Semantic Indexing Output

Example Structure & Deliverables

DELIVERABLE TYPE

Production-Ready Vector Index

Format: FAISS index + aligned chunks + metadata + manifest

Integration: LangChain / LlamaIndex / Haystack compatible

Validation: Verified alignment and quality metrics included

This document describes the standard output structure for semantic indexing deliverables, enabling clients to integrate vector search capabilities without managing embedding infrastructure.

1. Folder Structure

Standard Deliverable Layout

```
indexed_corpus/
├── chunks.jsonl         → Text segments
├── metadata.jsonl       → Aligned identifiers
├── vectors.index        → FAISS index file
├── summary.json         → Manifest + status
└── [optional artifacts]
    ├── semantic_split_analysis.json
    └── batch_calibration.json
```

File Purposes

chunks.jsonl: One JSON object per line containing the text content and any domain prefixes applied during embedding.

metadata.jsonl: Line-aligned with chunks, contains unique IDs, source paths, and domain labels.

vectors.index: FAISS binary index file, ready for `faiss.read_index()`

summary.json: Manifest declaring status, counts, model info, and quality metrics.

2. File Specifications

chunks.jsonl Format

```
{
  "id": 0,
  "text": "passage: The semantic...",
  "source": "doc_001.pdf",
  "domain": "legal"
}
```

metadata.jsonl Format

```
{
  "id": 0,
  "source_path": "/docs/doc_001.pdf",
  "chunk_index": 0,
  "domain_prefix": "LEGAL_REGULATORY"
}
```

summary.json Format

```
{
  "status": "VERIFIED",
  "vector_count": 91229,
  "chunk_count": 91229,
  "dimensions": 1024,
  "embedding_model": "intfloat/e5-large-v2",
  "index_type": "IndexFlatIP",
  "quality": {
    "self_contained": 0.996,
    "mid_word_breaks": 0.004
  }
}
```

3. Quality Guarantees

1:1:1
ALIGNMENT RATIO

100%
INDEX LOADS

Zero
NULL VECTORS

Verification Checks

Check	Requirement
Vector-Chunk Alignment	Exact 1:1 match
Metadata Alignment	Line-for-line with chunks
Index Integrity	Loads without error
Null Detection	Zero null embeddings
Dimension Consistency	All vectors same dim

Quality Thresholds

Metric	Target
Self-Contained Chunks	≥ 90%
Mid-Semantic Breaks	≤ 10%
Mid-Word Breaks	≤ 1%
Duplicate Rate	< 0.1%

4. Framework Integration

LangChain

```
from langchain.vectorstores import FAISS

vectorstore = FAISS.load_local(
    "indexed_corpus",
    embeddings,
    index_name="vectors"
)
```

LlamaIndex

```
from llama_index import VectorStoreIndex

index = VectorStoreIndex.from_vector_store(
    faiss_store
)
```

Direct FAISS

```
import faiss
import json

index = faiss.read_index("vectors.index")

with open("chunks.jsonl") as f:
    chunks = [json.loads(l) for l in f]

# Query
D, I = index.search(query_vector, k=5)
results = [chunks[i] for i in I[0]]
```

Note: Query-time embedding must use the same model and prefix conventions documented in summary.json.

5. Deliverable Summary

WHAT YOU RECEIVE

Complete, Validated Vector Index

Included

- FAISS index file ready for production
- Aligned chunk and metadata files
- Quality metrics and validation status
- Framework integration documentation
- Embedding model and prefix specifications

Client Benefits

- No GPU infrastructure required
- No embedding pipeline maintenance
- Immediate RAG integration capability
- Independent verification possible
- Documented quality thresholds met

Output specifications and quality thresholds are engagement-specific. This document shows example structure only.