```
!pip install -U transformers sentence-transformers langchain langchain-community pypdf tqdm accelerate bitsandbytes
```

显示隐藏的输出项

```
import torch
from transformers import pipeline, AutoTokenizer
from sentence_transformers import SentenceTransformer, CrossEncoder, util
from langchain_community.document_loaders import PyPDFLoader
from langchain_text_splitters import RecursiveCharacterTextSplitter
import os
import pandas as pd
from tqdm import tqdm
import time
import requests
import zipfile
import io
```

```
#colab
from google.colab import drive
drive.mount('/content/drive')

from google.colab import userdata
hf_token = userdata.get('A5')    # 读取你输入的 token

from huggingface_hub import login

login(token=hf_token)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#kaggle
from kaggle_secrets import UserSecretsClient
user_secrets = UserSecretsClient()
secret_value_0 = user_secrets.get_secret("HF")

from huggingface_hub import login

login(token=secret_value_0)
print("Successfully logged in to Hugging Face!")
```

```
Successfully logged in to Hugging Face!
```

```
import re

def extract_expected_from_needle(needle: str) -> str:
    """
    更健壮地从 needle 字符串中提取 expected value。
    支持形式：
        - The secret fruit is "grape".
        - The secret shape is a "star".
        - The secret number is 42.
        - The secret thing is the big box.
    返回: 提取并去除引号/冠词/尾点后的核心答案 (不做小写处理，后续比较时再统一小写)
    """
    needle = needle.strip()
    # 1) 优先匹配带引号的值，允许前面有 a/an/the 也能匹配
    m = re.search(r'\bis\s+(?:a|an|the)?\s*"([^"]+)"', needle, flags=re.IGNORECASE)
    if m:
        return m.group(1).strip()

    # 2) 如果没有引号，尝试匹配 is 后面直到句点或字符串结尾的那一段
    m2 = re.search(r'\bis\s+(.*?)(?:\.\s*$|$)', needle, flags=re.IGNORECASE)
    if m2:
        val = m2.group(1).strip()
        # 如果以 a/an/the 开头，去掉冠词
        val = re.sub(r'^(?:a|an|the)\s+', '', val, flags=re.IGNORECASE).strip()
        # 去掉可能残留的引号或句尾标点
        val = val.strip(' ."\'')
        return val

    # 兜底：返回原始 needle（不太可能走到这里）
    return needle
```

```
import os
import pandas as pd
from langchain_community.document_loaders import PyPDFLoader # 确保已导入 [cite: 5]
from tqdm import tqdm # 确保已导入 [cite: 9]
```

```
def load_test_cases(company_path):
    """
    加载基线评估的测试用例，匹配老师的方法。
    此函数在每个子文件夹中使用 PyPDFLoader 加载 *_TextNeedles.pdf* 文件。
    """
    test_cases = []
    company_name = os.path.basename(company_path)

    # 1. 获取所有子文件夹 (例如 "AIG_5Pages", "AIG_10Pages"...)
    subfolders = [f for f in os.listdir(company_path) if os.path.isdir(os.path.join(company_path, f))]
    # 2. 按页数对子文件夹进行排序
    subfolders = sorted(subfolders, key=lambda x: int(''.join(filter(str.isdigit, x))) if any(ch.isdigit() for ch in x) els

    print(f"为 {company_name} 找到了 {len(subfolders)} 个子文件夹...")

