```
!pip uninstall -y faiss faiss-gpu faiss-cpu
!pip install -q faiss-cpu s3fs pyarrow polars sentence-transformers
```

```
WARNING: Skipping faiss as it is not installed.
WARNING: Skipping faiss-gpu as it is not installed.
Found existing installation: faiss-cpu 1.13.2
Uninstalling faiss-cpu-1.13.2:
  Successfully uninstalled faiss-cpu-1.13.2
```

```python
import os, gc, time, math
from typing import List, Dict

import numpy as np
import polars as pl
import pyarrow as pa
import pyarrow.dataset as ds
import pyarrow.parquet as pq
import s3fs

import torch
import faiss
from sentence_transformers import SentenceTransformer, CrossEncoder
```

```
2026-01-30 15:42:23.656595: I tensorflow/core/util/port.cc:153] oneDNN
custom operations are on. You may see slightly different numerical
results due to floating-point round-off errors from different
computation orders. To turn them off, set the environment variable
`TF_ENABLE_ONEDNN_OPTS=0`.
2026-01-30 15:42:23.669875: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to
register cuFFT factory: Attempting to register factory for plugin
cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
E0000 00:00:1769787743.687489    9773 cuda_dnn.cc:8310] Unable to
register cuDNN factory: Attempting to register factory for plugin
cuDNN when one has already been registered
E0000 00:00:1769787743.692969    9773 cuda_blas.cc:1418] Unable to
register cuBLAS factory: Attempting to register factory for plugin
cuBLAS when one has already been registered
2026-01-30 15:42:23.710036: I
tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow
binary is optimized to use available CPU instructions in performance-
critical operations.
To enable the following instructions: SSE4.1 SSE4.2 AVX AVX2 AVX512F
AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the
appropriate compiler flags.
```

```python
S3_BASE = "news-search-relevancy-apsouth1-dev/news-relevancy-
apiceberg/db_news_relevancy/news_relevancy_iceberg_table1/data"
```

```python
WORKDIR = "/home/sagemaker-user/faiss_work"
os.makedirs(WORKDIR, exist_ok=True)

EMB_MODEL = "sentence-transformers/all-MiniLM-L12-v2"
RERANK_MODEL = "cross-encoder/ms-marco-MiniLM-L-12-v2"

DEVICE_EMB = "cuda:0" if torch.cuda.is_available() else "cpu"
DEVICE_RERANK = "cpu"

# ▯ SPEED MAGIC
MAX_SEQ_LEN = 96
ARTICLE_PREFIX_CHARS = 350
TITLE_WEIGHT = 2
ENC_BATCH = 1536
ARROW_BATCH_ROWS = 8000

TRAIN_EMB_SAMPLES = 50_000
TRAIN_MAX_BATCHES = 80

MIN_NLIST = 1024
MAX_NLIST = 8192
NPROBE = 8
CHECKPOINT_EVERY_BATCHES = 20

index_path = os.path.join(WORKDIR, "news_ivfpq.index")
docstore_path = os.path.join(WORKDIR, "docstore_kept.parquet")

faiss.omp_set_num_threads(8)

print("Listing parquet files from Iceberg...")
fs = s3fs.S3FileSystem(anon=False)
all_files = fs.glob(f"{S3_BASE}/**/*.parquet")
all_files = [f"s3://{f}" for f in all_files]

dataset = ds.dataset(all_files, format="parquet", filesystem=fs)
total_rows = dataset.count_rows()
print(f"Total rows: {total_rows:,}")
```

```
Listing parquet files from Iceberg...
Total rows: 2,690,077
```

```python
embedder = SentenceTransformer(EMB_MODEL, device=DEVICE_EMB)
embedder.max_seq_length = MAX_SEQ_LEN
embedder = embedder.half()
EMB_DIM = embedder.get_sentence_embedding_dimension()

def cleanup():
    gc.collect()
    if torch.cuda.is_available():
        torch.cuda.empty_cache()
```

```python
def format_text(title: str, article: str) -> str:
    title = (title or "").strip()
    article = (article or "").strip()

    if ARTICLE_PREFIX_CHARS and len(article) > ARTICLE_PREFIX_CHARS:
        article = article[:ARTICLE_PREFIX_CHARS]

    title_block = " ".join([title] * TITLE_WEIGHT) if title else ""
    return f"{title_block}\n\n{article}".strip()

def choose_nlist(n_rows_est: int) -> int:
    nlist = int(max(MIN_NLIST, min(MAX_NLIST,
round(math.sqrt(n_rows_est)))))
    pow2 = 2 ** int(round(math.log2(nlist)))
    return int(max(MIN_NLIST, min(MAX_NLIST, pow2)))

def collect_training_embeddings():
    print("Collecting training samples for IVF...")
