Popescu Vlad Gabriel Veridion Challenge #4 — Product Deduplication (Python)

Goal: consolidate duplicate rows into one enriched entry per product, maximizing information while ensuring uniqueness.

Highlights

- Robust blocking → similarity → clustering → aggregation pipeline
- Canonical GTIN-14 unifies gtin/ean/upc as a strong identity signal
- Transparent and tunable scoring; explainable connected-components clustering
- Flat src/ structure for easy navigation; CLI for quick runs; tests included

Project Structure

```
veridion_dedup_src_flat/
├─ data/
│  ├─ raw/          # input files
│  ├─ interim/      # scratch / temp
│  └─ processed/    # outputs (dedup + mapping)
├─ src/
│  ├─ utils.py          # text normalization, GTIN-14, tokenization, stable UID
│  ├─ blocking.py       # block keys: GTIN, brand+model, first 3 name tokens,
brand+model_tokens
│  ├─ similarity.py     # hybrid score (Jaccard, difflib, model tokens, brand)
│  ├─ cluster.py        # pair generation per block + connected components (DSU)
│  ├─ aggregate.py      # deterministic field aggregation, price stats, URLs, product_uid
│  └─ pipeline.py       # orchestrates deduplicate() + assign_clusters()
├─ scripts/
│  └─ run_dedup.py      # CLI: read CSV/Parquet, write outputs
└─ tests/
   ├─ test_smoke.py     # pytest smoke
   └─ test.py           # quick demo runner
```

How It Works

1. Normalize text (lowercase, remove diacritics/punctuation), tokenize.
2. Canonicalize IDs: unify gtin/ean/upc into GTIN-14 (left-pad).
3. Block to avoid O(N^2): multiple keys per row catch candidates.
4. Score pairs with a hybrid similarity (tokens + character ratio + model tokens + brand).
5. Cluster using edges for pairs ≥ threshold (default 0.80) → connected components.
6. Aggregate per cluster: best text, identifier modal, price min/median/max, union URLs/images, stable product_uid.

Blocking Keys (redundant by design)

- GTIN:<14d>
- BM:<brand>:<model>

- NAME3:<first-3-name-tokens>
- B-MTOK:<brand>:<up-to-2-model-tokens>
- Fallback: NAMEx:<up-to-5 unique sorted tokens>

Pair Similarity (0..1)

Component	Weight
Jaccard(name tokens)	0.40
difflib ratio (name↔name)	0.20
Jaccard(model tokens)	0.30
Brand exact match	0.10

Shortcut: if canonical identifiers intersect (GTIN-14 from gtin/ean/upc) ⇒ score = 1.0.

Run It

Install

```
python -m venv .venv && source .venv/bin/activate    # Windows: .venv\Scripts\activate
pip install -r requirements.txt                                  # + pip install pyarrow
(for Parquet)
```

Run (CSV/Parquet)

```
python scripts/run_dedup.py   --input
data/raw/veridion_product_deduplication_challenge.parquet   --out-dir data/processed   --
threshold 0.80
```

Parameters

Flag	Default	Description
--input	(required)	Path to CSV/Parquet
--out-dir	data/processed	Output folder
--threshold	0.80	Min score to link two rows in same cluster

Outputs

- dedup.csv (and dedup.parquet if pyarrow available) — one row per product with aggregated attributes, price stats, URLs/images, product_uid, cluster_size, original_indices.
- cluster_assignments.csv — mapping original row index → product_uid.

Quality & Tests

```
pytest -q
python tests/test.py
```

Scaling & Next Steps

- Partition by block key → parallelizable (multiprocessing / Spark).
- Swap difflib with RapidFuzz for speed and quality.
- Persist pair scores for audit and safer threshold tuning.
- Category-aware model parsing; currency normalization and outlier handling.