    # 3. 遍历每个子文件夹
    for subfolder in tqdm(subfolders, desc="正在处理子文件夹"):
        subfolder_path = os.path.join(company_path, subfolder)

        # 4. 定义基线需要的文件路径
        # 老师的基线是纯文本的，所以我们使用 _TextNeedles.pdf
        pdf_file_name = f"{subfolder}_TextNeedles.pdf"
        pdf_path = os.path.join(subfolder_path, pdf_file_name)

        needles_file = os.path.join(subfolder_path, "needles.csv")
        questions_file = os.path.join(subfolder_path, "prompt_questions.txt")

        # 6. 加载 Needles 和 Questions
        try:
            needles_df = pd.read_csv(needles_file, header=None)
            needles_df.columns = ['needle']

            with open(questions_file, 'r', encoding="utf-8") as f:
                questions = [line.strip() for line in f.readlines() if line.strip()]
        except Exception as e:
            print(f"  加载 {subfolder} 的元数据时出错: {e}")
            continue

        # 7. 【关键】: 使用 PyPDFLoader 加载 PDF (老师的方法 )
        try:
            loader = PyPDFLoader(pdf_path)
            docs = loader.load_and_split()     # 更稳定
            full_text = " ".join(page.page_content for page in docs)
            print(len(full_text))
            doc_id = pdf_file_name
        except Exception as e:
            print(f"  加载 PDF {pdf_path} 时出错: {e}")
            continue

        # 8. 为这个文档构建所有测试用例
        # 假设 needles_df 和 questions 列表长度一致
        if len(needles_df) != len(questions):
            print(f"Warning: mismatch in {subfolder} — needles={len(needles_df)}, questions={len(questions)}")

        for ( _, row), question in zip(needles_df.iterrows(), questions):
            needle = row['needle']

            try:
                # 从 needle 中提取 expected_value
                expected_value = extract_expected_from_needle(needle)

            except Exception:
                expected_value = needle

            test_cases.append({
                "company": company_name,
                "doc_id": doc_id,
                "full_text": full_text,
                "question": question,                    # 顺序对应
                "expected_answer": expected_value,
                "needle": needle
            })

    for i in range(3):
        print(f"Q{i+1}: {test_cases[i]['question']}")
        print(f"Needle: {test_cases[i]['needle']}")
        print(f"Expected: {test_cases[i]['expected_answer']}")
        print()
```

```
        print(f"\n总共加载的测试用例: {len(test_cases)}")
        return test_cases
```

```
# Colab
company_path = "/content/drive/MyDrive/Colab Notebooks/GoldmanSachs"
test_cases = load_test_cases(company_path)
```

```
为 GoldmanSachs 找到了 8 个子文件夹...
正在处理子文件夹:  12%|██          | 1/8 [00:00<00:01,  5.39it/s]3580
正在处理子文件夹:  25%|████        | 2/8 [00:00<00:01,  4.04it/s]8649
正在处理子文件夹:  38%|██████      | 3/8 [00:01<00:02,  1.94it/s]46207
正在处理子文件夹:  50%|████████    | 4/8 [00:03<00:04,  1.19s/it]179208
正在处理子文件夹:  62%|██████████  | 5/8 [00:05<00:04,  1.35s/it]309016
正在处理子文件夹:  75%|████████████  | 6/8 [00:07<00:03,  1.81s/it]428813
正在处理子文件夹:  88%|██████████████ | 7/8 [00:11<00:02,  2.54s/it]658431
正在处理子文件夹: 100%|████████████████| 8/8 [00:18<00:00,  2.34s/it]871174
Q1: What is the secret flower in the document?
Needle: The secret flower is "lavender".
Expected: lavender

Q2: What is the secret tool in the document?
Needle: The secret tool is "scissors".
Expected: scissors

Q3: What is the secret shape in the document?
Needle: The secret shape is a "star".
