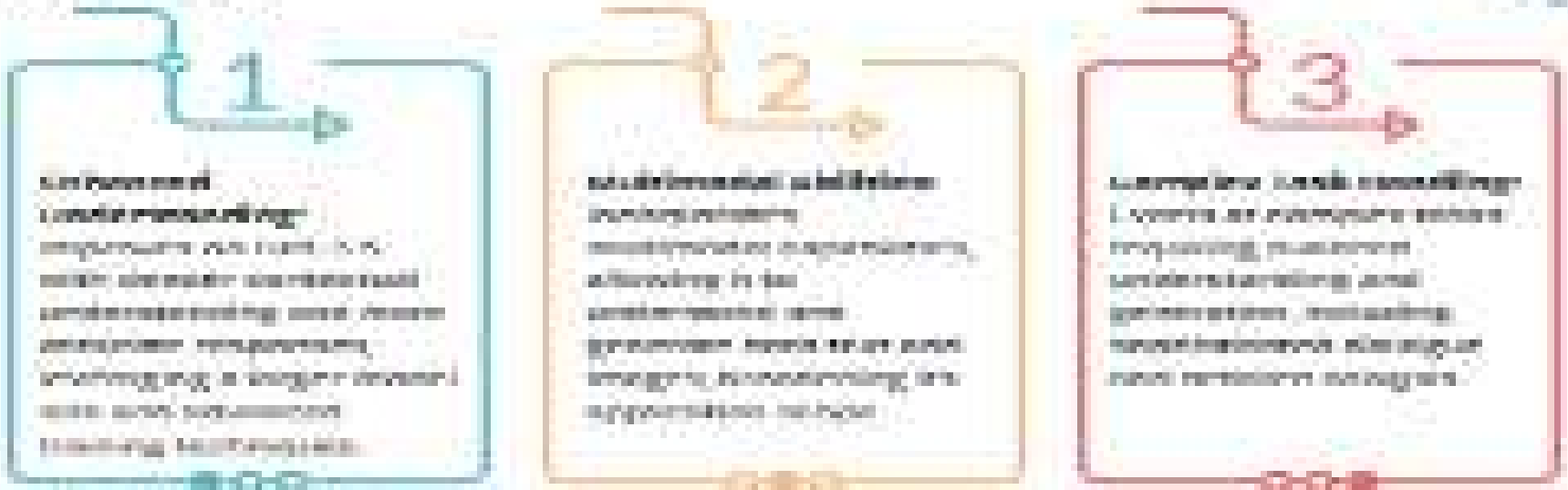


Closed -Source GPT-4.0 mini Model By OpenAI

Use Cases

Closed -Source GPT-4 Model By OpenAI

Capabilities



Closed -Source GPT-4 Model By OpenAI

Use Cases

Real-Time Translators Enables live language translation for instant global communication, cross-cultural collaboration, and multilingual support.

On-Device AI Ideal for personal assistants, offline translation, and privacy-sensitive tasks on smartphones and low-power devices.

Educational Tools Powers AI tutors, language learning, personalized learning experiences.

Comparison Of Open-Source And Closed -Source Models

Aspect	Open-source Models	Closed -source Models
[illegible]	[illegible]	[illegible]
Customizability	[illegible]	[illegible]
Performance	[illegible]	[illegible]
Cost	[illegible]	[illegible]

The Role of RAG in Enhancing LLM Capabilities

Improving Factual Consistency

Examples of LLM Outputs

- [illegible]
- [illegible]

Improving Factual Consistency

How RAG grounds responses in retrieved information

Reducing Hallucinations

Definition of Hallucination in LLM Context

Reducing Hallucinations

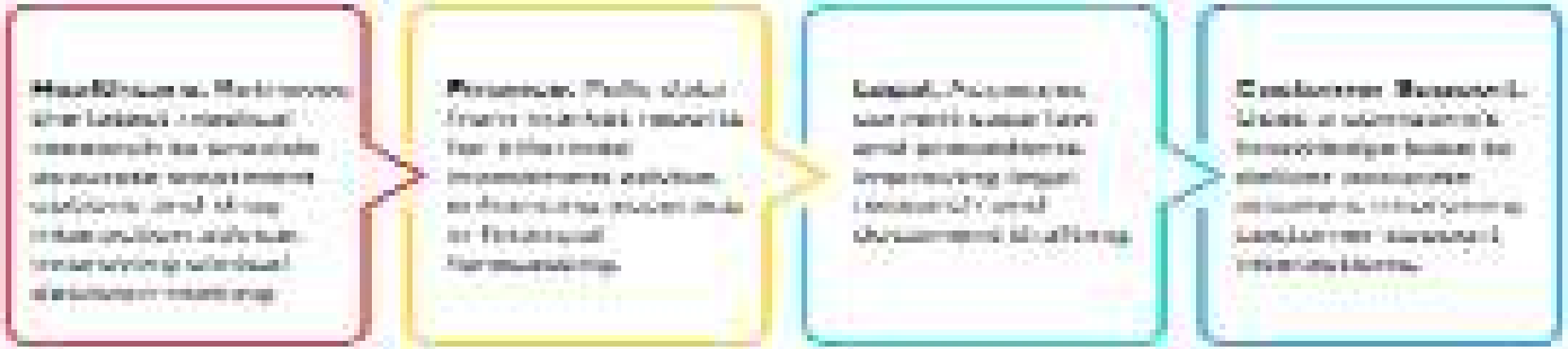
Techniques RAG Uses to Mitigate False Information

Enhancing Domain-Specific Knowledge

Adapting General LLMs to Specialized Fields

Enhancing Domain-Specific Knowledge

Case Studies of RAG in Various Industries



Demo

Running a Basic Open Source LLM Locally

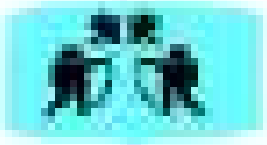
MODULE 2

Vector Databases And Embeddings

Understanding Vector Databases

Key Concepts

Vector Embeddings: [illegible]



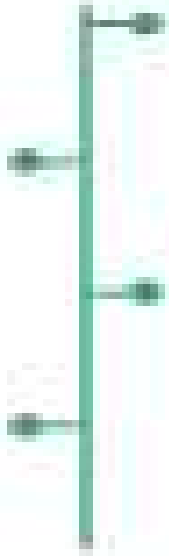
Embedding Models: [illegible]

- [illegible]
- [illegible]
- [illegible]

[illegible]

- [illegible]
- [illegible]
- [illegible]

Core Components of Vector Databases



Role in RAG Systems

Efficient Storage of Document Embeddings

Storage: RAG systems generate embeddings for documents, which are stored in vector databases. These embeddings capture the semantic content of the documents in a format that can be efficiently searched and compared.

Scalability: Vector databases are designed to handle extensive datasets, making them ideal for RAG systems that require storing embeddings for millions of documents or more.

Role in RAG Systems

Fast Similarity Search for Relevant Information Retrieval

Similarity Search: [illegible]

Nearest Neighbor Search: [illegible]

Real-Time Performance: [illegible]

Types Of Vector Databases

Use Cases Beyond RAG

Recommendation systems

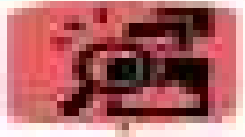
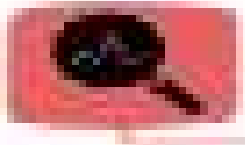


