CPER Environmental Product Declarations (EPD) Search Workflow

April 12, 2025

Abstract

This report outlines the complete end-to-end pipeline for processing, indexing, and querying Environmental Product Declarations (EPDs) based on the "Global Warming Potential" impact category. The solution involves preprocessing JSON data, constructing a searchable FAISS index using bi-encoder sentence embeddings, and refining query matches using a cross-encoder for semantic re-ranking. We use FastAPI to deploy the model as an interactive search interface.

1 Introduction

Environmental Product Declarations (EPDs) provide quantified environmental data for products under standardized conditions. In CPER, we focus on retrieving semantically similar EPDs given a user query to extract meaningful impact data such as A1–A5 lifecycle indicators.

The system uses both **bi-encoder** models for fast vector similarity and **cross-encoder** models for accurate re-ranking, creating a powerful hybrid search engine.

2 Data Processing

Raw EPD data is provided as JSON files. We begin by parsing each file and applying strict validation to ensure quality. Each entry must meet the following criteria:

- The `epd_impacts` must include the "Global Warming" impact category.
- All relevant impact values (A1, A2, A3, A4, A5, A1_A3_total) must not all be zero or null.
- The `product_names`, `product_ids`, and `product_description` must not be placeholders.

Valid entries are normalized and cleaned, and only the relevant impact category is preserved. The output is saved as `processed_json_data.json`.

3 Text Representation

To facilitate semantic search, each valid EPD is converted into a weighted text representation:

```
combined_text = (product_name * 3) + (product_id * 2) + product_description
```

This boosts the importance of the name and ID in similarity calculations.