Expected: star


总共加载的测试用例: 165
```

```
#Kaggle
import os

company_path = "/kaggle/input/glodmansachs/GoldmanSachs"

print("Listing files in root folder:")
print(os.listdir(company_path))

# --- Prepare test cases directly from Google Drive ---
test_cases = load_test_cases(company_path)
```

```
Listing files in root folder:
['GoldmanSachs_150Pages', 'GoldmanSachs_100Pages', 'GoldmanSachs_75Pages', 'prompt_questions.txt', 'GoldmanSachs_25Pages', 'GoldmanSachs_50Pa
```

```
# --- RAG Component Definitions ---

class Chunker:
    def __init__(self, chunk_size=500, chunk_overlap=100):
        self.splitter = RecursiveCharacterTextSplitter(
            chunk_size=chunk_size,
            chunk_overlap=chunk_overlap
        )
    def chunk(self, document_text):
        return self.splitter.split_text(document_text)

class Embedder:
    def __init__(self, model_name="BAAI/bge-small-en-v1.5"):
        self.model = SentenceTransformer(model_name)
    def embed(self, texts, batch_size=16):
        return self.model.encode(texts, convert_to_tensor=True, batch_size=batch_size, show_progress_bar=False)

class Retriever:
    def __init__(self, embedder):
        self.embedder = embedder
    def retrieve(self, query, chunks, chunk_embeddings, top_k=5):
        query_embedding = self.embedder.embed(query)
        similarities = util.pytorch_cos_sim(query_embedding, chunk_embeddings)[0]
        actual_k = min(top_k, len(chunks))
        top_k_indices = torch.topk(similarities, k=actual_k).indices
        return [chunks[i] for i in top_k_indices]

class Reranker:
    def __init__(self, model_name='cross-encoder/ms-marco-MiniLM-L-6-v2'):
        self.model = CrossEncoder(model_name)
    def rerank(self, query, chunks, top_k=5):
        pairs = [[query, chunk] for chunk in chunks]
        scores = self.model.predict(pairs)

        # Combine chunks with scores and sort
        reranked_chunks = sorted(zip(chunks, scores), key=lambda x: x[1], reverse=True)
```

```
                actual_k = min(top_k, len(reranked_chunks))
                return [chunk for chunk, score in reranked_chunks[:actual_k]]

from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline

class Generator:
        """
        Universal generator for RAG pipelines.
        Supports:
        - HuggingFaceTB/SmolLM-135M-Instruct
        - google/gemma-2b-it (and other Causal LMs)
        """

        def __init__(self, model_name="HuggingFaceTB/SmolLM-135M-Instruct", device=None):
                self.model_name = model_name

                # Device selection
                if device is None:
                        self.device = 0 if torch.cuda.is_available() else -1
                else:
                        self.device = device

                # Detect model type
                model_lower = self.model_name.lower()
                self.is_smol = "smol" in model_lower

                # Load model
                if self.is_smol:
                        # SmolLM 模型部分
                        print(f"Loading SmolLM model: {model_name}")
                        self.pipeline = pipeline(
                                "text-generation",
                                model=model_name,
                                device=self.device,
                        )
                        # SmolLM does not have a separate tokenizer object
                        self.tokenizer = None
                        print(f"SmolLM loaded on device: {self.device}")

                else:
                        # 通用 Causal LM (LLaMA, Gemma, Phi, etc.) 加载部分
                        print(f"Loading general Causal LM: {model_name}. This may take a moment...")
                        try:
                                self.tokenizer = AutoTokenizer.from_pretrained(model_name)

                                # Use 8-bit loading to prevent OOM
                                self.model = AutoModelForCausalLM.from_pretrained(
                                        model_name,
                                        device_map="auto",
                                        torch_dtype="auto"
                                )

                                self.pipeline = pipeline(
                                        "text-generation",
                                        model=self.model,
                                        tokenizer=self.tokenizer
                                )
                                print(f"Successfully loaded {model_name}.")
                                if torch.cuda.is_available():
                                          print(f"Model is on device: {self.model.device}")

                        except Exception as e:
                                print(f"Failed to load model {model_name} with AutoModelForCausalLM.")
                                raise e

                # System prompt
                self.system_prompt = (
                        "You are a helpful assistant that answers questions based on the provided context. "
                        "Provide direct, concise answers."
                )

        def generate(self, query, context_chunks, max_new_tokens=150):
                """
                Generate answer based on query and retrieved context chunks.