Image search

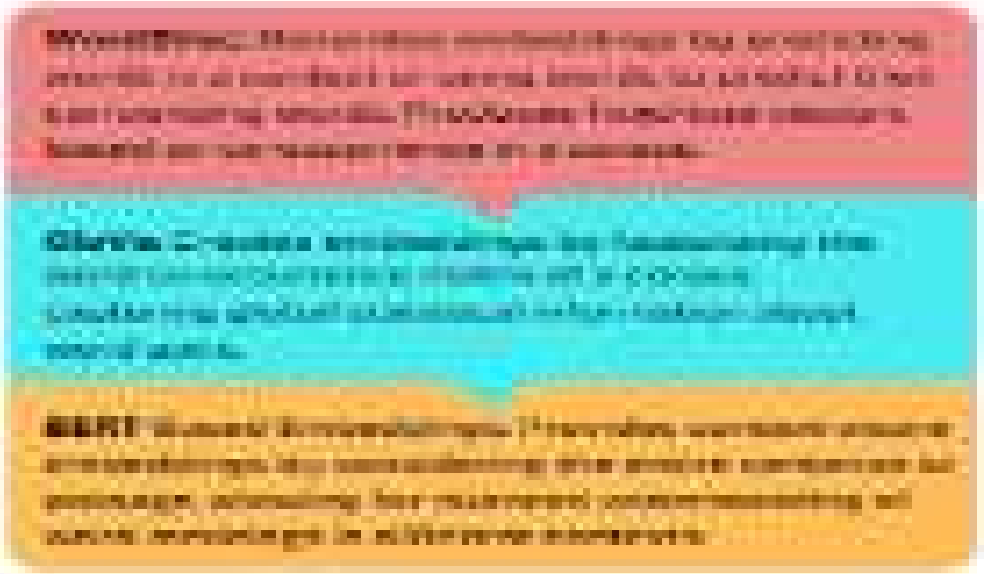


Anomaly Detection

Introduction to Embeddings

How Embeddings Work

Different Embedding Models



GPT Embeddings

Description: GPT models generate embeddings, a way of representing words and sentences as vectors in a high-dimensional space, capturing their semantic relationships and contextual information.

Advantages:

- **Contextual understanding:** Captures nuanced meanings and relationships.
- **Versatility:** Handles various tasks (generation, summarization, translation).
- **Improved performance:** Enhances machine learning models' ability to understand and process language.

Use Cases:

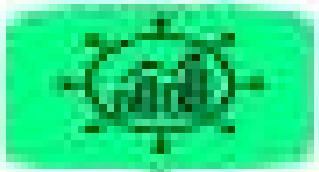
- **Conversational AI:** Enhances chatbots' responses and understanding.
- **Content Generation:** Creates coherent and contextually relevant text.
- **Semantic Search:** Improves search engines by matching queries with relevant content.

Indexing And Similarity Search

Introduction To Indexing and similarity

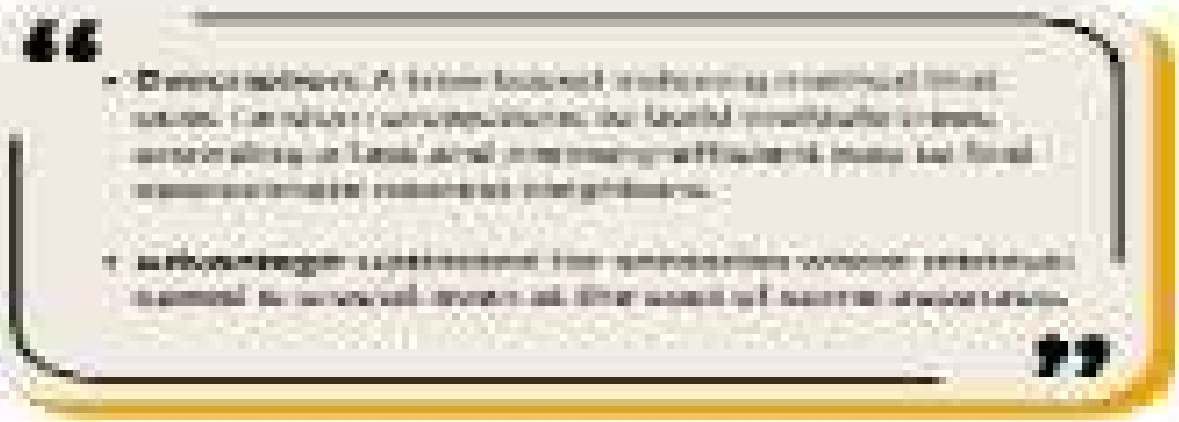
Techniques For Efficient Retrieval

FAISS (Facebook AI Similarity Search)

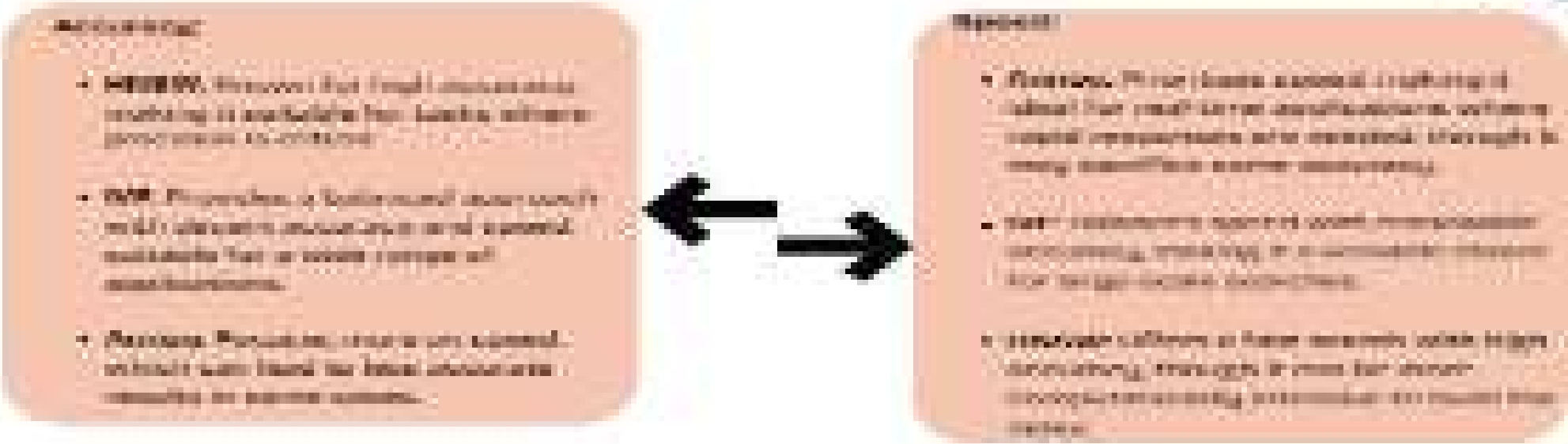


Techniques For Efficient Retrieval

Annoy (Approximate Nearest Neighbors Oh Yeah)



Speed vs. Accuracy Trade-offs



Similarity Metrics

Cosine similarity:

- **Description:** Cosine similarity measures the angle between two vectors, indicating their similarity.
- **When to Use:** Text analysis and high-dimensional spaces where magnitude is less important.

Euclidean distance:

- **Description:** Euclidean distance calculates the straight-line distance between points in a multi-dimensional space.
- **When to Use:** Clustering and classification in low-dimensional spaces.

Dot product:

- **Description:** Dot product measures the projection of one vector onto another, indicating their alignment.
- **When to Use:** Large-scale recommendation systems and neural network models where computational efficiency is crucial.

Demo

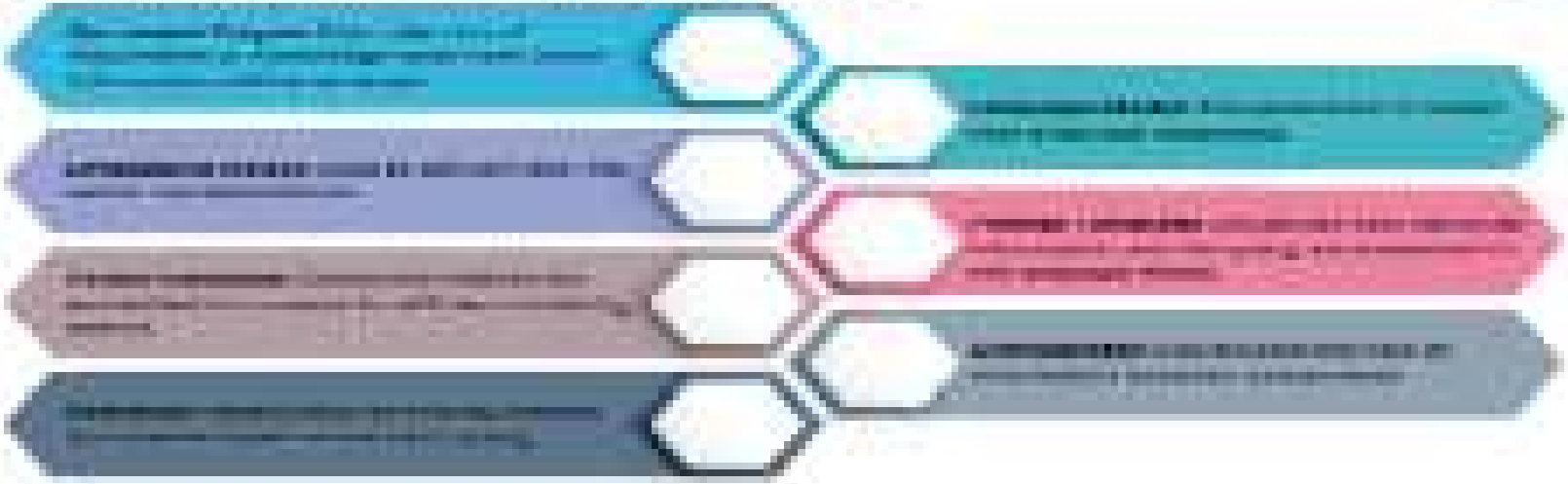
Setting up a Simple Vector Database

MODULE 3

Building a Basic RAG Pipeline

Building a Basic RAG Pipeline

Overview of the RAG Architecture:



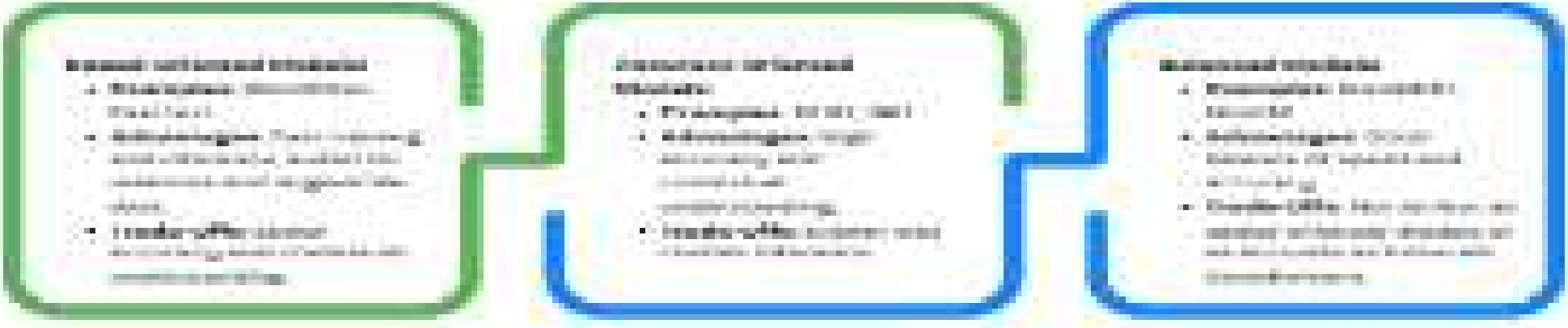
Document Loaders And Text Splitting

Types of Document Loaders

Text Splitting Techniques

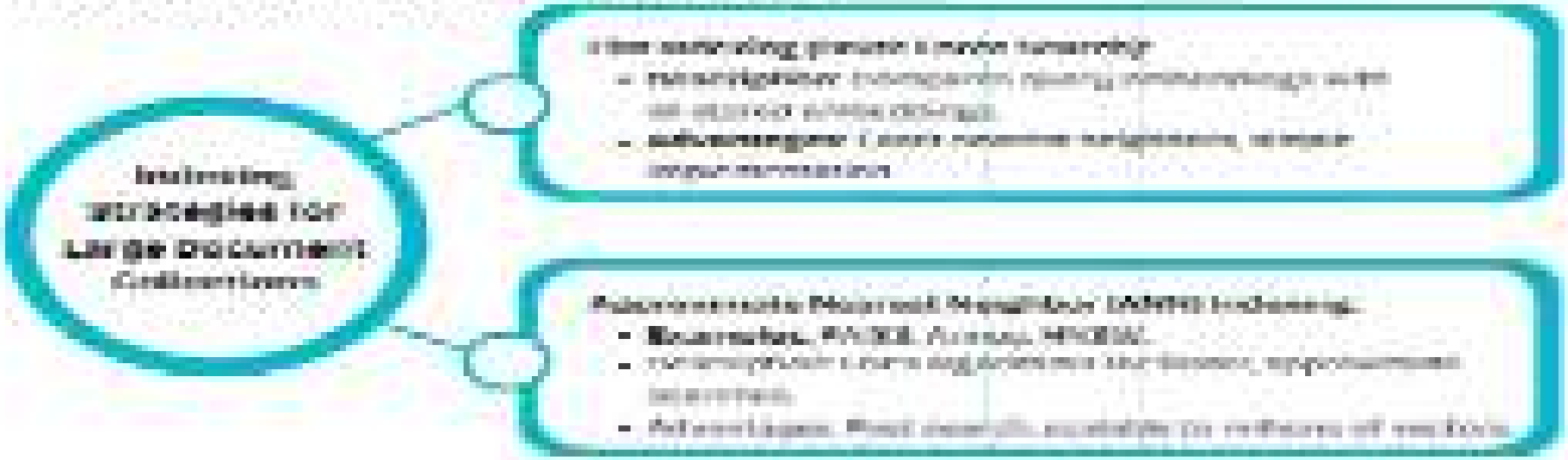
Embedding Generation And Vector Storage

choosing an embedding model



Embedding Generation And Vector Storage

Integrating with vector databases



Query Processing and Augmentation

Query Understanding and Reformulation

Query Processing and Augmentation

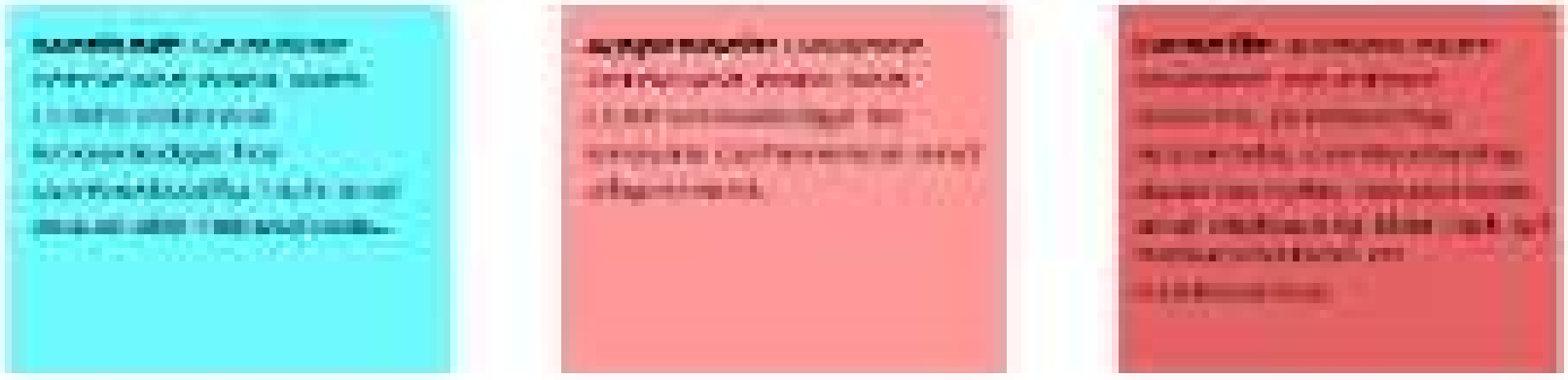
Techniques for Expanding Queries

LLM Integration For Generation

Prompt Engineering for Effective Use of Retrieved Context

LLM Integration For Generation

Balancing Retrieved Information with LLM Knowledge



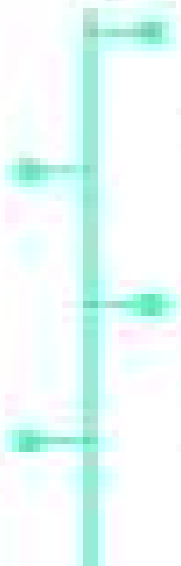
LangChain Framework

Introduction to LangChain

LangChain is a framework for integrating large language models with external data sources, facilitating natural language generation tasks. It offers modular components, including model interfaces, prompt management, and chains, enabling the development of context-aware applications.

