Hybrid Search: Combining BM25 and Semantic Search for Better Results with Langchain

Akash A Desai · Follow
Published in LanceDB · 4 min read · Dec 9, 2023

169 2

Have you ever wondered how search engines find exactly what you're looking for? Most often a combination of keyword matching and semantic search is used to search for user quries. This is known as hybrid search. Let's see how we can implement a simple hybrid search pipeline for document search.

Understanding BM25:

BM25 is a ranking algorithm used in information retrieval systems to estimate the relevance of documents to a given search query.

- **What it does:** It looks at how often your search words appear in a document and considers the document's length to provide the most relevant results.

- **Why it's useful:** It's perfect for sorting through huge collections of documents, like a digital library, without bias towards longer documents or overused words.

Key elements of BM25:

- **Term Frequency (TF):** This counts how many times your search terms appear in a document.
- **Inverse Document Frequency (IDF):** This gives more importance to rare terms, making sure common words don't dominate.
- **Document Length Normalization:** This ensures longer documents don't unfairly dominate the results.
- **Query Term Saturation:** This stops excessively repeated terms from skewing the results.

Overall

```
score(d, q) = ∑(tf(i, d) * idf(i) * (k1 + 1)) / (tf(i, d) + k1 * (1 - b + b * (d
```

When is BM25/ Keyword search Ideal?

- **Large Document Collections:** Perfect for big databases where you need to sort through lots of information.
- **Preventing Bias:** Great for balancing term frequency and document length.

- **General Information Retrieval**: Useful in various search scenarios, offering a mix of simplicity and effectiveness.

Practical Application: Building a Hybrid Search System

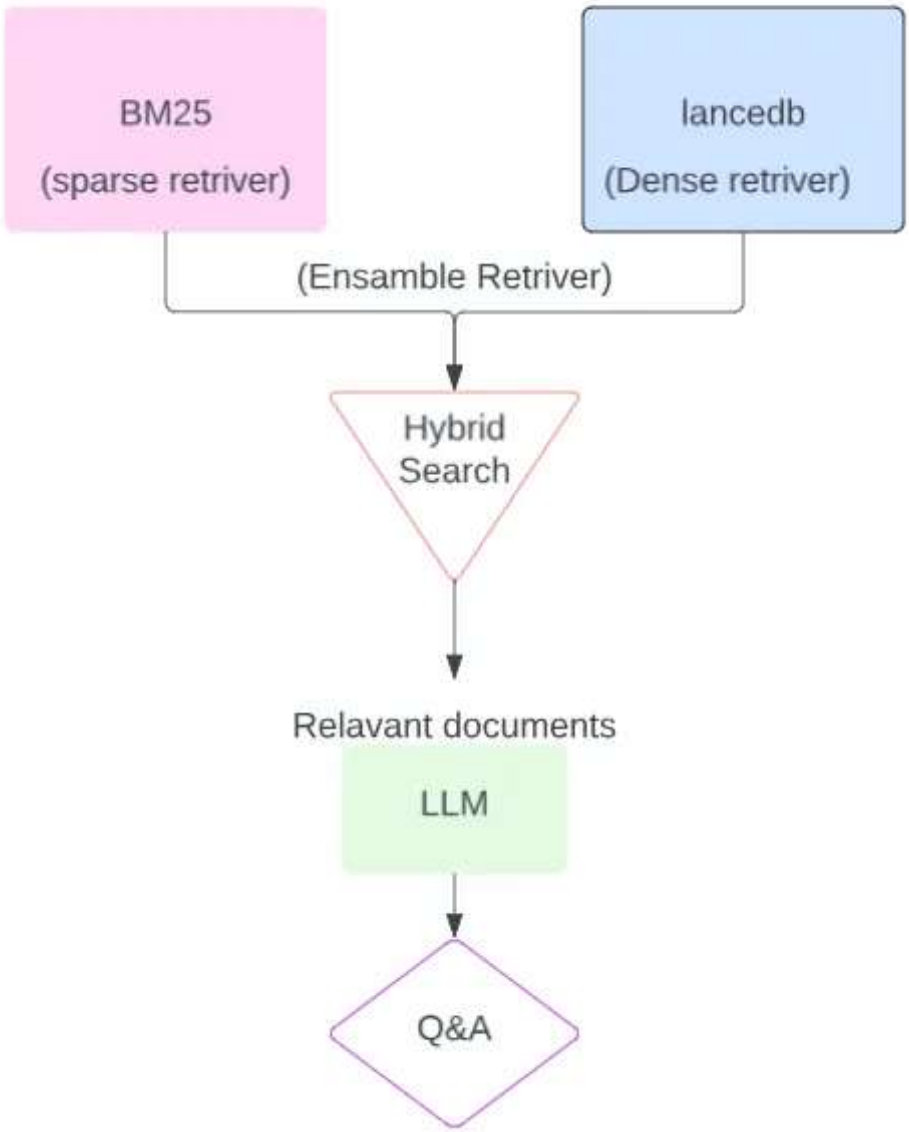
Imagine you're crafting a search system for a large digital library. You want it not only to find documents with specific keywords but also to grasp the context and semantics behind each query. Here's how:

- **Step 1**: BM25 quickly fetches documents with the search keywords.
- **Step 2**: VectorDB digs deeper to find contextually related documents.
- **Step 3**: The Ensemble Retriever runs both systems, combines their findings, and reranks the results to present a nuanced and comprehensive set of documents to the user.

What Exactly is Hybrid Search?

Hybrid search can be imagined as a magnifying glass that doesn't just look at the surface but delves deeper. It's a two-pronged approach:

- **Keyword Search:** This is the age-old method we're most familiar with. Input a word or a phrase, and this search hones in on those exact terms or closely related ones in the database or document collection.
- **Vector Search:** Unlike its counterpart, vector search isn't content with mere words. It works using semantic meaning, aiming to discern the query's underlying context or meaning. This ensures that even if your words don't match a document exactly if the meaning is relevant, it'll be fetched.



Follow along with this colab

Google Colaboratory

Edit description

colab.research.google.com

Let's get to the code snippets. Here we'll use langchain with LanceDB vector store

```
# example of using bm25 & lancedb -hybrid serch

from langchain.vectorstores import LanceDB
import lancedb
from langchain.retrievers import BM25Retriever, EnsembleRetriever
from langchain.schema import Document
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.document_loaders import PyPDFLoader

# Initialize embeddings
embedding = OpenAIEmbeddings()
```

and load a single PDF.

```
# load single pdf

loader = PyPDFLoader("/content/Food_and_Nutrition.pdf")
pages = loader.load_and_split()
```

Create BM25 sparse **keyword matching retriever**

```
# Initialize the BM25 retriever
bm25_retriever = BM25Retriever.from_documents(pages)
bm25_retriever.k =  2  # Retrieve top 2 results
```

Create lancedb vector store for dense **semantic search/retrieval.**

```
db = lancedb.connect('/tmp/lancedb')
table = db.create_table("pandas_docs", data=[
    {"vector": embedding.embed_query("Hello World"), "text": "Hello World", "id"
], mode="overwrite")


# Initialize LanceDB retriever
docsearch = LanceDB.from_documents(pages, embedding, connection=table)
retriever_lancedb = docsearch.as_retriever(search_kwargs={"k": 2})
```

Now ensemble both retrievers, here you can assign the weightage to it.

```
# Initialize the ensemble retriever
ensemble_retriever = EnsembleRetriever(retrievers=[bm25_retriever, retriever_lan
                                       weights=[0.4, 0.6])

# Example customer query
query = "which food needed for building strong bones and teeth ?
 which Vitamin & minerals importat for this?"
```

