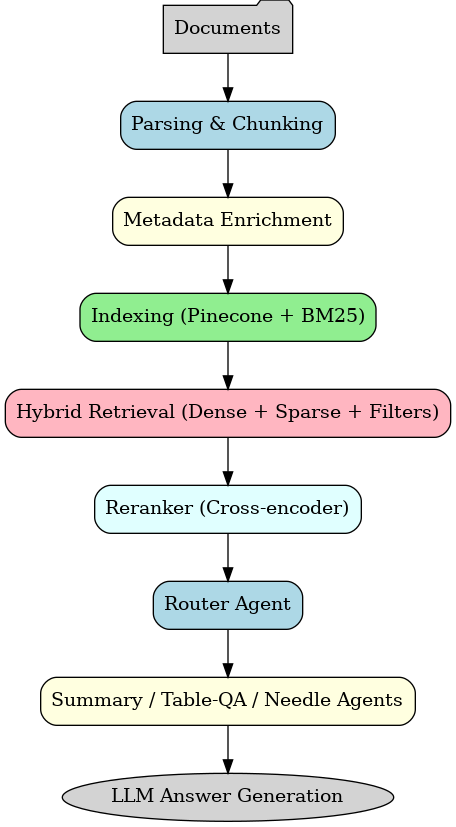
# פרויקט גמר קורס Driven Hybrid RAG —

## רקע והסבר מושגים

* **RAG (Retrieval‑Augmented Generation)** — שילוב של אחזור מידע ממסמכים עם מודל שפה גדול (LLM). המודל מקבל גם הקשר שנשלף מהמסמכים וגם את השאלה, ומחזיר תשובה מדויקת יותר.
* **Hybrid Retrieval (Dense + Sparse)**  — שילוב בין אחזור סמנטי (ווקטורים צפופים) לבין אחזור מבוסס מילות מפתח (sparse, לדוגמה BM25 נחליף אותו ל TF-IDF). זה נותן איזון בין משמעות מילולית (keywords) לבין קרבה סמנטית (embedding) .
* **Metadata‑Driven Retrieval**  — שימוש בשדות מטא־דאטה (כמו PageNumber, ClientId, SectionType) לסינון ולשיפור אחזור רלוונטי. מאפשר לענות על שאלות כמו: “תן לי רק טבלאות משנת 2023"
* **Reranker**  — מודל כמו ( cross‑encoder) שמדרג מחדש את תוצאות האחזור כדי להבטיח שהקונטקסט הכי רלוונטי יישלח (בחירה של6–8 הקטעים הרלוונטיים ביותר) ל - LLM
* **Agents & Routing** — בניית מספר סוכנים ייעודיים כמו ( Summary Agent,, Table‑QA Needle) כאשר Router Agent מנתב את השאלה לתת-הסוכן המתאים, לפי הכוונה וסוג שאלה.
* **Pinecone**  —(בונוס) שירות DB לווקטורים המאפשר שמירה, אחזור וניהול embedding-ים בסקייל גבוה, עם תמיכה מובנית בסינון מטא־דאטה.

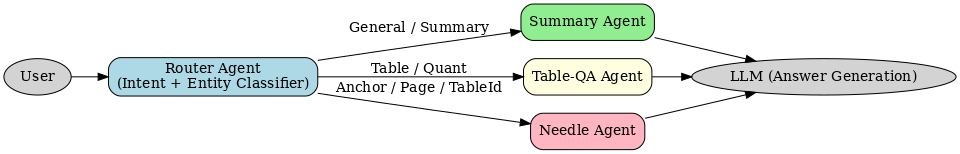
**ארכיטקטורה – תרשים**



## זרימת תהליך (Flowchart)

C:\Users\Natalia-2024\Downloads\Hybrid-RAG-Project\docs\flow_diagram.png.png

## Agent Router Flow (Flowchart)

  
בתרשים זה ניתן לראות כיצד השאלה עוברת דרך **Router Agent** אל הסוכן המתאים:

* **Router Agent** מקבל את השאלה ומבצע Classifier (Intent + Entity recognition).
* אם מדובר בשאלה כללית **Summary Agent**
* אם נדרשת תשובה מתוך טבלה **Table-QA Agent**
* אם מדובר במידע מאוד ספציפי **Needle Agent**

📷

## שלבי פיתוח

### Parsing & Chunking

* חלוקה לפי מבנה טבעי: כותרות, סעיפים, טבלאות, תרשימים.
* Distillation: לצמצם כל צ'אנק לשמירת ליבת המידע. יעד: סכום הטקסט של כל הצ'אנקים ≤ ~5% מגודל המסמך המקורי.
* שמירת **Anchors** בכל צ'אנק: PageNumber, TableId/FigureId, SectionType, ומיקום (שורה/עמודה בטבלה אם קיים).