                """
                context_text = "\n\n".join(context_chunks)
                user_question = f"Context:\n{context_text}\n\nQuestion: {query}\nAnswer:"

                if self.is_smol:
                        # SmolLM uses the simple string prompt
                        prompt = f"{self.system_prompt}\n{user_question}"
```

```
                        output = self.pipeline(
                                prompt,
                                max_new_tokens=max_new_tokens,
                                return_full_text=False, # Try to get only new text
                                do_sample=False,                # ✅ 关闭随机采样
                                temperature=0.0,                # ✅ 固定为贪心搜索
                                top_p=1.0
                        )

                        generated_text = output[0]['generated_text']
                        # If return_full_text=False didn't work, strip prompt
                        if generated_text.startswith(prompt):
                                return generated_text[len(prompt):].strip()
                        return generated_text.strip()

                else:
                        # --- MODIFIED PART (Fix for TemplateError) ---
                        #
                        # We combine the system prompt and user question into a single
                        # user message, as Gemma's template doesn't support a 'system' role.
                        #
                        full_user_prompt = f"{self.system_prompt}\n\n{user_question}"

                        messages = [
                                # NO 'system' role
                                {"role": "user", "content": full_user_prompt}
                        ]

                        # The 'text-generation' pipeline will handle this list
                        # and apply the correct (Gemma) chat template.
                        output = self.pipeline(
                                messages,
                                max_new_tokens=max_new_tokens,
                                return_full_text=False, # This is key
                                do_sample=False, # For factual QA
                                temperature=0.0, # For factual QA
                                top_p=1.0
                        )

                        # With return_full_text=False, this should just be the answer.
                        return output[0]['generated_text'].strip()


class RAGPipeline:
        def __init__(self, chunker, embedder, retriever, generator, reranker=None):
                self.chunker = chunker
                self.embedder = embedder
                self.retriever = retriever
                self.generator = generator
                self.reranker = reranker

        def prepare_document(self, document_text):
                chunks = self.chunker.chunk(document_text)
                embeddings = self.embedder.embed(chunks)
                return chunks, embeddings

        def query(self, query, chunks, chunk_embeddings, top_k=5, rerank_top_k=3):
                # 1. Retrieve
                retrieved_chunks = self.retriever.retrieve(query, chunks, chunk_embeddings, top_k=top_k)

                # 2. (Optional) Rerank
                if self.reranker:
                        final_chunks = self.reranker.rerank(query, retrieved_chunks, top_k=rerank_top_k)
                else:
                        final_chunks = retrieved_chunks

                # 3. Generate
                answer = self.generator.generate(query, final_chunks)
                return answer, final_chunks
```

```
import re
from tqdm import tqdm
import pandas as pd
import time

def evaluate_rag(test_cases, rag_pipeline, top_k=5, rerank_top_k=3, use_llm=True, log_errors=True):
        total_correct = 0
        total_time = 0
        errors = []

        print("Preparing all documents (chunking and embedding)...")
```

```
    embeddings_dict, chunks_dict = {}, {}
    docs = pd.DataFrame(test_cases)

    for _, row in tqdm(docs.iterrows(), total=len(docs)):
        chunks, embeddings = rag_pipeline.prepare_document(row['full_text'])
        chunks_dict[row['doc_id']] = chunks
        embeddings_dict[row['doc_id']] = embeddings

    print("\n🔹 Starting evaluation...")