Open in app ↗ Sign up Sign in

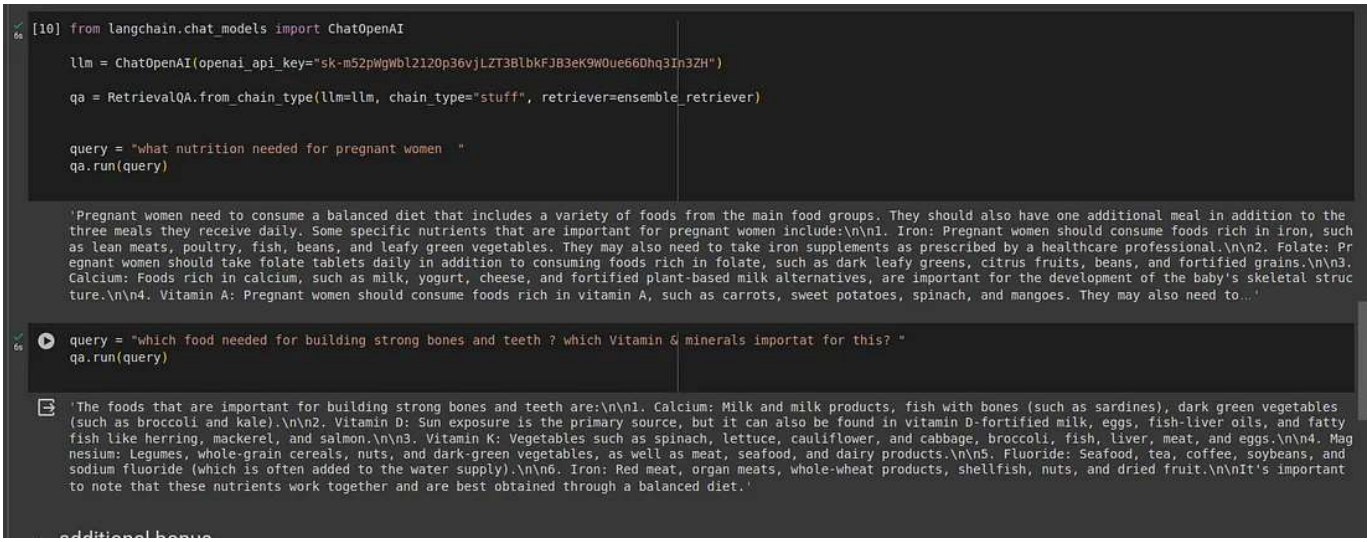
Medium Search Write

Using ensemble retriever it's trying to search each word in documents, such as strong bones & teeth as well as its also searching it in using lancedb which will find most similar documents based on similarity.

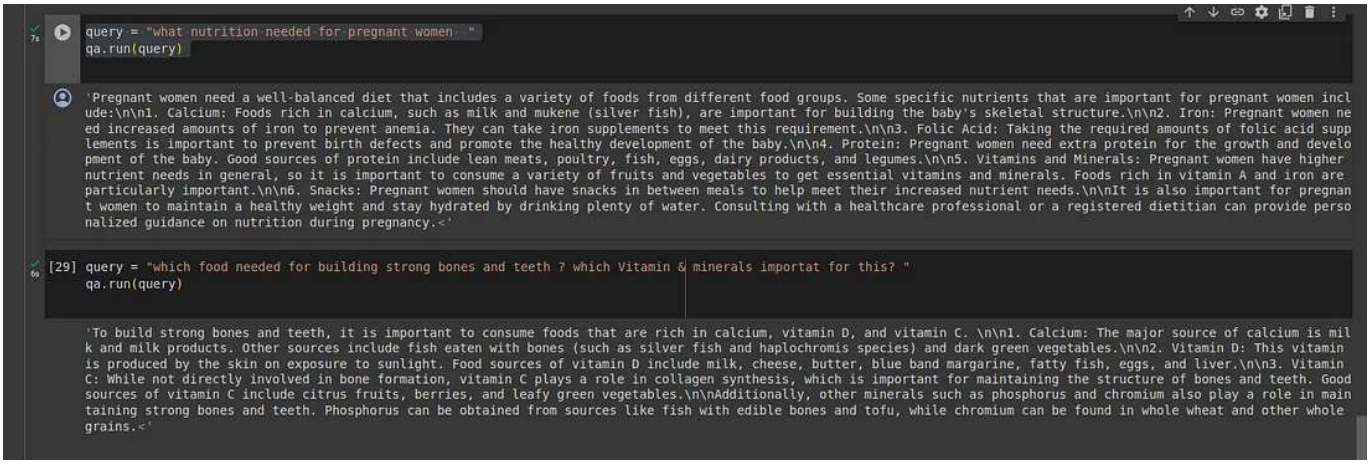
```
from langchain.chat_models import ChatOpenAI

llm = ChatOpenAI(openai_api_key="sk-yourapikey")
```

```
#if you want to use opensource models such as lama,mistral check this
# https://github.com/lancedb/vectordb-recipes/blob/main/tutorials/chatbot_using_

qa = RetrievalQA.from_chain_type(llm=llm, chain_type="stuff", retriever=ensemble

query = "what nutrition needed for pregnant women  "
qa.run(query)
```

again here its searching the keyword — “ nutrition pregnant women” in the database using bm25 & returning the best matching results & similarly at the same time we are using lancedb for this. this is how it's more powerful to extract text.



below are answers from the traditional rag, you can check this in our repo the answers may vary based on different parameters, models, etc.