    scanner = dataset.scanner(columns=["title", "article"],
batch_size=ARROW_BATCH_ROWS)

    chunks, collected = [], 0

    with torch.inference_mode():
        for b, batch in enumerate(scanner.to_batches()):
            if b >= TRAIN_MAX_BATCHES or collected >=
TRAIN_EMB_SAMPLES:
                break

            titles = batch.column("title").to_pylist()
            arts = batch.column("article").to_pylist()

            texts = [format_text(t, a) for t, a in zip(titles, arts)]
            texts = [t for t in texts if t][:TRAIN_EMB_SAMPLES -
collected]

            embs = embedder.encode(
                texts,
                batch_size=ENC_BATCH,
                convert_to_numpy=True,
                normalize_embeddings=True,
                show_progress_bar=False,
                device=DEVICE_EMB
            ).astype(np.float32)

            chunks.append(embs)
            collected += len(embs)
            print(f"[train] {collected:,} samples")
            cleanup()
    return np.vstack(chunks)

def make_ivfpq_index(nlist):
    quantizer = faiss.IndexFlatIP(EMB_DIM)
    m = 48 if EMB_DIM % 48 == 0 else 32
    index = faiss.IndexIVFPQ(quantizer, EMB_DIM, nlist, m, 8,
faiss.METRIC_INNER_PRODUCT)
    index.nprobe = NPROBE
    return index

nlist = choose_nlist(total_rows)
train_x = collect_training_embeddings()

index = make_ivfpq_index(nlist)
index.train(train_x)
del train_x
cleanup()
```

```
Collecting training samples for IVF...
[train] 7,999 samples
[train] 15,999 samples
[train] 23,998 samples
[train] 31,998 samples
[train] 39,998 samples
[train] 47,998 samples
[train] 50,000 samples
```

```
WARNING clustering 50000 points to 2048 centroids: please provide at
least 79872 training points
```

```
def build_index():
    writer = pq.ParquetWriter(
        docstore_path,
        pa.schema([
            ("row_id", pa.int64()),
            ("title", pa.string()),
            ("article", pa.string()),
        ]),
        compression="zstd"
    )

    scanner = dataset.scanner(columns=["title", "article"],
batch_size=ARROW_BATCH_ROWS)
    id_index = faiss.IndexIDMap2(index)

    global_row = 0
    start = time.time()

    with torch.inference_mode():
        for b, batch in enumerate(scanner.to_batches()):
            titles = batch.column("title").to_pylist()
            arts = batch.column("article").to_pylist()

            texts, row_ids = [], []
            kept_titles, kept_arts = [], []

            for i, (t, a) in enumerate(zip(titles, arts)):
                txt = format_text(t, a)
                if txt:
                    texts.append(txt)
                    row_ids.append(global_row + i)
                    kept_titles.append(t)
                    kept_arts.append(a)

            global_row += len(titles)

            if texts:
                embs = embedder.encode(
                    texts,
                    batch_size=ENC_BATCH,
                    convert_to_numpy=True,
                    normalize_embeddings=True,
                    show_progress_bar=False,
                    device=DEVICE_EMB
                ).astype(np.float32)

                row_ids = np.asarray(row_ids, dtype=np.int64)
                id_index.add_with_ids(embs, row_ids)

                writer.write_table(pa.table({
                    "row_id": row_ids,
                    "title": kept_titles,
                    "article": kept_arts
                }))

            cleanup()

            if (b+1) % CHECKPOINT_EVERY_BATCHES == 0:
                faiss.write_index(id_index, index_path)
                print(f"[build] batch {b+1} |
ntotal={id_index.ntotal:,}")

    writer.close()
    faiss.write_index(id_index, index_path)
    print("Build complete in", (time.time()-start)/60, "minutes")

build_index()
```

```
[build] batch 20 | ntotal=147,373
[build] batch 40 | ntotal=294,708
```

```
[build] batch 60 | ntotal=444,286
[build] batch 80 | ntotal=598,473
[build] batch 100 | ntotal=742,547