LangChain Framework

Key Features



LangChain Framework

How LangChain Simplifies RAG Implementation

Data Ingestion and Workflow: LangChain streamlines data ingestion by integrating various data sources, such as databases, APIs, and document repositories, into a unified workflow.

Pre-Built Components: It provides ready-to-use components for tasks like document loading, text splitting, embedding generation, and vector storage, reducing the need for custom code.

Flexibility and Customization: Developers can easily swap components, such as different LLMs or vector databases, to optimize performance and cost for specific use cases.

Orchestration and Deployment: LangChain facilitates the orchestration of complex workflows, including retrieval, generation, and post-processing steps, and supports deployment in various environments.

LangChain Framework

LangChain Components

LangChain Framework

LangChain Expression Language (LCEL)

Syntax and Basic Usage

Syntax:

- LCEL [illegible]

Example:

```
[illegible]
```

[illegible]

Advantages:

- [illegible]
- [illegible]
- [illegible]

LangChain Framework

Benefits of Using LCEL in RAG Pipelines

Demo

Implementing a Simple RAG Pipeline

MODULE 4

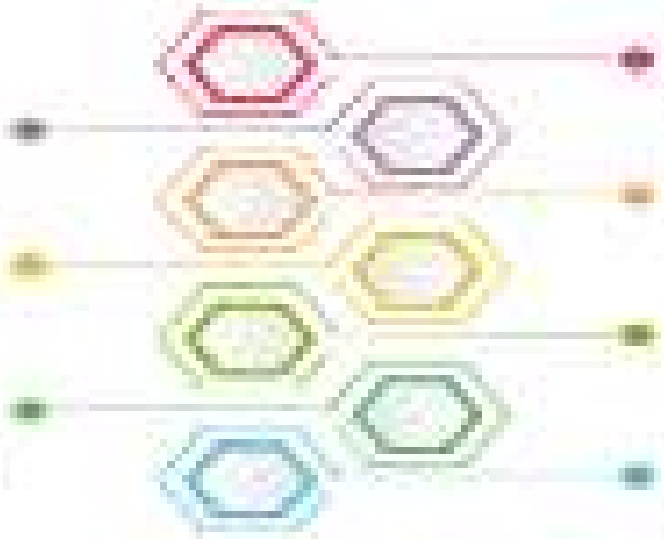
Advanced RAG Techniques

Advanced RAG Techniques

Introduction To Advanced RAG Techniques

Advanced RAG Techniques are sophisticated methodologies that enhance the performance and accuracy of Retrieval-Augmented Generation systems. These techniques refine the integration of information retrieval and generative models, enabling more accurate and contextually relevant responses. They involve leveraging complex mechanisms to improve the relevance and quality of retrieved information.

Key Components Of Advanced RAG Techniques

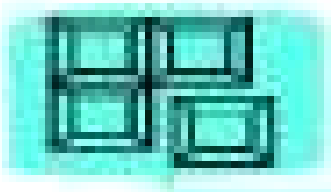


Text Splitting And Chunking Strategies

Importance of Effective Text Splitting

Text Splitting And Chunking Strategies

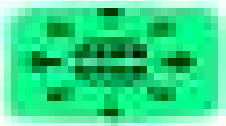
Balancing Chunk Size with Semantic Coherence



Different Splitting Techniques

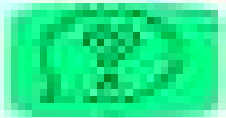
Query Transformation And Expansion

Techniques for Improving Retrieval Accuracy



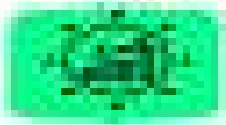
Synonym Expansion:

- Description: [illegible]
- Advantage: [illegible]



Concept Expansion:

- Description: [illegible]
- Advantage: [illegible]



Query Reformulation:

- Description: [illegible]
- Advantage: [illegible]

Hypothesis Generation: HyDE Technique

Explanation of Hypothetical Document Embeddings (HyDE)

Hypothesis Generation: HyDE Technique

Implementing HyDE with LangChain

LangChain Overview:

- A framework for building applications that leverage large language models.

Implementation Steps:

1. **Define Scenarios:**
2. **Generate Embeddings:** Use LangChain to create embeddings
3. **Analogies:**

Advantages: HyDE with LangChain streamlines

Demo

Comparing retrieval results with and without query expansion

Reranking and Filtering Retrieved Documents

Reranking:

Reranking refers to the process of reordering the retrieved documents based on their relevance to the query.

1. **Cross-Encoder Reranking:** This method uses a cross-encoder model to score the relevance of each document with respect to the query.
2. **Bi-Encoder Reranking:** This method uses a bi-encoder model to encode the query and documents into vectors and then ranks the documents based on their similarity to the query vector.
3. **Ensemble Reranking:** This method combines multiple reranking models to improve the overall performance.

Filtering:

Filtering is the process of removing irrelevant or low-quality documents from the retrieved set.

1. **Content-Based Filtering:** This method filters documents based on their content, such as keywords or topics.
2. **Metadata Filtering:** This method filters documents based on their metadata, such as publication date or author.
3. **User-Feedback Filtering:** This method uses feedback from users to filter out irrelevant documents.

Using Cross-Encoders For Reranking

Explanation of Cross-Encoders vs. Bi-Encoders

Popular Models: SentenceTransformers, MonoT5

1

SentenceTransformers:

1. **Description:** Uses an ensemble for generating high-quality sentence embeddings, leveraging pre-trained transformer models.
2. **Use Case:** Ideal for semantic search and clustering tasks.

2

MonoT5:

1. **Description:** Employs a sequence-to-sequence framework for text generation and summarization improvements.
2. **Use Case:** Excellent for tasks requiring nuanced text generation.

3

Benefits:

1. **Cross-Encoders:** Enhances interaction understanding for precise ranking.
2. **Bi-Encoders:** Efficient and scalable, best for large-scale systems.

LLM-based Document Filtering

Demo

- Implementing a reranking pipeline
- Comparing results before and after reranking

Advanced RAG Architectures

Overview

Advanced Retrieval-Augmented Generation (RAG) architectures enhance traditional RAG systems with sophisticated components and methods. They aim to boost accuracy, efficiency, and scalability, making them ideal for complex tasks and large-scale applications.

Multi-Step Retrieval

Coarse-to-Fine Retrieval Strategies

Description: A multi-step retrieval approach refines an initial broad search with subsequent, more focused searches to gradually improve the relevance of retrieved documents.

Types:

- **Coarse Retrieval:** Quickly identifies a broad set of documents using methods like keyword matching or ANN search.
- **Fine Retrieval:** Refines the coarse results using computationally intensive techniques like cross-encoder reranking or LLM assessment.

Advantages:

- **Scalability:** Efficiently manages large datasets by filtering out irrelevant documents early.
- **Accuracy:** Improves precision and relevance by applying sophisticated methods selectively.

Multi-Step Retrieval

Implementing Multi-Step Retrieval with LangChain

LangChain Overview: A framework for developing applications powered by language models, including multi-step retrieval.

Implementation Steps:

- Initial Retrieval: Set up a basic retriever using a vector store.
- Query Refinement: Develop a mechanism to refine queries based on initial results.
- Multi-Step Process: Implement a loop for multiple retrieval iterations.

Key Components:

1. Document Loaders: For loading data.
2. Embeddings: For converting text to vectors.
3. Vector Stores: For storing and searching vectors.
4. Retrievers: For fetching relevant documents.

Hybrid Search

[illegible]

Combining Lexical and Semantic Retrieval (e.g., BM25 + Embeddings)

- Lexical search (e.g., BM25) ranks documents based on exact keyword matches.
- Semantic search uses embeddings to find documents with similar meaning, even if they don't share exact words.

Reciprocal Rank Fusion (RRF)

- Combines results from lexical and semantic searches by assigning scores based on their rankings in each list.
- Documents that appear high in both lists get the highest final scores.

Weighted Sum of Scores

- Assigns weights to lexical and semantic scores, then adds them together.
- Allows you to prioritize one type of search over the other.

Re-ranking with Embeddings

- Use lexical search to get an initial set of results.
- Re-rank these results using embeddings to find the most semantically relevant ones.