### Metadata מינימלית (לבחירת 5 שדות לפחות)

שדות מומלצים לצ'אנק:

* FileName, PageNumber, SectionType (Summary/Timeline/Table/Figure/Analysis/Conclusion)
* ChunkSummary (תקציר קצר), Keywords, CriticalEntities
* IncidentType/IncidentDate (אם רלוונטי), AmountRange (אם רלוונטי)
* TableId/FigureId (אם קיים), ClientId/CaseId

### Indexing (טקסט + טבלאות)

* טבלאות המרה ל‑**CSV/Markdown**  + תיאור טקסטואלי קצר (Caption) לשיפור אחזור סמנטי.
* יצירת Embeddings של טקסט ושמירתם ב‑**Pinecone**  יחד עם המטא‑דאטה כולל תמיהכ ב-namespaces למשל לכל ClientId (לצורך סינון בזמן Retrieval)
* אחזור Sparse (BM25) במקביל.

### Hybrid Retrieval

1. בניית **Metadata Filters -** חילוץ מילות מפתח (ישויות, תאריכים, (keywords
2. שילוב של Dense Retrieval (K≈10) ∪ Sparse Retrieval (K≈10)
3. שימוש בפילטרים לפי Metadata כמו (ClientId, Year, SectionType)
4. Reranker LLM - לצמצום ל 6–8 תוצאות סופיות.

### ) Multi‑Document Supportתיקי עבודה)

* תמיכה במסמכים מרובים לכל לקוח /מקרה , סינון לפי ClientId / CaseId
* אפשרות לבצע אחזור רק מתיק אחד או מכלל המסמכים.
* ניהול ב‑Pinecone בעזרת namespaces

### Agents

* **Router Agent** — מקבל שאילתה ומנתב.
* **Summary Agent**  — מסכם חלקים נבחרים.
* **Needle Agent**  — מאתר פסקה / Anchor ספציפי.
* **Table‑QA Agent**  — מענה לשאלות כמותיות/טבלאיות (כולל הפניות ל‑TableId/PageNumber)

תיק עבודה — קוד מייצג

 src/chunking.py — חלוקה לפי מבנה + Distillation ל~5% ושמירת Anchors.

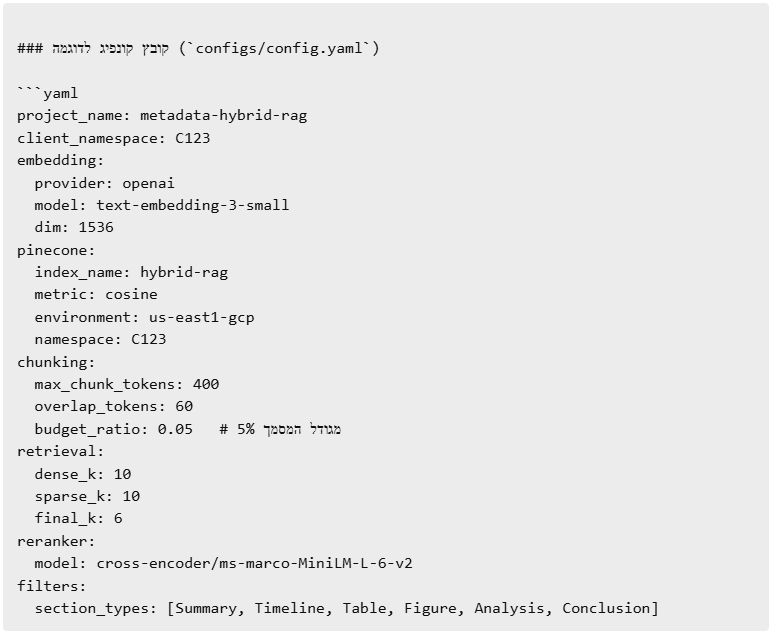
 src/indexing.py — אינדוקס Pinecone (Dense) + TF-IDF(sparce) עם מטא-דאטה צמודה לפילטרים.