    for case in tqdm(test_cases):
        doc_id = case['doc_id']
        chunks = chunks_dict[doc_id]
        embeddings = embeddings_dict[doc_id]
        start_time = time.time()

        # Retrieval-only 模式
        if not use_llm:
            context_chunks = rag_pipeline.retriever.retrieve(
                case['question'], chunks, embeddings, top_k=top_k
            )
            pattern = re.escape(case['needle'].lower())
            is_correct = any(re.search(pattern, chunk.lower()) for chunk in context_chunks)

        # End-to-end 模式
        else:
            answer, context_chunks = rag_pipeline.query(
                case['question'], chunks, embeddings, top_k=top_k, rerank_top_k=rerank_top_k
            )
            pattern = re.escape(case['expected_answer'].lower())
            is_correct = bool(re.search(pattern, answer.lower()))

        total_time += time.time() - start_time
        if is_correct:
            total_correct += 1
        elif log_errors:
            errors.append({
                "company": case["company"],
                "doc_id": doc_id,
                "question": case["question"],
                "expected_answer": case["expected_answer"],
                "needle": case["needle"],
                "retrieved_context": "\n---\n".join(context_chunks[:2])  # 前两段上下文
            })

    accuracy = total_correct / len(test_cases) * 100
    print(f"\n==================================")
    print(f"EVALUATION RESULTS")
    print(f"==================================")
    print(f"Mode: {'End-to-end' if use_llm else 'Retrieval-only'}")
    print(f"Accuracy: {accuracy:.2f}% ({total_correct}/{len(test_cases)} correct)")
    print(f"Total Time: {total_time:.2f}s ({total_time/len(test_cases)*1000:.2f} ms/query)")

    if log_errors:
        print(f"\n❌ Misclassified samples: {len(errors)}")
        for e in errors[:5]:
            print("-" * 80)
            print(f"Question: {e['question']}")
            print(f"Expected: {e['expected_answer']}")
            print(f"Needle: {e['needle']}")
            print(f"Retrieved snippet:\n{e['retrieved_context'][:400]}...")
        print("（仅显示前5个错误样本，完整错误保存在返回值 errors 中）")

    return accuracy, errors
```

```
# --- Baseline Configuration ---
print("--- Configuring Baseline RAG Pipeline ---")
baseline_chunker = Chunker(chunk_size=500, chunk_overlap=100)
baseline_embedder = Embedder(model_name="BAAI/bge-small-en-v1.5")
baseline_retriever = Retriever(baseline_embedder)
baseline_generator = Generator("HuggingFaceTB/SmolLM-135M-Instruct")

baseline_pipeline = RAGPipeline(baseline_chunker, baseline_embedder, baseline_retriever, baseline_generator)
```

```
--- Configuring Baseline RAG Pipeline ---
Loading SmolLM model: HuggingFaceTB/SmolLM-135M-Instruct
Device set to use cuda:0
SmolLM loaded on device: 0
```

```
# %%
# --- Running Baseline Evaluation (Retrieval-Only) ---
print("\n--- [Baseline] Starting Retrieval-Only Evaluation ---")
results_baseline_retrieval = evaluate_rag(test_cases, baseline_pipeline, top_k=5, use_llm=False)
```

```
--- [Baseline] Starting Retrieval-Only Evaluation ---
Preparing all documents (chunking and embedding)...
100%|██████████| 8/8 [00:12<00:00, 1.58s/it]

🔹 Starting evaluation...
100%|██████████| 165/165 [00:02<00:00, 76.95it/s]
==================================
EVALUATION RESULTS
==================================
Mode: Retrieval-only
Accuracy: 83.64% (138/165 correct)
Total Time: 2.14s (12.99 ms/query)
```

```
# %%
# --- Running Baseline Evaluation (End-to-End) ---
print("\n--- [Baseline] Starting End-to-End Evaluation ---")
results_baseline_full = evaluate_rag(test_cases, baseline_pipeline, top_k=5, use_llm=True)
```

```
--- [Baseline] Starting End-to-End Evaluation ---
Preparing all documents (chunking and embedding)...
100%|██████████| 165/165 [05:31<00:00, 2.01s/it]

🔹 Starting evaluation...
100%|██████████| 165/165 [16:23<00:00, 5.96s/it]
==================================
EVALUATION RESULTS
==================================
Mode: End-to-end
Accuracy: 36.36% (60/165 correct)
Total Time: 983.63s (5961.42 ms/query)

❌ Misclassified samples: 105
--------------------------------------------------------------------------------
Question: What is the secret tool in the document?
Expected: scissors
Needle:   The secret tool is "scissors".
Retrieved snippet:
Annual Report
2023
THE GOLDMAN SACHS GROUP , INC.
The secret flower is "lavender". The secret tool is "scissors". 1
Fellow Shareholders:
Twenty twenty-three was a year of
execution for Goldman Sachs. We took
swift, decisive action to refocus the
firm’ s strategy while at the same time
strengthening our core businesses, and
I’ m proud of the progress we made.