Demo

Building an advanced RAG system with multi-step retrieval

MODULE 5

Scaling and Deploying RAG Systems

Metrics for Evaluating RAG Performance

Precision

- Definition: Precision measures the proportion of relevant documents retrieved out of all the retrieved documents.
- Formula: Precision = True Positives / (True Positives + False Positives)

Metrics for Evaluating RAG Performance

Recall

- Formula:

Metrics for Evaluating RAG Performance

F1-Score

- Definition: [illegible]
- Formula: $F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

Metrics for Evaluating RAG Performance

Mean Reciprocal Rank (MRR)

- Definition: MRR is a metric for evaluating systems that return ranked lists of results. It measures the average of the reciprocal ranks of the first relevant result.
- Formula: $\text{MRR} = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\text{rank}_i}$, where rank_i is the rank

Metrics for Evaluating RAG Performance

BLEU (Bilingual Evaluation Understudy)

- **Definition:** [illegible] measures how many n-grams (sequences of n words) in the generated text match those in the reference text.
- **Purpose:** BLEU is commonly used to evaluate [illegible] machine translation models by comparing the similarity between generated and reference translations.
- **Key Point:** Increasingly used in [illegible] applications, it helps measure how accurate and grammatically correct the generated text is. The higher the BLEU score, the more similar the generated text is to the reference text, indicating better performance. However, it can sometimes miss out on [illegible] and [illegible] nuances.

Metrics for Evaluating RAG Performance

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

1. **Definition:** (Level) Measures the overlap between the generated text and the reference text, used in tasks like [illegible] and machine [illegible]
   1. [illegible]
2. Measures the [illegible] between generated summaries and reference texts.
- **Example:** [illegible]
- [illegible]

Metrics for Evaluating RAG Performance

METEOR (Metric for Evaluation of Translation with Explicit ORdering)

- **Definition:** [illegible] to evaluate machine translation quality by considering synonyms, stemming, and paraphrasing, along with exact matches, to provide a more comprehensive assessment.
- **Formula:** [illegible] combines precision and recall, giving more weight to recall. It uses a penalty for fragmentation, considering the number of chunks that are not contiguous in both the reference and candidate translations.
- **Use Case:** [illegible] is useful when you want a metric that correlates well with human judgment and considers linguistic variations.

Metrics for Evaluating RAG Performance

BERTScore

- **BERTScore** (BERTScore) uses contextual embeddings from BERT to compare the semantic similarity between the generated and reference texts, rather than focusing on exact word matches.
- **How it Works:** It computes the cosine similarity between embeddings of the words in the generated and reference sentences to measure how similar they are in meaning.
- **Use Cases:** Useful for applications where understanding the meaning is important, such as in paraphrase identification and detailed text generation, as it captures more than just surface-level similarity.

RAG-specific metrics

Faithfulness

RAG-specific metrics

Relevance

Introduction to RAGAS Framework

Overview of RAGAS

The RAGAS (Retrieval-Augmented Generation Assessment System) framework is designed to assess the performance of Retrieval-Augmented Generation (RAG) models. RAG models combine the strengths of retrieval-based and generation-based AI, making them increasingly used in applications that require up-to-date information.

Introduction to RAGAS Framework

Purpose and key features

[illegible]

Key Features:

1. [illegible]
2. [illegible]

Introduction to RAGAS Framework

Integration with LangChain

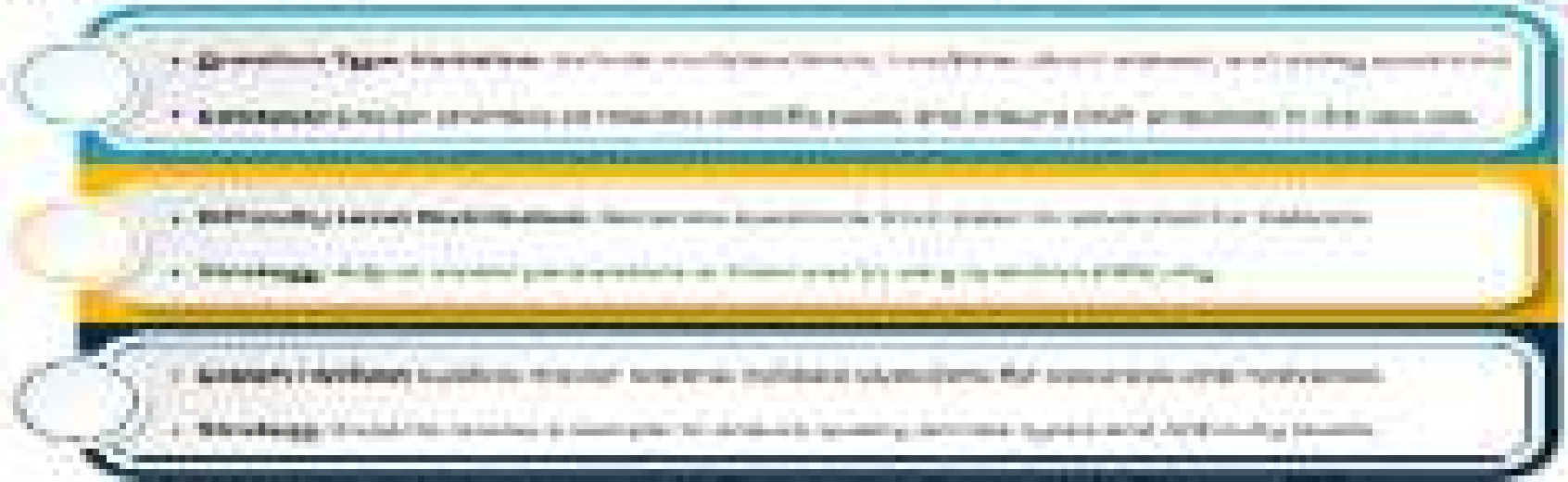
RAGAS evaluation metrics

Creating AI-augmented test sets

Techniques for generating diverse test questions

Creating AI-augmented test sets

Ensuring coverage of different question types and difficulties



Optimizing RAG Pipelines

Fine-tuning strategies for embeddings:

Optimizing RAG Pipelines

Fine-tuning strategies for embeddings:

Optimizing RAG Pipelines

Contrastive learning approaches

Optimizing RAG Pipelines

Contrastive learning approaches

Optimizing RAG Pipelines

LLM optimization for RAG

Optimizing RAG Pipelines

LLM optimization for RAG

[redacted]

[redacted]

Optimizing RAG Pipelines

Hyperparameter optimization

Optimizing RAG Pipelines

Using Bayesian Optimization for Parameter Tuning

Demo

Evaluating and Optimizing a RAG System

MODULE 6

Scaling and Deploying RAG Systems

Scaling And Deploying RAG Systems

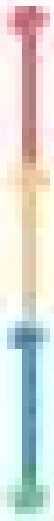
Efficient Indexing Strategies

Scaling And Deploying RAG Systems

Incremental Updating of Vector Stores

Scaling And Deploying RAG Systems

Load Balancing and Caching Techniques



Caching: [illegible]

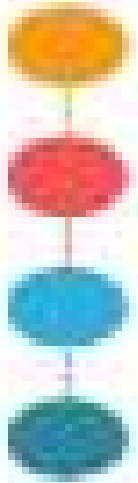
Load balancing: [illegible]

Caching: [illegible]

Load balancing: [illegible]

Scaling And Deploying RAG Systems

Asynchronous Processing for Improved Throughput



Message queues [illegible]

Caching [illegible]

Load balancing [illegible]

Rate limiting [illegible]

Scaling And Deploying RAG Systems

Scaling Retrieval with Distributed Vector Databases

Scaling And Deploying RAG Systems

Parallel Processing of Queries and Document Chunks

Query Parallelism: Distributes queries across multiple servers to reduce latency and support concurrent requests.

Document Chunking: Splits documents into chunks for parallel processing across multiple servers.

Load Balancing: Evenly distributes tasks to avoid bottlenecks and maximize resource use.

Scaling And Deploying RAC Systems

Overview of relevant AWS services (EC2, ECS, Lambda)

Amazon EC2 (Elastic Compute Cloud)

- **Purpose:** Offers scalable virtual servers for running applications and services.
- **Use Case:** Suitable for customized server configurations, data processing, and custom software.
- **Scaling:** Automatically adjusts resources based on demand, ensuring optimal performance.