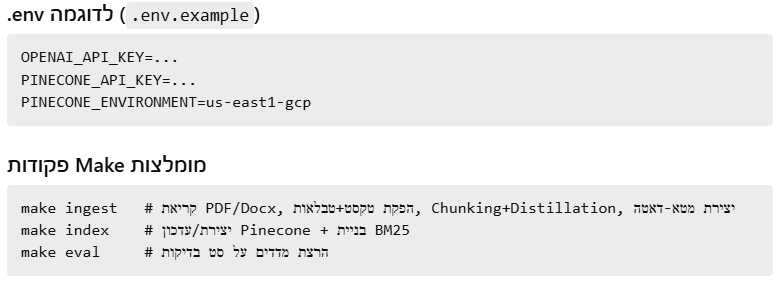
 src/retrieval.py — Hybrid Retrieval (K≈10+10) + ריראנקר לצמצום ל־6–8 הקשרים.

 src/agents/ — Router, Summary, Needle, Table‑QA.

 src/eval/ —למטריקות שביקשו RAGAS הרחבה של



קובץ קונפיג לדוגמא:



## סכמות נתונים (חשוב להגשה)

**קבצי דוגמא להערכה וחקר:**

* שאלות הערכה- data/evaluation\_questions.json
* Ground Truth- data/ground\_truth.json
* סט בדיקות RAGAS- data/ragas\_testset.json

סכמת צ'אנק (JSON)



### סכמת טבלה (CSV + תיאור)

* קובץ CSV data/processed/tables/{file}\_{table\_id}.csv
* קובץ Markdown data/processed/tables/{file}\_{table\_id}.md ( (תקציר/תיאור קצר)
* מטא‑דאטה להצמדה לווקטורים: TableId, PageNumber, SectionType="Table", FileName, ClientId/CaseId.

## משימות המשך (כלולות בתיק + הנחיות לביצוע)

1. **בניית סט דאטה לדוגמה (מסמכים אמיתיים, אנונימיזציה)**
   * הוסיפי מסמכים בתיקייה data/documents/ בפורמטים PDF/MD/CSV.
   * בצעי הסרה/החלפה לשמות פרטיים, מזהים, סכומים מדויקים (אפשר טווחים).
   * שמרי ClientId/CaseId עקביים לכל התיק.
2. **יצירת סט שאלות הערכה**
   * הוסיפי ל־data/evaluation\_questions.json שאלות מסוגים: טקסטואליות, טבלאיות, ותרשים/גרף.
   * עבור כל שאלה צרפי תיאור קצר מה מצופה בתשובה.
3. **הפקת Ground Truth בהתאם לחוקי הצ׳אנקים**
   * לכל שאלה הגדירי ground\_truth ותיעוד עוגניםPageNumber, SectionType, TableId/FigureId.
   * ודאי שהראיות מצביעות לצ׳אנקים כפי שנוצרו בפייפליין.
4. **סט בדיקות RAGAS מלא והרצה**
   * צרי/עדכני data/ragas\_testset.json במבנהquestion, contexts, answer, ground\_truth.
   * הריצי מחולל המדדים (או השתמשי ב־notebook) להפקת:
     + Context Precision ≥ 0.75
     + Context Recall ≥ 0.70
     + Faithfulness ≥ 0.85
     + Table‑QA Accuracy ≥ 0.90

### לדוגמה

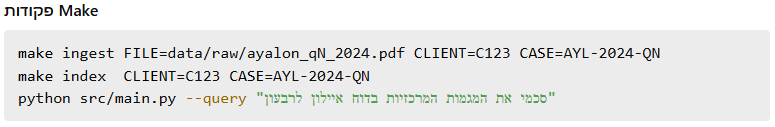
את מסמכי מאיה לעיתים לא ניתן למשוך אוטומטית דרך סקרייפר. נוח להוריד ידנית מהקישור שנתת ולשמור בתיקייה data/raw/. מכאן הפייפליין שלנו מטפל בכל השאר.

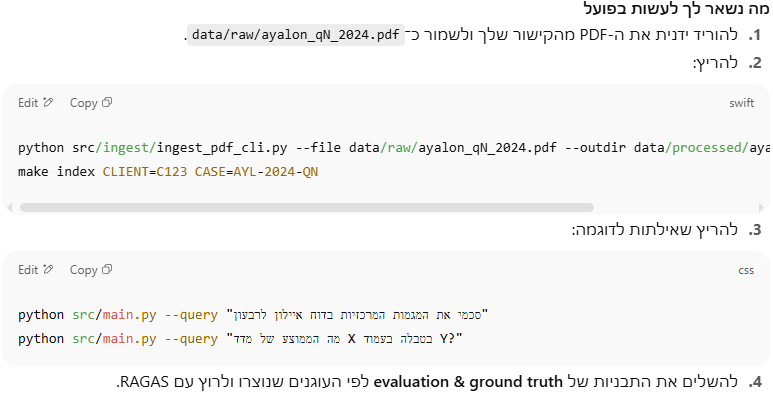
#### שלבים

1. הורידי את ה‑PDF מהקישור (דוח רבעוני של **איילון חברה לביטוח**; Maya ID: **1687518**), שמרי כ־data/documents/ayalon\_qN\_2024.pdf.
2. הריצי **ingest**: חילוץ טקסט + טבלאות + Anchors + Distillation ל־~5% → יצירת צ'אנקים עם מטא־דאטה.
3. הריצי **index**:
   * Dense: Pinecone (כולל מטא־דאטה לסינון ClientId/CaseId/PageNumber/SectionType...
   * Sparse: **TF‑IDF** (במקום BM25).
4. בדקי **retrieval** בשאילתות לדוגמה (טקסט, טבלה, גרף/תרשים).
5. צרי **סט הערכה** + **Truth Ground** לפי הצ'אנקים שנוצרו.

