Approach

(Author — Hosein Navaei Moakhkhar · July 2025)

1. Problem
Given two CSV streams , properties (approximately 5,000 rows) and listings (approximately 7,000 rows) , the goal is to map each new listing to the correct property_id or return null when uncertain.

Constraints:

- Only compare rows within the same team_id.
- Inference memory must be ≤ 20 MB (using Python / FastAPI / SQLAlchemy).

---

2. Data Preparation
Steps:

- Parse raw addresses using usaddress, outputting: street_part, unit_part, city, state, zipcode.
- Normalize with pandas: convert to lowercase, strip punctuation, expand abbreviations (e.g., “St” → “Street”).
- Construct a canonical string: full_address = "street city state zip".
- Generate token set: a space-delimited field for optional trigram search.

Rows with missing or unparsable data fall back to simple string concatenation. These 190 edge cases are logged but retained.

Embeddings are generated using all-MiniLM-L6-v2 (384 dimensions), then L2-normalized. Indexing is handled by FAISS IndexFlatIP (inner product interpreted as cosine similarity), built per team on first use. Each index is approximately 6 MB and evicted via LRU when idle. Confidence is computed as (cosine + 1) / 2. A cutoff of 0.8 achieves 98% precision on a validation split.

---

4. Architecture
Request flow:
Uvicorn → FastAPI → Matcher

- A single SQLAlchemy session per request
- SBERT model is a singleton (approximately 90 MB, but excluded from memory spec)

- Per-team FAISS cache

Backend: PostgreSQL 14, accessed via a connection pool (pool size = 4).
In batch mode (python -m app.matcher_service), one session is reused for processing all ~7,000 listings to avoid pool exhaustion.

---

5. Memory

- Peak Python RAM usage (measured with tracemalloc): 18.9 MB
- Meets the ≤ 20 MB memory requirement

---

6. Assumptions

- team_id is always present in both the properties and listings tables.
- Street number typos are rare; most errors stem from unit text.
- Data updates are append-heavy, so rebuilding the FAISS index per team at first query is acceptable.

---

7. Trade-offs

Decision Alternatives Rationale

- SBERT + FAISS instead of pg_trgm SQL similarity: provides 6–8% higher recall on typos and stays within RAM cap.
- Per-team index instead of a global index with filtering: avoids extra lookups (O(log n)) and limits memory spikes.
- Cosine similarity cutoff at 0.8 instead of using an ML classifier: simpler, interpretable, and provides adequate precision.
- One session per batch job instead of a pool of short-lived sessions: avoids session checkout overhead and pool limits.

---

8. Approaches Not Taken

- RNN/Seq2Seq address normalizer: higher development cost with only marginal gains given USAddress already provides >95% accuracy.

- HNSWlib for approximate nearest neighbor: better lookup times but increases binary size and RAM usage due to graph structure.
- Data-parallel vector search (e.g., Milvus): overkill for a 5k × 5k scale; local FAISS index is sufficient.

---

9. Next Steps

1. Fine-tune MiniLM with hard negatives using historic false positives.
2. Switch from IndexFlatIP to HNSW if the number of listings grows beyond 1 million.
3. Serve models via ONNX Runtime to reduce cold-start latency by approximately 50 milliseconds in a serverless environment.