LilVect - User Manual

LilVect - small, local vector DB (memmap + sqlite) with journaling and an optional HTTP API.

1. Installation:

```
pip install lilvect
```

2. Basic Usage:

```
from vectorlite import VectorLiteClient
import numpy as np

vl = VectorLiteClient(path="./vl_db", dim=4)
vl.upsert("a", np.array([1, 0, 0, 0], dtype="float32"), {"name": "vec-a"})
vl.upsert_batch([("b", np.array([0, 1, 0, 0], dtype="float32"), {"name": "vec-b"})])
print(vl.search(np.array([1, 0, 0, 0], dtype="float32"), k=2))
```

3. HTTP Server:

```
VECTORLITE_DB=./vl_db VECTORLITE_DIM=4 uvicorn vectorlite.vectorlite.server:app --port 8000

curl -X POST "http://127.0.0.1:8000/upsert" -H "Content-Type: application/json" -d '{"id":"alpha","vector":[1,0,0,0],"metadata":{"name":"alpha"}}'
```

4. File Structure:

- vectors.dat (memmap file)
- metadata.db (sqlite DB)
- journal files (.json or .bin)
- write.lock

5. Key Methods:

- upsert(id, vector, metadata)

- upsert_batch(list_of_items)
- get(id)
- delete(id)
- search(query, k=10, metric="cosine")
- storage.compact()

6. Troubleshooting:
- UNIQUE constraint: use upsert, not add.
- ImportError: use vectorlite.vectorlite.server for uvicorn.
- Journal recovery messages are normal.

7. Example CLI:
python -m vectorlite.maintenance ./vl_db

8. Advanced:
- Journaling ensures crash recovery.
- Single-writer safe (file lock).
- Compaction reclaims deleted space.
- Designed for local vector DB workloads.

Author: Safvan

License: MIT