Hybrid LCA-MRIO Data Processing & Sector Mapping: Detailed Workflow

---

1. Data Sources and Goals

Objective:\ To hybridize a Multi-Regional Input-Output (MRIO) model with micro-level Life Cycle Assessment (LCA) data for food products, ensuring consistent units and accurate sector mapping.

Data Used:

- **Open Food Facts / Products Data**: Contains `product_name`, `ingredients_text`, and initial sector assignment (`MRIO_CPC_description`).
- **LCA Data (Clark et al. 2022)**: LCA greenhouse gas emissions (kg CO_2-eq per kg food).
- **FAOSTAT Prices**: Average commodity prices (USD/tonne, converted to EUR/kg).
- **MRIO Sector List**: Numerical sector codes and descriptions from EXIOBASE or a similar database.

---

2. Data Preparation & Cleaning

A. Main Ingredient Extraction

- Extracted the main ingredient from each product's `ingredients_text` using:

```
def get_main_ingredient(row):
    ingredients = str(row['ingredients_text']).split(',')
    return ingredients[0].strip().lower() if ingredients else None

products_df['main_ingredient'] = products_df.apply(get_main_ingredient,
axis=1)
```

B. LCA and Price Data Normalization

LCA Data:

- Lowercased and stripped all `Entity` (food group) names for matching.
- Created a dictionary:

```
lca_map = dict(zip(lca_df['Entity_lower'], lca_df['ghg_kg']))
```

Price Data:

- Converted prices from USD/tonne to EUR/kg:

```
faostat_df['price_eur_per_kg'] = faostat_df['Value'] / 1000 * usd_to_eur
```

- Built price mapping:

```
ingredient_prices = dict(zip(faostat_df['Item'].str.lower().str.strip(),
faostat_df['price_eur_per_kg']))
```

---

3. Fuzzy Matching for Ingredient → LCA Group

To deal with spelling and naming differences, we used fuzzy string matching:

```
from rapidfuzz import process

def fuzzy_lca(food, lca_entities, min_score=85):
    matches = process.extract(food, lca_entities, limit=1,
score_cutoff=min_score)
    return matches[0][0] if matches else None

lca_entities = lca_df['Entity_lower'].unique().tolist()
products_df['lca_entity_fuzzy'] = products_df['main_ingredient'].apply(lambda
x: fuzzy_lca(str(x), lca_entities))
products_df['lca_ghg_kg'] = products_df['lca_entity_fuzzy'].map(lca_map)
```

---

4. Unit Conversion: LCA per kg → LCA per EUR

Why?

The MRIO model requires environmental coefficients as **kg CO_2-eq per EUR of output**, but LCA data is per kg of product.

How?

For each product:

```
products_df['main_ingredient_price_eur_per_kg'] =
products_df['main_ingredient'].map(ingredient_prices)
products_df['lca_ghg_per_eur'] = products_df['lca_ghg_kg'] /
products_df['main_ingredient_price_eur_per_kg']
```

- This computes the **GHG impact per unit of economic output** for each product, consistent with MRIO conventions.

---

5. Sector Mapping: Fuzzy Matching Product Sectors to MRIO Sectors

Challenge:

Product sectors (`MRIO_CPC_description`) and MRIO sector descriptions do not match exactly in wording.

Solution: Fuzzy Sector Matching

```
sector_desc_to_idx = {k.strip().lower(): str(v) for k, v in
zip(desc_df['CPC_description'], desc_df['code'])}
sector_keys = list(sector_desc_to_idx.keys())

def fuzzy_match_sector(desc, candidates=sector_keys, threshold=75):
    if pd.isnull(desc):
        return None
    match = process.extractOne(desc, candidates, score_cutoff=threshold)
    return match[0] if match else None

products_df['MRIO_CPC_description_clean'] =
products_df['MRIO_CPC_description'].str.strip().str.lower()
products_df['sector_description_matched'] =
products_df['MRIO_CPC_description_clean'].apply(fuzzy_match_sector)
products_df['sector_index_str'] =
products_df['sector_description_matched'].map(sector_desc_to_idx)
products_df['sector_index'] =
products_df['sector_index_str'].astype(float).astype('Int64')
```

---

6. Aggregating and Updating the Q Matrix

A. Aggregate Micro Q by Sector

```
sector_q_idx = (
    products_df
    .dropna(subset=['lca_ghg_per_eur', 'sector_index'])
    .groupby('sector_index')['lca_ghg_per_eur']
    .mean()
    .to_dict()
)
```

B. Update Q Matrix in MRIO Model

```
import numpy as np
```

```python
def update_Q_with_micro(Q_matrix, sector_to_new_coeff, num_regions,
num_sectors, env_idx=0):
    Q_new = Q_matrix.copy()
    for sector, new_val in sector_to_new_coeff.items():
        for r in range(num_regions):
            idx = r * num_sectors + sector
            Q_new[env_idx, idx] = new_val
    return Q_new

Q_cpu = model.Q_matrix
Q_updated = update_Q_with_micro(
    Q_cpu,
    sector_q_idx,
    num_regions=model.num_regions,
    num_sectors=model.num_sectors,
    env_idx=0,
)
model.Q_matrix = Q_updated
```

---

7. Results and Notes

- All units are now **kg CO_2-eq per EUR** for the Q matrix.
- Fuzzy matching ensures maximal coverage of products to LCA and MRIO sectors.
- The pipeline is fully reproducible and can be adapted for other environmental indicators.

End of Report.