HelpMate AI Project_V2

September 2, 2025

```
[1]: !pip install sentence-transformers transformers torch diskcache pypdf
     ↪scikit-learn


import os, re, hashlib, warnings
from typing import List, Tuple, Dict
from pypdf import PdfReader
from sentence_transformers import SentenceTransformer, CrossEncoder
from transformers import pipeline
from diskcache import Cache
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
```

Requirement already satisfied: sentence-transformers in
/usr/local/lib/python3.12/dist-packages (5.1.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-
packages (4.55.4)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages
(2.8.0+cu126)
Collecting diskcache
  Downloading diskcache-5.6.3-py3-none-any.whl.metadata (20 kB)
Collecting pypdf
  Downloading pypdf-6.0.0-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-
packages (1.6.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages
(from sentence-transformers) (4.67.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages
(from sentence-transformers) (1.16.1)
Requirement already satisfied: huggingface-hub>=0.20.0 in
/usr/local/lib/python3.12/dist-packages (from sentence-transformers) (0.34.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.12/dist-packages
(from sentence-transformers) (11.3.0)
Requirement already satisfied: typing_extensions>=4.5.0 in
/usr/local/lib/python3.12/dist-packages (from sentence-transformers) (4.15.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-
packages (from transformers) (3.19.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-

packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.21.4)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (2.27.3)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in

```
/usr/local/lib/python3.12/dist-packages (from torch) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.11.1.6)
Requirement already satisfied: triton==3.4.0 in /usr/local/lib/python3.12/dist-
packages (from torch) (3.4.0)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-
packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-hub>=0.20.0->sentence-
transformers) (1.1.8)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers) (2025.8.3)
Downloading diskcache-5.6.3-py3-none-any.whl (45 kB)
                              45.5/45.5 kB
3.2 MB/s eta 0:00:00
Downloading pypdf-6.0.0-py3-none-any.whl (310 kB)
                              310.5/310.5 kB
18.4 MB/s eta 0:00:00
Installing collected packages: pypdf, diskcache
Successfully installed diskcache-5.6.3 pypdf-6.0.0
```

1 New Section

```python
[13]: # ---------------- CONFIG ----------------
      PDF_PATH = "/content/sample_data/Principal-Sample-Life-Insurance-Policy.pdf"
      CACHE_DIR = "/content/sample_data/cache"
      EMBED_MODEL = "all-MiniLM-L6-v2"
      RERANK_MODEL = "cross-encoder/ms-marco-MiniLM-L-6-v2"
      LLM_MODEL = "google/flan-t5-base"   # lightweight; replace with "gpt2" or
        ↪OpenAI API

      TOPK_RETRIEVE = 15     # recall first
      TOPK_RERANK = 3        # precision later
      CHUNK_WORDS = 180
      CHUNK_OVERLAP = 40
```

```python
RERANK_CLIP_WORDS = 100 # Reduced from 180

warnings.filterwarnings("ignore", category=UserWarning)
```

```python
# ---------------- HELPERS ----------------
def clean_text(text: str) -> str:
    return re.sub(r'\s+', ' ', (text or "")).strip()

def sentence_split(text: str) -> List[str]:
    # Simple sentence split; robust enough for policy docs
    parts = re.split(r'(?<=[\.\?\!])\s+', text)
    return [s.strip() for s in parts if s.strip()]

def chunk_page_sentences(sentences: List[str], max_words=CHUNK_WORDS,
 ↪overlap=CHUNK_OVERLAP) -> List[str]:
    chunks = []
    curr, count = [], 0
    for sent in sentences:
        w = len(sent.split())
        if count + w > max_words and curr:
            chunks.append(" ".join(curr))
            # create overlap
            if overlap > 0:
                back = []
                words_kept = 0
                for s in reversed(curr):
                    sw = len(s.split())
                    if words_kept + sw >= overlap:
                        back.insert(0, s)
                        break
                    words_kept += sw
                    back.insert(0, s)
                curr = back
                count = sum(len(s.split()) for s in curr)
            else:
                curr, count = [], 0
        curr.append(sent)
        count += w
    if curr:
        chunks.append(" ".join(curr))
    return chunks

def load_pdf_chunks(pdf_path: str, max_words=CHUNK_WORDS,
 ↪overlap=CHUNK_OVERLAP) -> Tuple[List[str], List[Dict]]:
    reader = PdfReader(pdf_path)
    chunks, metas = [], []
    for i, page in enumerate(reader.pages):
```

```python
            text = clean_text(page.extract_text())
            if not text:
                continue
            sents = sentence_split(text)
            page_chunks = chunk_page_sentences(sents, max_words=max_words,␣
    ↪overlap=overlap)
            for j, ch in enumerate(page_chunks):
                chunks.append(ch)
                metas.append({"page": i + 1, "chunk_id": j})
        return chunks, metas
```

