Contradiction Detection Across Structured Datasets

1. Defining a Contradiction Between Structured Datasets

A contradiction occurs when two structured datasets refer to the same real-world entity but contain different values for fields that should remain consistent. These contradictions may appear when data is collected from multiple operational systems, partner feeds, or analytics platforms, each with its own update cadence, data rules, or data quality maturity.

Common examples of contradictions include:

- **Value mismatch:** e.g., two different customer addresses.
- **Numeric deviation:** revenue or balance differences beyond tolerance levels.
- **Categorical disagreement:** e.g., "Gold tier" vs. "Premium tier."
- **Boolean conflicts:** e.g., "active = true" in one system and "false" in another.
- **Formatting differences:** e.g., "John Smith" vs. "Smith, John."

Contradictions do not always imply incorrect data; sometimes one dataset is simply more recent or authoritative. Therefore, contradictions must be detected, contextualized, and scored rather than treated as errors.

---

2. Detecting Inconsistencies Across Sources

Inconsistency detection happens at two primary levels:

Record-Level Detection

Records must first be matched across datasets using identifiers such as customer ID, phone number, email address, or surrogate keys. Where no direct identifier exists, probabilistic or fuzzy matching techniques may be required. Once alignment is complete, full record comparisons can reveal conflict patterns.

Field-Level Detection

After records have been linked, each field should be compared based on appropriate logic. This may include:

- Exact matching for uniquely identifying or immutable fields.
- Numeric difference thresholds for amounts, metrics, or balances.
- Similarity scoring (Levenshtein distance, cosine similarity, phonetic matching) for free text fields such as names or addresses.
- Domain rules (e.g., birthdate and legal identifiers should not change).

Every field comparison should result in a classification such as **match, near-match, anomaly, or conflict.**

---

3. Confidence Scoring Framework

Instead of applying binary judgments (right or wrong), a confidence scoring approach helps determine which value is more reliable when contradictions occur. This supports data governance, master data management (MDM), and automated anomaly handling workflows.

A confidence score may incorporate:

- **Source Reliability:** Some systems are inherently authoritative (e.g., CRM for customer identity, billing for financials).
- **Recency:** More recent timestamps often represent more accurate information.
- **Conflict Frequency:** Fields repeatedly contradicted across systems should receive lower trust weighting.
- **Agreement Across Sources:** If multiple systems agree, the consensus value receives higher confidence.

A simplified scoring model may look like:

confidence_score = (source_weight × 0.5) + (recency_score × 0.3) + (consensus_score × 0.2)

Higher-scoring values can be selected as canonical, while low-confidence cases may be routed for human validation.

---

4. Validation Summary Output

After contradictions are detected and scored, the final output should be structured in a way that enables reporting, automated system updates, exception handling, and auditing.

A recommended output structure includes:

- Entity identifier (e.g., customer_id)
- Field name
- Value from Source A
- Value from Source B
- Comparison result (match / similar / conflict)
- Final confidence score

This output can feed dashboards, anomaly detection systems, governance processes, or automated golden record generation.