We put the firm in a stronger posi...
--------------------------------------------------------------------------------
Question: What is the secret office supply in the document?
Expected: envelope
Needle:   The secret office supply is an "envelope".
Retrieved snippet:
David Solomon
  “  Over the past year,
we have continued to
enhance our franchise.
  ... It’ s clear that our
One Goldman Sachs
operating ethos and
client-centric approach
are having an impact.”
The secret office supply is an "envelope".
---
•        Run

world-class,

differentiated

and

durable
```

```
# --- Experiment 1 Configuration ---
print("--- Configuring Experiment 1: Upgrading Generator ---")
# Reusing chunker, embedder, retriever from baseline
```

```
exp1_generator = Generator(model_name="google/gemma-2b-it")
#exp1_embedder = Embedder(model_name="BAAI/bge-base-en-v1.5")
exp1_pipeline = RAGPipeline(baseline_chunker, baseline_embedder, baseline_retriever, exp1_generator)

# %%
# --- Running Experiment 1 Evaluation (End-to-End) ---
print("\n--- [Experiment 1] Starting End-to-End Evaluation ---")
results_exp1 = evaluate_rag(test_cases, exp1_pipeline, top_k=5, use_llm=True)
```

```
--- Configuring Experiment 1: Upgrading Generator ---
Loading general Causal LM: google/gemma-2b-it. This may take a moment...
tokenizer_config.json: 100%  34.2k/34.2k [00:00<00:00, 3.01MB/s]
tokenizer.model: 100%  4.24M/4.24M [00:00<00:00, 7.91MB/s]
tokenizer.json: 100%  17.5M/17.5M [00:00<00:00, 76.5MB/s]
special_tokens_map.json: 100%  636/636 [00:00<00:00, 59.9kB/s]
config.json: 100%  627/627 [00:00<00:00, 59.4kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 100%  13.5k/13.5k [00:00<00:00, 1.36MB/s]
Fetching 2 files: 100%  2/2 [04:35<00:00, 275.72s/it]
model-00001-of-00002.safetensors: 100%  4.95G/4.95G [04:35<00:00, 74.1MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:05<00:00, 13.7MB/s]
Loading checkpoint shards: 100%  2/2 [00:20<00:00, 8.56s/it]
generation_config.json: 100%  137/137 [00:00<00:00, 8.97kB/s]
Device set to use cuda:0
Successfully loaded google/gemma-2b-it.
Model is on device: cuda:0

--- [Experiment 1] Starting End-to-End Evaluation ---
Preparing all documents (chunking and embedding)...
100%|██████████| 165/165 [05:30<00:00, 2.00s/it]

🔹 Starting evaluation...
100%|██████████| 165/165 [05:14<00:00, 1.91s/it]
==================================
EVALUATION RESULTS
==================================
Mode: End-to-end
Accuracy: 76.97% (127/165 correct)
Total Time: 314.37s (1905.30 ms/query)

❌ Misclassified samples: 38
--------------------------------------------------------------------------------
Question: What is the secret flower in the document?
Expected: lavender
Needle:  The secret flower is "lavender".
Retrieved snippet:
Annual Report
2023
THE GOLDMAN SACHS GROUP , INC.
The secret flower is "lavender". The secret tool is "scissors". 1
Fellow Shareholders:
Twenty twenty-three was a year of
execution for Goldman Sachs. We took
swift, decisive action to refocus the
firm's strategy while at the same time
```

```
# --- Experiment 2 Configuration ---
print("--- Configuring Experiment 2: Introducing Reranker ---")
# Reusing the powerful generator from Experiment 1
exp2_reranker = Reranker(model_name='cross-encoder/ms-marco-MiniLM-L-6-v2')

exp2_pipeline = RAGPipeline(
        chunker=baseline_chunker,
        embedder=baseline_embedder,
        retriever=baseline_retriever,
        generator=exp1_generator, # Use the upgraded generator
        reranker=exp2_reranker
)

# %%
# --- Running Experiment 2 Evaluation (End-to-End) ---
# Note: We increase the initial retrieval top_k to 10 to provide more candidates for the reranker.
# rerank_top_k=3 means the top 3 reranked chunks are passed to the LLM.
print("\n--- [Experiment 2] Starting End-to-End Evaluation ---")
results_exp2 = evaluate_rag(test_cases, exp2_pipeline, top_k=10, rerank_top_k=3, use_llm=True)
```

```
Question: What is the secret fruit in the document?