Scaling And Deploying RAG Systems

Overview of relevant AWS services (EC2, ECS, Lambda)

Amazon ECS (Elastic Container Service)

1. **Purpose:** [illegible]
2. **Use Case:** [illegible]
3. **Scaling:** [illegible]

Scaling And Deploying RAG Systems

Overview of relevant AWS services (EC2, ECS, Lambda)

AWS Services:

1. **Amplify:** [illegible]
2. **User Lists:** [illegible]
3. **Lending:** [illegible]

Scaling And Deploying RAG Systems

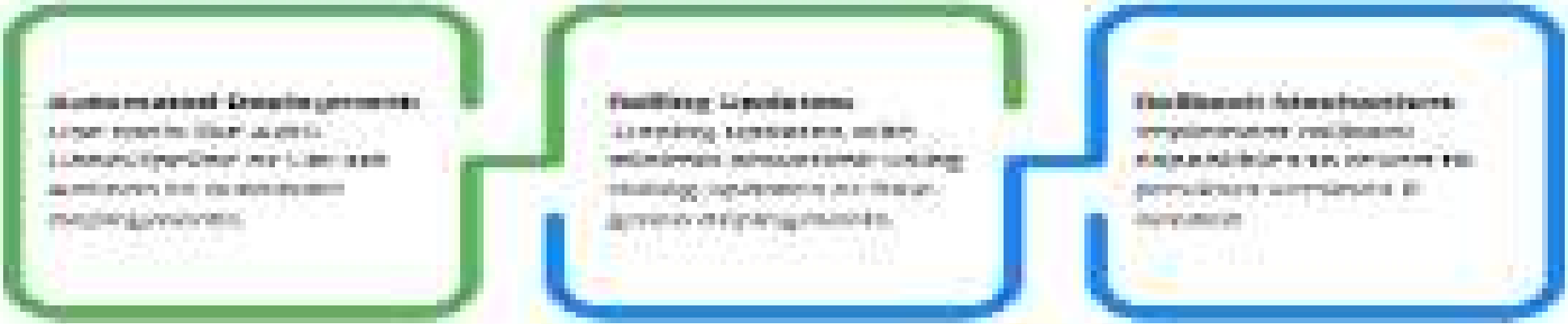
CI/CD pipelines for RAG systems

Scaling And Deploying RAC Systems

Continuous Integration (CI)

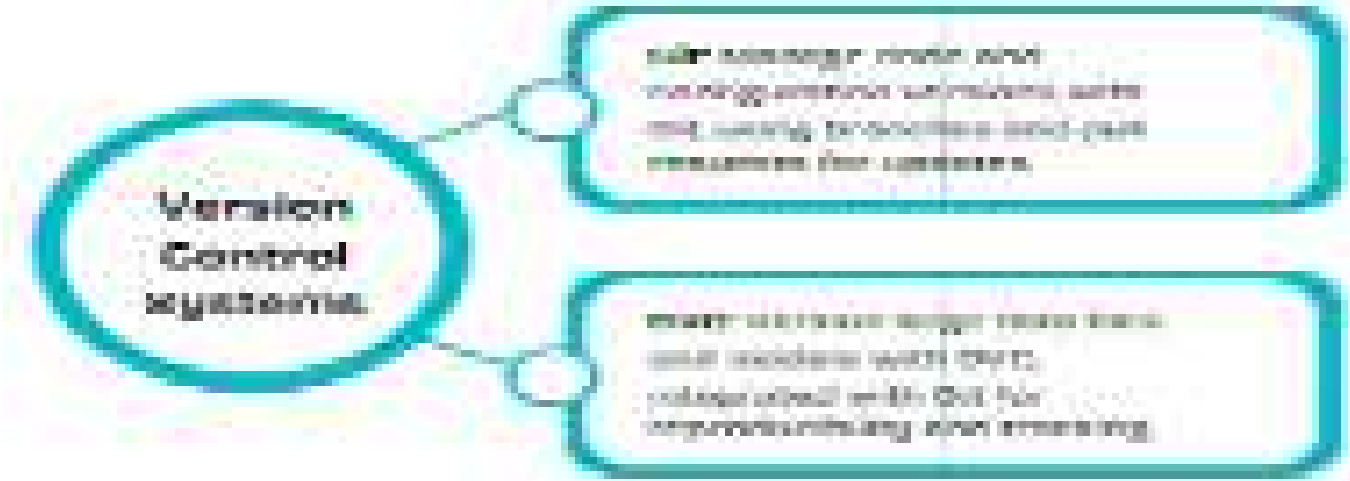
Scaling And Deploying RAC Systems

Continuous Deployment (CD)



Scaling And Deploying RAG Systems

Version Control for Documents and Embeddings



Scaling And Deploying RAG Systems

Version Control for Documents and Embeddings

Scaling And Deploying RAG Systems

Version Control for Documents and Embeddings

Scaling And Deploying RAG Systems

Monitoring and Maintaining Deployed Models

Latency: Time taken for model predictions; high latency affects real-time use.

Throughput: Number of predictions per second under concurrent load.

Error Rates: Failed predictions and exceptions, signaling issues.

Resource Utilization: CPU, GPU, and memory usage.

Scaling And Deploying RAG Systems

Setting Up Monitoring Dashboards

Scaling And Deploying RAG Systems

Strategies for Keeping the Knowledge Base Current

Scaling And Deploying RAG Systems

When to Update LLMs and Embedding Models

Scaling And Deploying RAG Systems

When to Update LLMs and Embedding Models

Scaling And Deploying RAG Systems

How to Update LLMs and Embedding Models

Scaling And Deploying RAG Systems

A/B Testing New Model Versions

Scaling And Deploying RAG Systems

A/B Testing New Model Versions

Demo

Deploying a RAG System

MODULE 7

Real-World RAG Applications and Future Trends

Real-World RAG Applications And Future Trends

Implementing RAG for Corporate Knowledge Bases

Real-World RAG Applications And Future Trends

Implementation Steps

Real-World RAG Applications And Future Trends

Outcome

Real-World RAG Applications And Future Trends

CI/Demo: Building an Advanced Enterprise Search System with RAG

Scaling And Deploying RAG Systems

Step 1: Data Collection and Preprocessing

- [illegible]

- [illegible]

Scaling And Deploying RAG Systems

Step 2: Setting Up the Retrieval System

- [illegible]

- [illegible]

Scaling And Deploying RAG Systems

Step 3: Fine-Tuning the LLM

Scaling And Deploying RAG Systems

Step 4: Integrating RAG

- [illegible]
- [illegible]

Scaling And Deploying RAG Systems

Steps To Building The User Interface

1. [illegible]
2. [illegible]

Scaling And Deploying RAG Systems

Step 8: Testing and Feedback

- [illegible]
- [illegible]

Scaling And Deploying RAC Systems

Step 7: Deployment

- [illegible]
- [illegible]

Scaling And Deploying RAG Systems

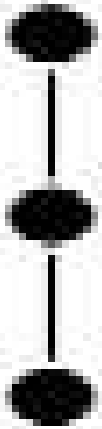
Enhancing Customer Support Chatbots with RAG Capabilities