[4]:
```python
# ---------------- EMBEDDING ----------------
class Embedder:
    def __init__(self, model_name=EMBED_MODEL):
        self.model = SentenceTransformer(model_name)
    def embed(self, texts):
        return self.model.encode(texts, convert_to_numpy=True)
```

[5]:
```python
# ---------------- VECTOR SEARCH (SKLEARN) ----------------
def build_index(embeddings):
    return np.array(embeddings)


def search_index(query_vec, index, top_k):
    sims = cosine_similarity([query_vec], index)[0]
    top_ids = sims.argsort()[-top_k:][::-1]
    return top_ids.tolist()
```

[10]:
```python
# ---------------- SEARCH + RERANK ----------------
cache = Cache(CACHE_DIR)
reranker = CrossEncoder(RERANK_MODEL)

def _pdf_signature(pdf_path: str) -> str:
    try:
        stat = os.stat(pdf_path)
        sig = f"{os.path.basename(pdf_path)}::{stat.st_mtime_ns}::{stat.
    ↪st_size}"
    except FileNotFoundError:
        sig = pdf_path
    return hashlib.md5(sig.encode()).hexdigest()

def clip_words(text: str, limit_words=RERANK_CLIP_WORDS) -> str:
    words = text.split()
    if len(words) <= limit_words:
        return text
    return " ".join(words[:limit_words])
```

```python
def retrieve(query: str, embedder: Embedder, index: np.ndarray, chunks:␣
 ↪List[str], metas: List[Dict],
              top_k: int = TOPK_RETRIEVE, pdf_sig: str = "") -> List[int]:
    key = ("retr", query, top_k, pdf_sig, EMBED_MODEL)
    if key in cache:
        return cache[key]
    qvec = embedder.embed([query])[0]
    ids = search_index(qvec, index, top_k)
    cache[key] = ids
    return ids


def rerank_ids(query: str, ids: List[int], chunks: List[str], top_k: int =␣
 ↪TOPK_RERANK) -> List[int]:
    # Shorten text to reduce truncation bias in cross-encoder
    pairs = [(query, clip_words(chunks[i])) for i in ids]
    scores = reranker.predict(pairs)
    ranked = sorted(zip(ids, scores), key=lambda x: x[1], reverse=True)
    return [i for i, _ in ranked[:top_k]]
```

```python
[12]: # ---------------- GENERATION ----------------
generator = pipeline("text2text-generation", model=LLM_MODEL, device=-1 )

def build_prompt(query: str, items: List[Tuple[str, Dict]]) -> str:
    # Compose a compact, citation-friendly context
    lines = []
    for t, m in items:
        p = m.get("page", "?")
        lines.append(f"[p.{p}] {t}")
    context = "\n".join(lines)
    prompt = (
        "You are a precise assistant that answers ONLY using the policy context␣
 ↪below.\n"
        "Extract the exact answer in your own words and include the page␣
 ↪number(s) as citations like [p.X].\n"
        "If the answer is not present, reply: Not found in policy.\n\n"
        f"Context:\n{context}\n\n"
        f"Question: {query}\n"
        "Answer:")
    return prompt

def generate_answer(query: str, chosen: List[int], chunks: List[str], metas:␣
 ↪List[Dict]) -> str:
    items = [(chunks[i], metas[i]) for i in chosen]
    prompt = build_prompt(query, items)
    out = generator(prompt, max_new_tokens=180,␣
 ↪do_sample=False)[0]["generated_text"]
    return out.strip()
```

```
Device set to use cpu
```

```python
[8]:  # ---------------- MAIN PIPELINE ----------------
      def run_pipeline(pdf_path: str, queries: List[str]) -> None:
          pdf_sig = _pdf_signature(pdf_path)

          # Step 1: Ingest
          chunks, metas = load_pdf_chunks(pdf_path, CHUNK_WORDS, CHUNK_OVERLAP)
          print(f"Loaded {len(chunks)} chunks from PDF.")

          # Step 2: Embed + index
          embedder = Embedder()
          embeddings = embedder.embed(chunks)
          index = build_index(embeddings)

          # Step 3: Query loop
          for q in queries:
              # retrieve a wide set, then rerank down to the best
              cand_ids = retrieve(q, embedder, index, chunks, metas,␣
          ↪top_k=TOPK_RETRIEVE, pdf_sig=pdf_sig)
              best_ids = rerank_ids(q, cand_ids, chunks, top_k=TOPK_RERANK)

              # Pretty print top-3 with pages
              print("\n=============================")
              print("Query:", q)
              print("\nTop Retrieved (after rerank):")
              for rank, i in enumerate(best_ids, 1):
                  pg = metas[i]["page"]
                  print(f"{rank}) [p.{pg}] {chunks[i][:220]}...")

              # Generate final answer
              answer = generate_answer(q, best_ids, chunks, metas)
              print("\nFinal Answer:")
              print(answer)
              print("=============================")
```

```python
[14]: # ---------------- RUN ----------------
      if __name__ == "__main__":
          # Example queries - replace with your own
          queries = [
              "What is the grace period for premium payment?",
              "Who is eligible under this policy?",
              "List the major exclusions."
          ]
          if not os.path.exists(PDF_PATH) or os.path.isdir(PDF_PATH):
              print(" Please set PDF_PATH at the top of the script to your policy␣
          ↪PDF file.")
```

```
    else:
        run_pipeline(PDF_PATH, queries)
```

Loaded 160 chunks from PDF.