Expected: grape
```

```
Needle:   The secret fruit is a "grape".
--- Configuring Experiment 2: Introducing Reranker ---
Retrieved snippet:
config.json: 100%                                                  794/794 [00:00<00:00, 80.5kB/s]
Certain Relationships and Related Transactions, and Director
Independence 241
Item 14
model.safetensors: 100%                                          90.9M/90.9M [00:26<00:00, 3.20MB/s]
Principal Accountant Fees and Services 241
PART IV 24
tokenizer_config.json:      1.33k/? [00:00<00:00, 63.2kB/s]
Item 15
vocab.txt:      232k/? [00:00<00:00, 7.84MB/s]
Item 16
tokenizer.json:      711k/? [00:00<00:00, 27.8MB/s]
SIGNATURES 246
special_tokens_map.json: 100%                                      132/132 [00:00<00:00, 8.62kB/s]
Goldman Sachs-2023-Form10-K
README.md:      3.67k/? [00:00<00:00, 215kB/s]
The secret ve   able is a "mushroom". PART I
Item 1. Business
Introduction
--- [Experiment 2] Starting End-to-End Evaluation ---
Goldman Sachs is a leading global financial institution that
Preparing all documents (chunking and embedding)...
delivers a broad r...
100%|██████████████| 165/165 [05:42<00:00,  2.08s/it]
Question: What is the secret animal #4 in the document?
🔹 Starting evaluation...
Expected: horse
100%|██████████████| 165/165 [04:10<00:00,  1.52s/it]
Needle:   The secret animal #4 is a "horse".
==================================================
Retrieved snippet:
EVALUATION RESULTS
David Solomon
==================================================
“ Over the past year,
Mode: End-to-end
we have continued to
Accuracy: 83.03% (137/165 correct)
enhance our franchise.
Total Time: 250.39s (1517.54 ms/query)
... It's clear that our
One Goldman Sachs
❌ Misclassified samples: 28
operating ethos and
--------------------------------------------------------------------------------
client-centric approach
Question: What is the secret shape in the document?
are having an impact.
Expected: star
The secret animal #2 is a "koala". ANNUAL REPORT 2023 LETTER TO SHAREHOLDERS
Needle:   The secret shape is a "star".
The secret office supply is an "envelope". David Solomon
Retrieved snippet:
3
Pyn
• “2023 was a year
of execution for
world-class
Goldman Sachs.”
The secre...
differentiated
--------------------------------------------------------------------------------
Question: What is the secret object #3 in the document?
and
Expected: plate
Needle:   The secret object #3 is a "plate".
durable
Retrieved snippet:
The secret object #5 is a "candle". “There’s no
businesses
ambiguity about
•
who we are —
a preeminent
global investment
There’s no ambiguity about who we are — a preeminent global investment bank, serving the most
bank — and
important companies, institutions and individuals in the world — and we’re playing to our strengths
we’re playing to
as a trusted advisor, proven risk manager and experienced asset manager.
our strengths.
The secret shape is a "star"...
It
--------------------------------------------------------------------------------
David SolomonThe secret transportation is an "airplane". ANNUAL REPORT 2023 LETTER TO SHAREHOLDERS
Question: What is the secret clothing in the document?
Investing in Our Culture
Expected: dress
Needle:   We made a significant commitment to
Retrieved snippet:
Annual Report
2023
THE GOLDMAN SACHS GROUP, INC.