Scaling And Deploying RAG Systems

Enhancing Customer Support Chatbots with RAG Capabilities

Real-World RAG Applications And Future Trends

Content Generation and Summarization Using RAG



Role of Generative

Application in Content Creation

Summarization Techniques

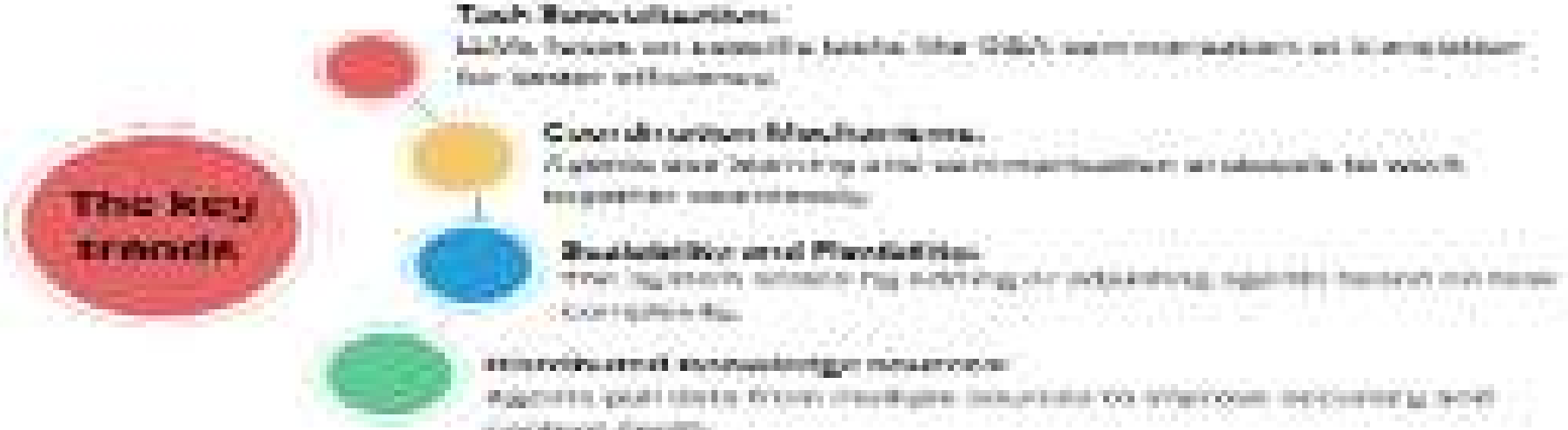
Real-World RAG Applications And Future Trends

Long-Form Content Summarization Techniques

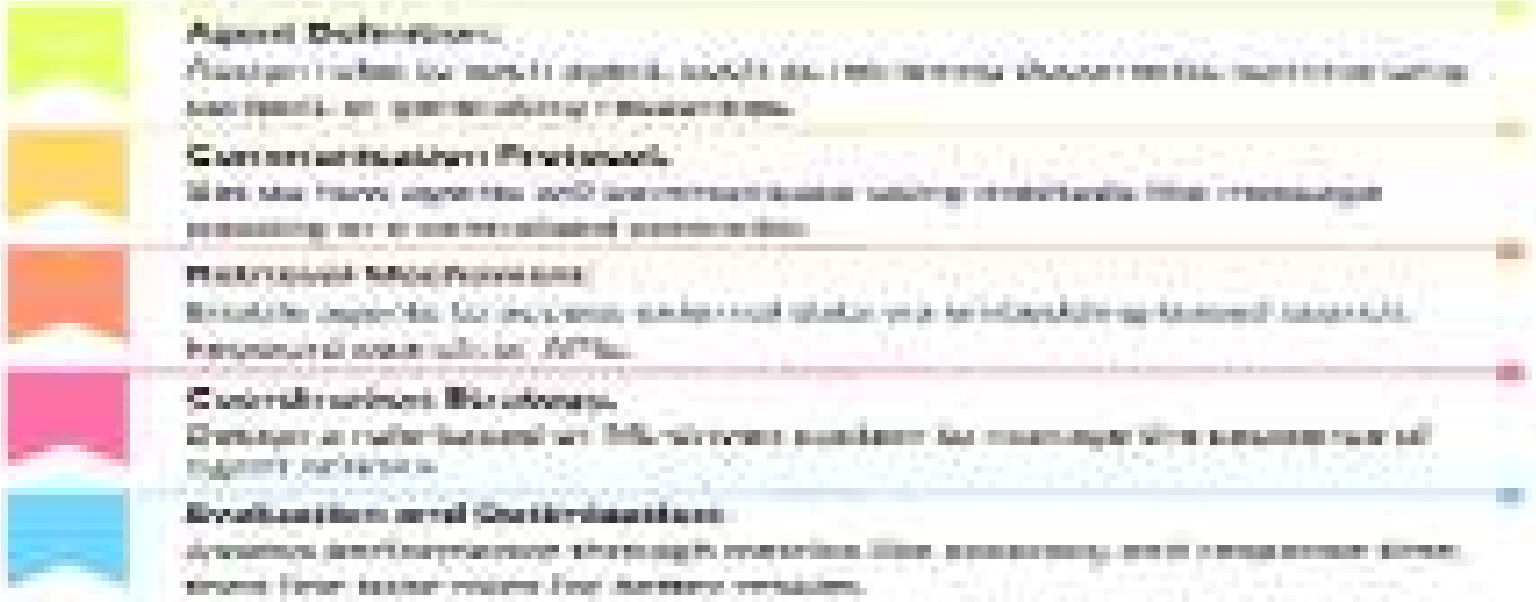
Real-World RAG Applications And Future Trends

Coordinating Multiple LLMs for Complex Tasks

Coordinating Multiple LLMs For Complex Tasks



Implementing a Simple Multi-Agent RAG System

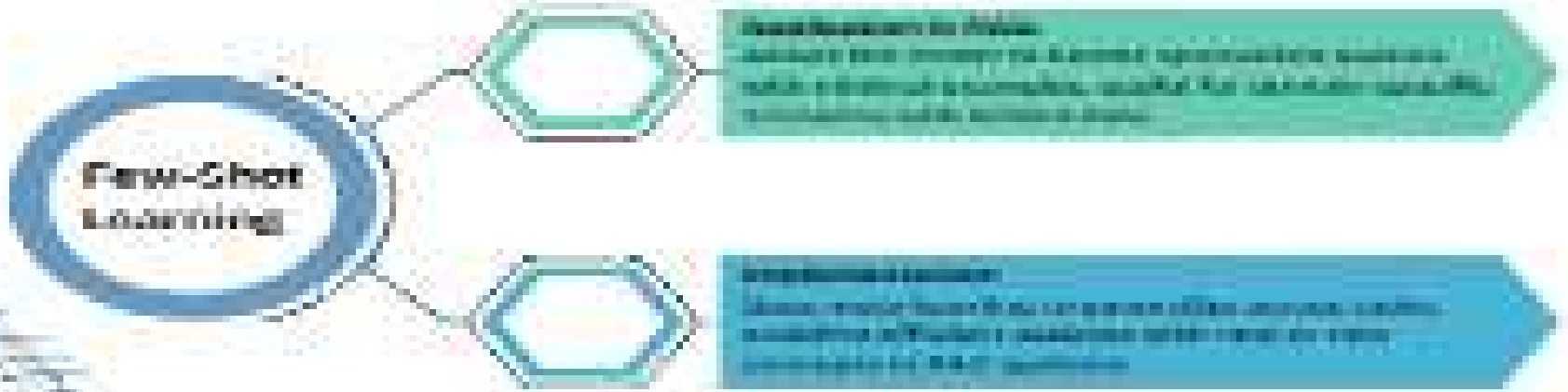


Continuous Learning And Adaptive Retrieval In RAG Systems

Techniques for Dynamically Updating Knowledge Bases

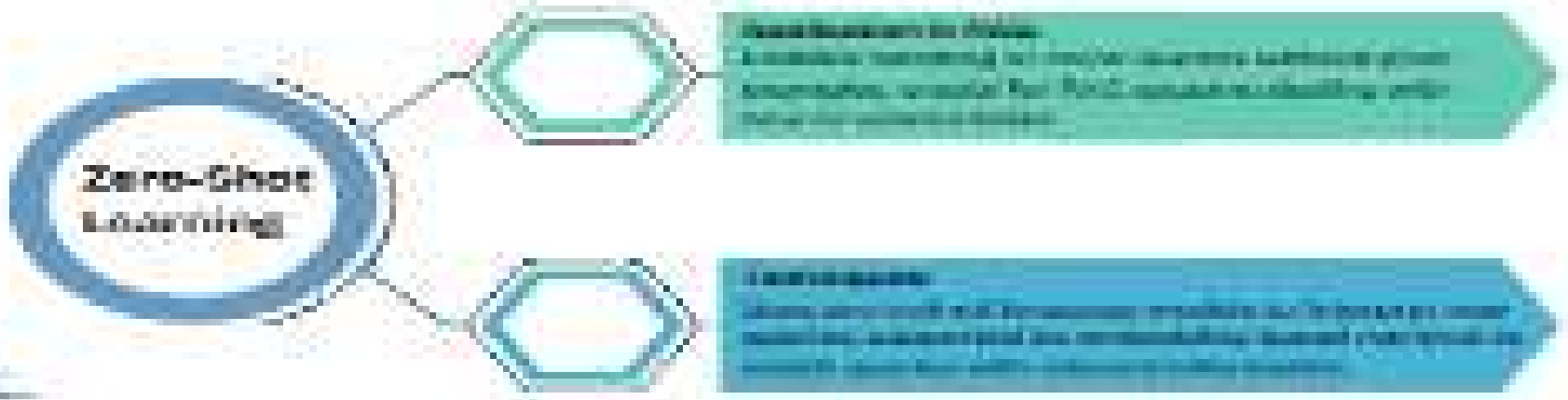
Continuous Learning And Adaptive Retrieval In RAG Systems