Token indices sequence length is longer than the specified maximum sequence length for this model (719 > 512). Running this sequence through the model will result in indexing errors

==============================
Query: What is the grace period for premium payment?

Top Retrieved (after rerank):
1) [p.20] This policy has been updated effective January 1, 2014 PART II -
POLICY ADMINISTRATION GC 6004 Section B - Premiums, Page 1 Section B - Premiums
Article 1 - Payment Responsibility; Due Dates; Grace Period The Policyholde…
2) [p.20] "Grace Period" means the first 31- day period following a premium due
date. The Group Policy will remain in fo rce until the end of the Grace Period,
unless the Group Policy has been terminated by notice as described in …
3) [p.23] This policy has been updated effective January 1, 2014 PART II -
POLICY ADMINISTRATION GC 6005 Section C - Policy Termination, Page 1 Section C -
Policy Termination Article 1 - Failure to Pay Premium This Group Policy wi…

Final Answer:
The first premium is due on the Date of Issue of this Group Policy. Each premium
thereafter will be due on the first of each Insurance Month. Except for the
first premium, a Grace Period of 31 days will be allowed for payment of premium.
"Grace Period" means the first 31- day period following a premium due date. The
Group Policy will remain in fo rce until the end of the Grace Period, unless the
Group Policy has been terminated by notice as described in PART II, Section C.
[p.20] "Grace Period" means the first 31- day period following a premium due
date. The Group Policy will remain in fo rce until the end of the Grace Period,
unless the Group Policy has been terminated by notice as described in PART II,
Section C.
==============================

==============================
Query: Who is eligible under this policy?

Top Retrieved (after rerank):
1) [p.27] This policy has been updated effective January 1, 2014 PART III -
INDIVIDUAL REQUIREMENTS AND RIGHTS GC 6006 Section A - Eligibility, Page 2 If a
Member's Dependent is employed and is covered under group term life covera…
2) [p.26] This policy has been updated effective January 1, 2014 PART III -
INDIVIDUAL REQUIREMENTS AND RIGHTS GC 6006 Section A - Eligibility, Page 1 PART
III - INDIVIDUAL REQUIREMENTS AND RIGHTS Section A - Eligibility Article 1…
3) [p.33] When insurance under this Group Policy replaces coverage under a Prior
Policy, the Period of Limited Activity requirement may be waived for those

Dependent spouses' who: (1) are eligible and enrolled under this Group Pol…

Final Answer:
A person will be eligible for Member Life Insurance on the date the person completes 30 consecutive days of continuous Active Work with the Policyholder as a Member. In no circumstance will a person be eligible for Member Life Insurance under this Group Policy if the person is eligible under any other Group Term Life Insurance policy underwritten by The Principal. Article 1 – Member Life Insurance A person will be eligible for Member Life Insurance on the date the person completes 30 consecutive days of continuous Active Work with the Policyholder as a Member. In no circumstance will a person be eligible for Member Life Insurance under this Group Policy if the person is eligible under any other Group Term Life Insurance policy underwritten by The Principal. Article 2 – Member Accidental Death and Dismemberment Insurance A person will be eligible for Member Accidental Death and Dismemberment Insurance on the latest of:
==============================

==============================
Query: List the major exclusions.

Top Retrieved (after rerank):
1) [p.17] if the Member is to contribute part of the premium, maintain the following participation percentages with respect to eligible employees and Dependents, excluding those for whom Proof of Good Health is not satisfactory to…
2) [p.14] Terminally Ill A Member will be considered Terminally Ill, for Accelerated Benefits, if he or she has experienced a Qualifying Event and is expected to die within 12 months of the date he or she requests payment of Accel…
3) [p.42] Purchase Qualification A Member will qualify for individual purchase if insurance under this Group Policy terminates and: (1) the Member's total Lif e Insurance, or any portion of it, terminates because he or she ends Ac…

Final Answer:
[p.15] Terminally Ill A Member will be considered Terminally Ill, for Accelerated Benefits, if he or she has experienced a Qualifying Event and is expected to die within 12 months of the date he or she requests payment of Accelerated Benefits. Total Disability; Totally Disabled A Member's inability, as determined by The Principal, due to sickness or injury, to perform the majority of the material duties of any occupation for which he or she is or may reasonably become qualified based on education, training or experience. Written or Writing [p.42] Purchase Qualification A Member will qualify for individual purchase if insurance under this Group Policy terminates and: (1) the Member's total Lif e Insurance, or any portion of it, terminates because he or she ends Active Work or
==============================

```
[ ]:
```