The secret flower is "lavender". The secret tool is "scissors". 1
Fellow Shareholders:
Twenty twenty-three was a year of
🔹 Starting evaluation...
execution for Goldman Sachs. We took
100%|██████████████| 165/165 [03:43<00:00,  1.36s/it]
swift, decisive action to refocus the
==================================================
firm's strategy while at the same time
EVALUATION RESULTS
strengthening our core businesses, and
I'm proud of the progress we made.
Mode: End-to-end
We put the firm in a stronger posi...
Accuracy: 94.55% (156/165 correct)
--------------------------------------------------------------------------------
```

```
# --- Experiment 3a Configuration (Smaller Chunks) ---
print("--- Configuring Experiment 3a: Smaller Chunks (size=256) ---")
exp3a_chunker = Chunker(chunk_size=256, chunk_overlap=50)

exp3a_pipeline = RAGPipeline(
        chunker=exp3a_chunker,
        embedder=baseline_embedder,
        retriever=baseline_retriever,
        generator=exp1_generator,
        reranker=exp2_reranker
)

# %%
# --- Running Experiment 3a Evaluation ---
print("\n--- [Experiment 3a] Starting End-to-End Evaluation ---")
results_exp3a = evaluate_rag(test_cases, exp3a_pipeline, top_k=10, rerank_top_k=3, use_llm=True)
```

Total Time: 223.56s (1354.88 ms/query)
Misclassified samples: 9

Question: What is the secret animal #4 in the document?
Expected: horse
Needle: The secret animal #4 is a "horse".
Retrieved snippet:
Goldman Sachs.
The secret animal #5 is a "rabbit". ANNUAL REPORT 2023 LETTER TO SHAREHOLDERS
The firm's performance has produced strong
returns for our shareholders. Over the past
five
years
book
Labor
per
share
has
grown
by
appr
oximately
50 percent,
and
our
...

Question: What is the secret shape in the document?
Expected: star
Needle: The secret shape is a "star".
Retrieved snippet:
important companies, institutions and individuals in the world — and we're playing to our strengths
as a trusted advisor, proven risk manager and experienced asset manager.
The secret shape is a "star". ANNUAL REPORT 2023 LETTER TO SHAREHOLDERS

The secret object #1 is a "door". The secret sport is "skiing". THE GOLDMAN SACHS GROUP, INC. ANDSUBSIDIARIES
ANNUAL REPORT ONFORM 10-K FOR THE FISCA...

Question: What is the secret clothing in the document?
Expected: dress
Needle: The secret clothing is a "dress".
Retrieved snippet:
The secret object #1 is a "door". The secret sport is "skiing". THE GOLDMAN SACHS GROUP, INC. ANDSUBSIDIARIES
ANNUAL REPORT ONFORM 10-K FOR THE FISCAL YEAR ENDED DECEMBER31, 2023
INDEX
Form 10-K Item Number Page No.
PART I 1
Item 1
Business 1

The secret landmark is the "Colosseum". 13
The secret object #4 is a "mirror". ANNUAL REPORT 2023 LETTER TO SHAREHOLDERS
David Solomon
Chairman
and
our
...

Question: What is the secret tool in the document?
Expected: scissors
Needle: The secret tool is "scissors".
Retrieved snippet:
24 Goldman Sachs 2023 Form 10-K
The secret instrument is a "drum". Anti-Money Laundering and Anti-Bribery Rules and
Regulations
The U.S. Bank Secrecy Act, as amended (BSA), including by
y approximately 130 percent (compared to a
the USA PATRIOT Act of 2001, contains anti-money
peer average of approximately 60 perce...

Question: What is the secret landmark in the document?
Expected: Colosseum
Needle: The secret landmark is the "Colosseum".
Retrieved snippet:
"Built upon our core values of partnership,
client service, integrity and excellence,
our culture is what defines us, it is our
identity and it is at the heart of our
commercial success."
David Solomon
The secret landmark is the "Colosseum". 13
The secret object #4 is a "mirror". ANNUAL REPORT 2023 LETTER TO SHAREHOLDERS
David Solomon
Chairman