Analysis of Retrieval Methods (RAG Systems)

Prepared By -

Rahul Madhiwalla
Kunal Jai
Rajashekhar Keesari
Shangana Yadav
Trisha Ghoshal

Quick Overview

Retrieval is a critical component in any RAG systems that fetches relevant documents from a corpus to augment language model response generation. The effectiveness of the entire pipeline depends heavily on the retrieval quality.

This report discussing following retrieval techniques:

Key-word based search data retrieval

This is a kind of document retrieval technique often known as lexical search, it works by matching exact words, terms provided in the search query input.

It mainly prioritizes the literal presence of words and phrases over their contextual meaning or the user's intent behind the query.

e.g suppose a user searching on an online shop website - *“**show me top rated induction heater**”.*

The Lexical search method will fetch those documents or product listings that include the exact terms, **“induction heater**” and **“top rated**,” prioritizing results based on ranking scores.

Ranking works based on two main factors:

- **Term Frequency**: How often the keywords appear in a document. For instance, if a product listing or review contains the phrase “induction heater” multiple times, it may be considered more relevant because of the higher frequency.

- **Keyword Relevance/Weighting**: Some words carry more importance in a query. In this case, “top rated” might signal a preference for high-quality or popular items. If a document includes both “induction heater” and indicators like high ratings or reviews, it will score higher. The search engine may also factor in the keyword position (e.g., appearing in the title vs. body), boosting relevance.

Behind the scenes, the system builds an **inverted index**, a structure that maps each keyword to the list of documents where it appears. This indexing enables the search engine to quickly retrieve relevant documents without scanning the entire content corpus for every query.

Example:

A sample Product Listings for Lexical Indexing

Let’s consider three product descriptions from an e-commerce catalog:

Doc ID	Product Description
D1	*"Induction heater with fast heating"*
D2	*"Budget induction heater ideal for small kitchens"*
D3	*"Top performance gas stove with advanced burner"*

Tokenization and Normalization

Each product description is processed into lowercase tokens (Using NLTK, spaCy):

- **D1** → `["induction", "heater", "with", "fast", "heating"]`
- **D2** → `["budget", "induction", "heater", "ideal", "for", "small", "kitchens"]`
- **D3** → `["top", "performance", "gas", "stove", "with", "advanced", "burner"]`

Inverted Index (Lexical Index)

Keyword	Document IDs
induction	D1, D2
heater	D1, D2
fast	D1
heating	D1
budget	D2
ideal	D2
small	D2
kitchens	D2
top	D3
performance	D3
gas	D3

stove	D3
advanced	D3
burner	D3
with	D1, D3

Sample Search Query:

User query: *“Top rated induction heater”*

Matching Flow (Lexical Matching):

- **“top”** → D3
- **“rated”** → ❌ Not found in any document
- **“induction”** → D1, D2
- **“heater”** → D1, D2

Result:

- **D1** and **D2** match **"induction"** and **"heater"**
- **D3** matches only **"top"**

Since **“rated”** is not present in any document, it doesn't contribute to the match.
Therefore, **D1** and **D2** are considered more relevant than D3.

Ranking:
Among D1 and D2, the final ranking will depend on:

- Keyword frequency (how many times “induction” or “heater” appears),
- Field importance (e.g., title vs. body),
- Matching score (e.g., TF-IDF, BM25).

.

Semantic search data retrieval

Semantic search prioritizes contextual understanding and user intent over literal word matching, making it particularly effective for handling synonyms, related concepts, and natural language queries.

e.g Lets consider the same user example where user searching on an online shop website - *"show me top rated induction heater"*.

The semantic search method will fetch documents that are contextually related to the query, even if they don't contain the exact terms. It might retrieve products described as "**highly recommended electric cooktop**" or **"best-selling electromagnetic cooking device**" because it understands these are semantically similar to "top rated induction heater".

How Semantic Search Works:

Semantic search relies on **vector embeddings** - mathematical representations of words, phrases, and documents in high-dimensional space. These embeddings capture semantic relationships where similar concepts are positioned closer together.