Exploration of Few-Shot and Zero-Shot Learning in RAG



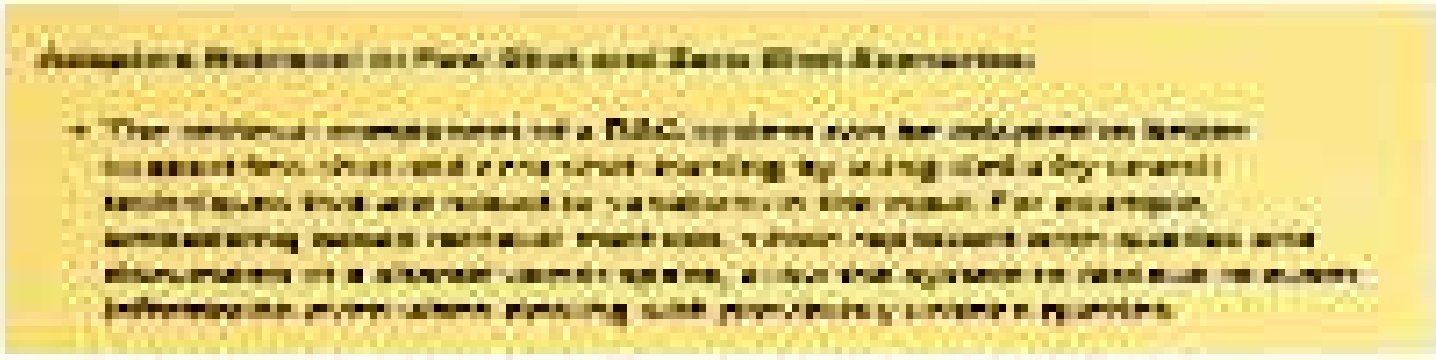
Continuous Learning And Adaptive Retrieval In RAG Systems

Exploration of Few-Shot and Zero-Shot Learning in RAG



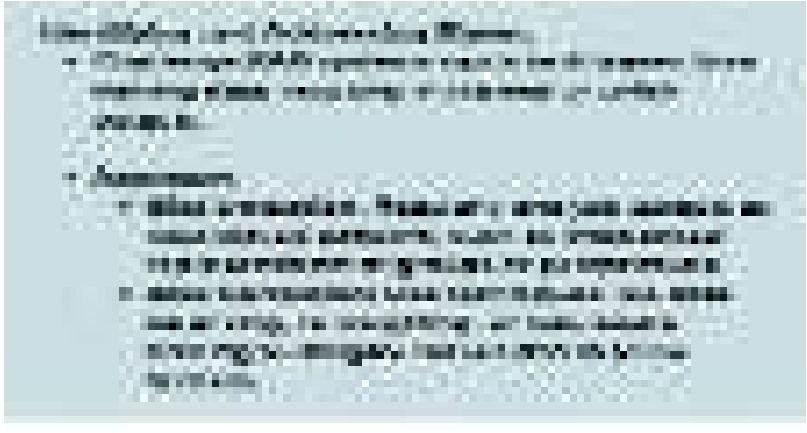
Continuous Learning And Adaptive Retrieval In RAG Systems

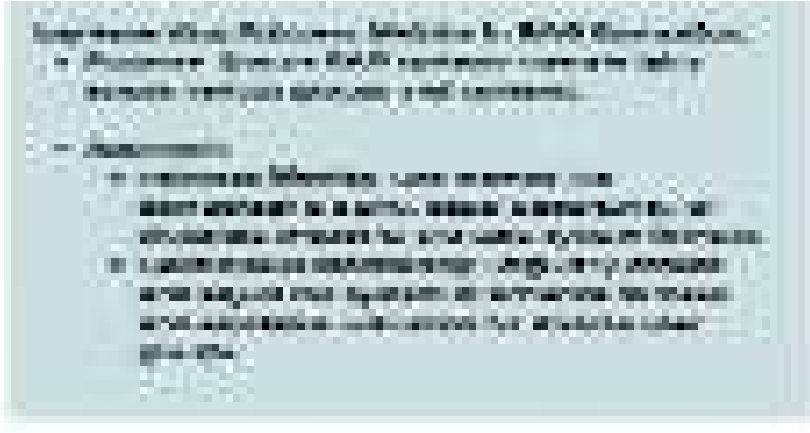
Exploration of Few-Shot and Zero-Shot Learning in RAG



Ethical Considerations And Responsible AI

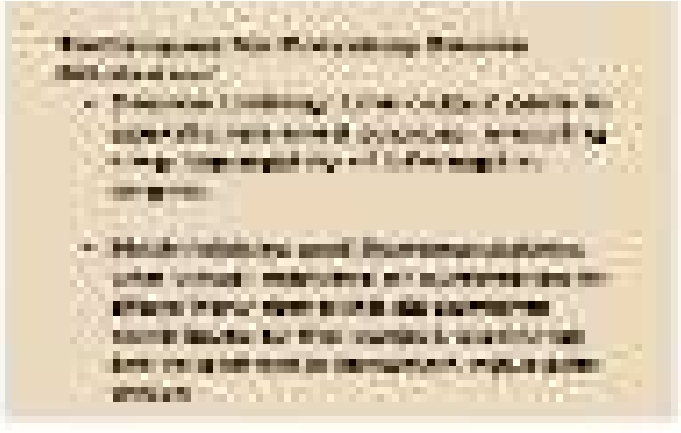
Bias Mitigation in RAG Systems

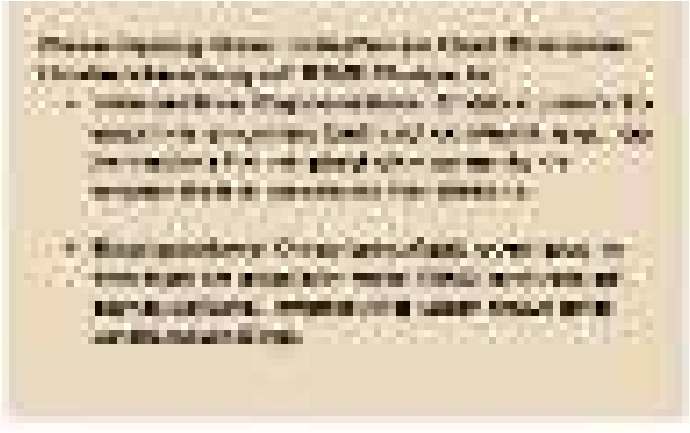




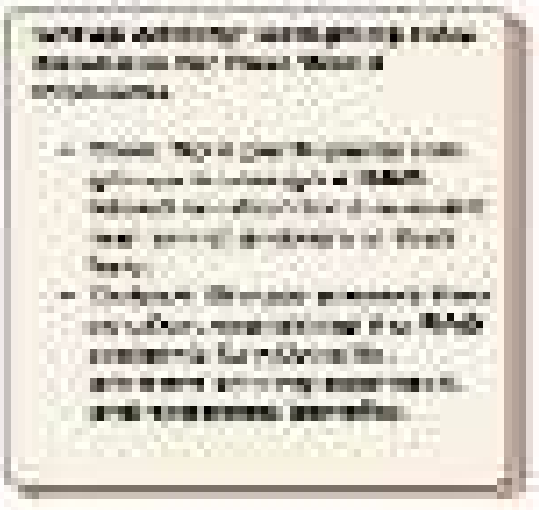
Ethical Considerations And Responsible AI

Ensuring Transparency and Explainability





Discussion And Brainstorming Session




THANK
YOU!