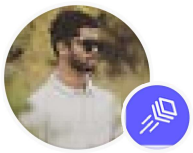


you can try this on colab with your pdf & use case.This is how you can use hybrid search to improve your search quality.

Explore More with Our Resources

Discover the full potential of hybrid search and beyond in our LanceDB repository, offering a setup-free, persisted vectorDB that scales on on-disk storage. For a deep dive into applied GenAI and vectorDB applications, examples, and tutorials, don’t miss our VectorDB-Recipes at https://github.com/lancedb/vectordb-recipes. From advanced RAG methods like Flare, Rerank, and HyDE to practical use cases, our resources are designed to inspire your next project .

LIm | Vector Database | Semantic Search



Written by Akash A Desai

Follow

77 Followers · Writer for LanceDB

Data Scientist with 4+ years of exp ! open source !Vision ! Generative AI ! Vectordb ! llms

More from Akash A Desai and LanceDB

Akash A Desai in LanceDB

Better RAG with Active Retrieval Augmented Generation FLARE

Welcome to our deep dive into Forward-Looking Active Retrieval Augmented...

7 min read · Nov 18, 2023

195

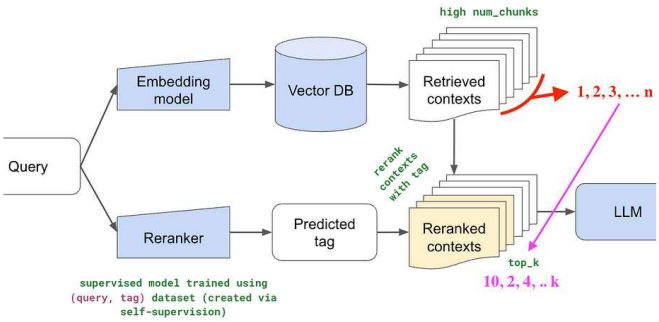
Chang She in LanceDB

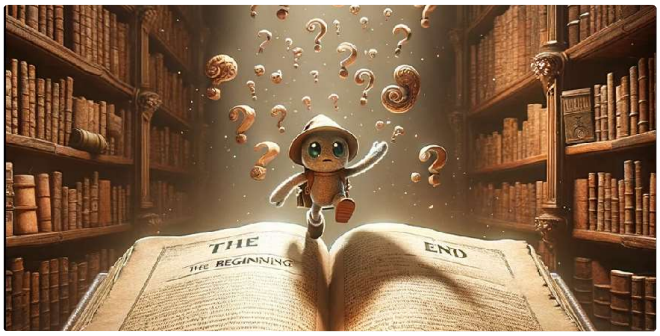
LanceDB + Polars

A (near) perfect match

6 min read · Jan 19, 2024

202 2





Mahesh Deshwal in LanceDB

Simplest Method to improve RAG pipeline: Re-Ranking

Problem Statement:

4 min read · Nov 23, 2023

176

Akash A Desai in LanceDB

Better RAG with LOTR-Lord of Retriever

Implementing LOTR merger retriever & solving the ‘Lost in the Middle’ Challenge in…

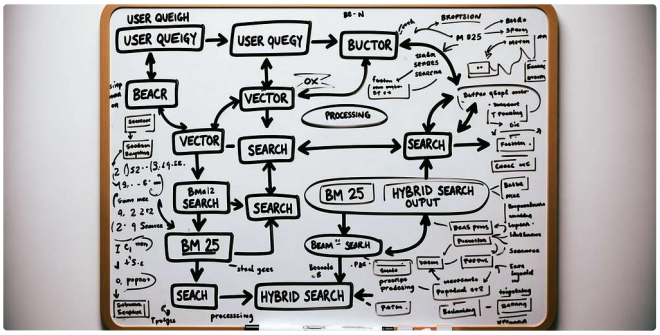
7 min read · Dec 21, 2023

121 1

See all from Akash A Desai

See all from LanceDB

Recommended from Medium



azhar in azhar labs

Vector Search and BM 25 (hybrid search): Advanced Ensemble...