#### קבצים שיתווספו

* data/processed/ayalon\_qN\_2024/chunks.jsonl — כל צ'אנק בשורה (עם Anchors + מטא‑דאטה).
* data/processed/ayalon\_qN\_2024/tables/\*.csv — טבלאות שהומרו ל‑CSV.
* data/processed/ayalon\_qN\_2024/tables/\*.md — תיאור קצר לכל טבלה (ל‑dense retrieval).





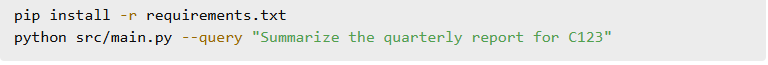
### איך מריצים לוקאלית (תקציר)

1) Fill .env with OPENAI and PINECONE keys

2) pip install -r requirements.txt

3) Configure src/config.yaml

4) Run: python src/main.py --query "Summarize Q2 incidents for Client C123"



5) Evaluate: use data/evaluation\_questions.json and data/ground\_truth.json

6) RAGAS: load data/ragas\_testset.json

## קוד לדוגמה (Python + LangChain + Pinecone)

from langchain.text\_splitter import RecursiveCharacterTextSplitter  
from langchain\_openai import OpenAIEmbeddings  
from langchain\_pinecone import PineconeVectorStore  
from rank\_bm25 import BM25Okapi  
import pinecone  
  
# 1. Chunking  
text\_splitter = RecursiveCharacterTextSplitter(  
 chunk\_size=800,  
 chunk\_overlap=100  
)  
  
chunks = text\_splitter.split\_text(document\_text)  
  
# 2. Metadata enrichment  
chunk\_metadata = []  
for i, chunk in enumerate(chunks):  
 meta = {  
 "FileName": "case123.pdf",  
 "PageNumber": i // 3 + 1,  
 "SectionType": "Analysis",  
 "ChunkSummary": chunk[:100],  
 "Keywords": ["contract", "client"],  
 "ClientId": "C123"  
 }  
 chunk\_metadata.append(meta)  
  
# 3. Indexing (Dense with Pinecone)  
embeddings = OpenAIEmbeddings(model="text-embedding-3-small")  
pinecone.init(api\_key="YOUR\_KEY", environment="us-east1-gcp")  
  
# יצירת אינדקס (פעם אחת)  
if "hybrid-rag" not in pinecone.list\_indexes():  
 pinecone.create\_index("hybrid-rag", dimension=1536, metric="cosine")  
  
index = pinecone.Index("hybrid-rag")  
  
vectorstore = PineconeVectorStore.from\_texts(  
 texts=chunks,  
 embedding=embeddings,  
 metadatas=chunk\_metadata,  
 index\_name="hybrid-rag",  
 namespace="C123" # Namespace לפי ClientId  
)  
  
# 4. Sparse Index (BM25)  
tokenized\_corpus = [chunk.split(" ") for chunk in chunks]  
bm25 = BM25Okapi(tokenized\_corpus)  
  
# 5. Hybrid Retrieval function  
def hybrid\_search(query, top\_k=10, namespace="C123"):  
 # Dense retrieval from Pinecone  
 dense\_results = vectorstore.similarity\_search(  
 query, k=top\_k, namespace=namespace, filter={"ClientId": "C123"}  
 )  
  
 # Sparse retrieval  
 tokenized\_query = query.split(" ")  
 scores = bm25.get\_scores(tokenized\_query)  
 sparse\_results = [chunks[i] for i in scores.argsort()[-top\_k:][::-1]]  
  
 # Combine + rerank (placeholder for cross-encoder)  
 combined = dense\_results + sparse\_results  
 return combined[:6]

## סיכום

בפרויקט זה הראינו כיצד ניתן להקים מערכת **Metadata‑Driven Hybrid RAG**: אחזור היברידי, ניהול מטא־דאטה עשיר, ריראנקר, ותמיכה במסמכים מרובים. בנוסף, שילבנו מבנה של Router Agent עם שלושה תתי‑סוכנים ייעודיים כדי לטפל בסוגי שאילתות שונים.

כבונוס, שולב שימוש ב‑**Pinecone** כ‑Vector Database מרכזי לניהול embedding‑ים בסקייל גדול, כולל סינון לפי Metadata ו‑namespaces לכל לקוח.