[build] batch 120 | ntotal=890,068
[build] batch 140 | ntotal=1,041,145
[build] batch 160 | ntotal=1,191,743
[build] batch 180 | ntotal=1,331,168
[build] batch 200 | ntotal=1,474,200
[build] batch 220 | ntotal=1,619,397
[build] batch 240 | ntotal=1,763,223
[build] batch 260 | ntotal=1,917,532
[build] batch 280 | ntotal=2,069,940
[build] batch 300 | ntotal=2,207,825
[build] batch 320 | ntotal=2,357,762
[build] batch 340 | ntotal=2,507,516
[build] batch 360 | ntotal=2,655,790
Build complete in 24.086429623762765 minutes

doc_lazy = pl.scan_parquet(docstore_path)
cpu_index = faiss.read_index(index_path)
reranker = CrossEncoder(RERANK_MODEL, device=DEVICE_RERANK)

def fetch_rows(row_ids):
    df = (
        doc_lazy
        .filter(pl.col("row_id").is_in(row_ids))
        .collect(streaming=True)
    )
    return df.to_dicts()

def search(query, k=5):
    q = embedder.encode([query],
normalize_embeddings=True).astype(np.float32)
    D, I = cpu_index.search(q, 200)

    ids = [int(i) for i in I[0] if i != -1]
    rows = fetch_rows(ids)

    pairs = [(query, f"{r['title']}\n\n{r['article'][:1000]}") for r
in rows]
    scores = reranker.predict(pairs)

    top = np.argsort(-scores)[:k]
    return [rows[i] for i in top]

results = search("economic impact of elections")
for r in results:
    print("\nTITLE:", r["title"])
```

```
/tmp/ipykernel_9773/2877209531.py:5: DeprecationWarning: the
`streaming` parameter was deprecated in 1.25.0; use `engine` instead.
  .collect(streaming=True)


TITLE: BOEGÇÖS CARNEY DECLINES TO ANSWER ON ECONOMIC IMPACT OF A
POTENTIAL GENERAL ELECTION

TITLE: Does the economy affect elections any more? - Can’t buy me love

TITLE: Why a B-Minus Economy May Be Causing a Turbulent Election

TITLE: UK's winter election: What's in it for markets?

TITLE: Brazil election: Markets could quickly spiral into crisis mode
```

```
results = search("economic impact of elections", k=5)

for r in results:
    print(r["row_id"], "|", r["title"])
```

```
/tmp/ipykernel_9773/2877209531.py:5: DeprecationWarning: the
`streaming` parameter was deprecated in 1.25.0; use `engine` instead.
  .collect(streaming=True)

2207720 | BOEGÇÖS CARNEY DECLINES TO ANSWER ON ECONOMIC IMPACT OF A
POTENTIAL GENERAL ELECTION
1783183 | Does the economy affect elections any more? - Can’t buy me
love
1478004 | Why a B-Minus Economy May Be Causing a Turbulent Election
296447 | UK's winter election: What's in it for markets?
2391535 | Brazil election: Markets could quickly spiral into crisis
mode
```

```
eval_queries = [
    {
        "query": "economic impact of elections",
        "relevant": [
            2207720,
            1783183,
            1478004,
            296447,
            2391535
        ]
    }
]

def evaluate(eval_queries, k=5):
    recalls, mrrs, ndcgs = [], [], []

    for item in eval_queries:
        query = item["query"]
        relevant = set(item["relevant"])

        results = search(query, k=50)
        retrieved_ids = [r["row_id"] for r in results]

        # Recall@k
        recalls.append(int(len(set(retrieved_ids[:k]) & relevant) >
0))

        # MRR
        rr = 0
        for i, rid in enumerate(retrieved_ids, 1):
            if rid in relevant:
                rr = 1 / i
                break
        mrrs.append(rr)

        # nDCG
        dcg = 0
        for i, rid in enumerate(retrieved_ids[:k], 1):
            if rid in relevant:
                dcg += 1 / np.log2(i + 1)

        idcg = sum(1 / np.log2(i + 1) for i in range(1,
min(len(relevant), k) + 1))
        ndcgs.append(dcg / idcg)

    print(f"Recall@{k}: {np.mean(recalls):.3f}")
    print(f"MRR:        {np.mean(mrrs):.3f}")
    print(f"nDCG@{k}:   {np.mean(ndcgs):.3f}")

evaluate(eval_queries, k=5)
```

```
/tmp/ipykernel_9773/2877209531.py:5: DeprecationWarning: the
`streaming` parameter was deprecated in 1.25.0; use `engine` instead.
  .collect(streaming=True)
```

```
Recall@5: 1.000
MRR:        1.000
nDCG@5:   1.000
```