Key Components:

- **Vector Embeddings**: Documents and queries are converted into dense numerical vectors (numerical arrays of large size) using pre-trained language models (like BERT, Sentence-BERT, or OpenAI embeddings). These vectors capture semantic meaning.
- **Similarity Calculation**: The system calculates similarity between query and document vectors using metrics like cosine similarity or dot product.
- **Contextual Understanding**: The model understands that "induction heater" and "electromagnetic cooktop" refer to the same type of product, even without exact keyword matches.

Example: Semantic Vector Representation

Let's consider the same three product descriptions:

Doc ID	Product Description
D1	*"Induction heater with fast heating"*
D2	*"Budget induction heater ideal for small kitchens"*
D3	*"Top performance gas stove with advanced burner"*

Vector Embeddings Process: Each product description is converted into a high-dimensional vector (typically 768 or 1024 dimensions):

- **D1** → [0.2, -0.1, 0.8, 0.3, ...] (768-dimensional vector)
- **D2** → [0.25, -0.05, 0.75, 0.28, ...] (768-dimensional vector)
- **D3** → [-0.1, 0.4, 0.2, -0.3, ...] (768-dimensional vector)

Sample Search Query: User query: *"Top rated induction heater"*

Semantic Matching Flow:

1. **Query Vectorization**: The query is converted to a vector: [0.22, -0.08, 0.77, 0.31, ...]
2. **Similarity Calculation**: Cosine similarity is calculated between query vector and all document vectors
3. **Ranking**: Documents are ranked by similarity scores

Result:

- **D1**: Similarity score = 0.89 (high semantic similarity)
- **D2**: Similarity score = 0.87 (high semantic similarity)
- **D3**: Similarity score = 0.34 (lower semantic similarity)

Even though none of the documents contain "rated", the semantic search understands the intent and ranks induction heater products higher than gas stoves.

Hybrid Search data retrieval

Hybrid Search combines both **Lexical (keyword-based)** and **Semantic** retrieval techniques to leverage the strengths of both.

- Advantage of Hybrid method-
  Keyword Search is fast, interpretable, and precise for exact term matching.

- **Semantic Search** captures user intent and concept similarity.

- Hybrid balances both: improved **recall** (more relevant results) and **precision** (more accurate matches).

Summary of Retrieval Methods in RAG

Method	Pros	Cons	Best Use Cases
Keyword Search	Fast, interpretable, great for precise queries	Fails on synonyms, context, user intent	Search engines, logs, rule-based systems
Semantic Search	Captures intent, handles paraphrasing and synonyms	Requires embedding models, slower retrieval	Q&A bots, recommendation systems, support
Hybrid Search	Best of both worlds, tunable balance between exact match and intent		

Sample Queries Tested for Demonstration

1. Keyword-heavy query (Lexical dominant)

Query: "Leave policy for 2025 in Company XYZ"

- **Full-text hit:** Will find exact match on documents containing “leave”, “policy”, “2025”.

- **Semantic value:** May enhance ranking if document paraphrases it as “employee time-off guidelines for the current year”.

2. Semantically rich, vague query

Query: "How much time off do I get in a year?"

- **Lexical match:** Might fail because these exact words aren't used.

- **Semantic match:** Would understand it's about *leave entitlements* or *vacation days*.
- **Hybrid strength:** Scores boost due to semantic similarity even if keywords don't match exactly.

3. Short, ambiguous query

Query: "joining steps"

- **Lexical:** May not match if docs say "onboarding procedure" instead.
- **Semantic:** Recognizes that "joining" and "onboarding" are related.
- **Hybrid:** Balances weak keyword match with strong embedding similarity.

4. Paraphrased intent

Query: "Benefits offered to employees"

- **Lexical:** Might miss it if doc says "employee perks" or "company advantages".
- **Semantic:** Embeddings capture the similarity.
- **Hybrid:** Keeps relevant results even with vocabulary mismatch.

5. Multiple intent query

Query: "Who approves my leaves and how to apply?"

- **Lexical:** Could match "apply" or "approval" but not understand intent structure.
- **Semantic:** Identifies it relates to the "Leave application process".
- **Hybrid:** Combines both to surface the relevant section from HR policy.

6. Unseen query with domain context

Query: "How to connect to HRMS from home?"

- **Lexical:** Might miss unless doc uses "HRMS" and "VPN" or "remote access".
- **Semantic:** Helps bridge gap if user talks in different terms like "portal" or "system login".
- **Hybrid:** Keeps the connection with text like "remote HR portal access".