Introduction

✨ · 5 min read · Oct 14, 2023

77

Fanghua (Joshua) Yu in Neo4j Developer Blog

Building A Graph+LLM Powered RAG Application from PDF...

A Step-by-step Walkthrough with GenAI-Stack and OpenAI

6 min read · Jan 15, 2024

544 3

Lists

Natural Language Processing

1215 stories · 692 saves

ChatGPT prompts

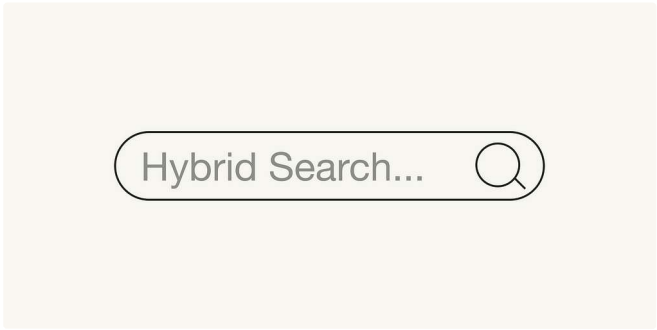
42 stories · 1146 saves

data science and AI

40 stories · 80 saves

Generative AI Recommended Reading

52 stories · 743 saves

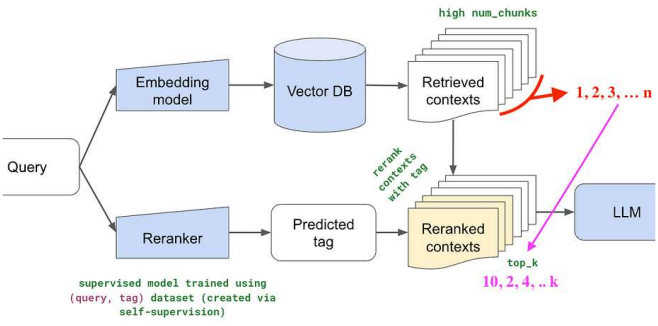


Leonie Monigatti in Towards Data Science

Improving Retrieval Performance in RAG Pipelines with Hybrid Search

How to find more relevant search results by combining traditional keyword-based searc...

8 min read · Nov 28, 2023



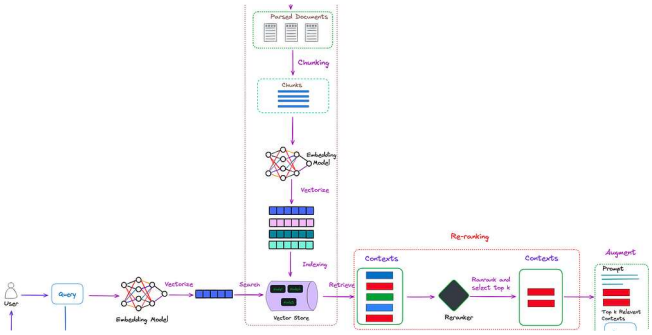
Mahesh Deshwal in LanceDB

Simplest Method to improve RAG pipeline: Re-Ranking

Problem Statement:

4 min read · Nov 23, 2023

558 8



Florian June in Towards AI

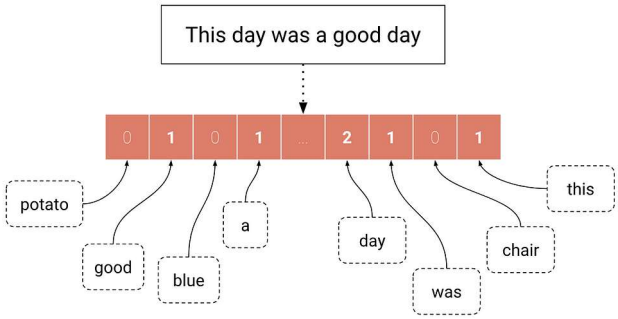
Advanced RAG 04: Re-ranking

From Principles to Two Mainstream Implementation Methods

✦ · 11 min read · Feb 14, 2024

346

176



Anar Aliyev in LLMRails

Write your own Hybrid Search with RRF and 'Ember' rerank

Embedding models are good for vector search with the help of vector databases. Bu...

3 min read · Oct 23, 2023

48

See more recommendations