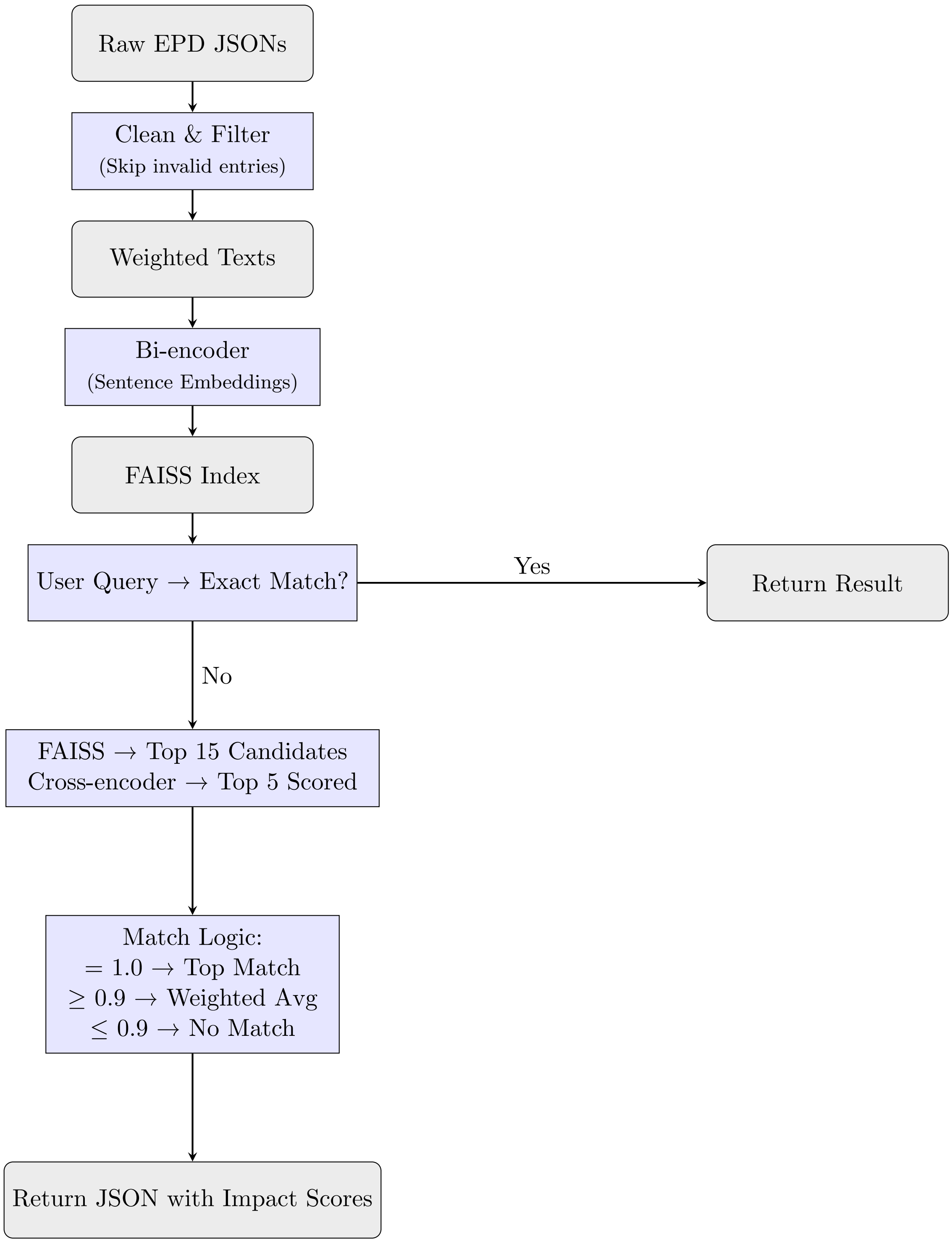

Figure 1: Workflow of the CPER EPD semantic search system. The user query is processed through exact match, FAISS-based filtering, and cross-encoder re-ranking to determine impact data.

4 Sentence Embedding and FAISS Indexing

We use the `all-MiniLM-L6-v2` model from `sentence-transformers` to encode each EPD into a dense vector embedding. These embeddings are normalized and stored in a FAISS `IndexFlatIP`, enabling fast inner-product searches (which approximate cosine similarity).

The following files are generated:

- `revised_faiss_index.index`: FAISS index
- `revised_json_mapping.pkl`: JSON mapping (EPDs)
- `embedding_model_name.txt`: Embedding model reference

5 Hybrid Search Pipeline

The user query flows through the following steps:

1. **Exact Match Check**: Compares query directly with product names, IDs, and descriptions.
2. **FAISS Search**: Embeds the query using the same bi-encoder and retrieves top-K candidate EPDs.
3. **Cross-Encoder Re-ranking**: Each candidate is paired with the query and passed to a `cross-encoder/ms-marco-MiniLM-L-6-v2` model for a more accurate similarity score.

Scoring Logic

- **Similarity = 1.0**: Return top match.
- **Similarity $\geq$ 0.9**: Compute weighted average from top matches.
- **Similarity < 0.9**: Return "No match found."

6 Impact Score Aggregation

If multiple similar EPDs are retrieved (similarity ≥ 0.9), we compute a weighted average for A1–A5 values using cosine similarity as weights.

$$A_i = \frac{\sum_k s_k \cdot A_i^{(k)}}{\sum_k s_k} \tag{1}$$

Where:

- s_k = similarity score of the k-th matched product
- $A_i^{(k)}$ = impact value of category A_i for the k-th product

7 FastAPI Deployment

The system is exposed via FastAPI, with endpoints:

- `GET /` – Returns an HTML search interface.
- `POST /search` – Accepts a JSON query and returns relevant products and impacts.

The server uses models and FAISS index preloaded into memory for performance.

8 API Response Schema

The response returned from the search API follows a structured JSON schema. Below is an example of a typical response using the `weighted_average` scoring method.

JSON Response Example


```
{
  "message": "High similarity, using weighted average.",
  "score_type": "weighted_average",
  "similarity_scores": [
    0.9993903636932373,
    0.9993659853935242,
    0.9987083673477173
  ],
  "impact": {
    "unit": "kg CO2 eq.",
    "A_values": {
      "A1": 1303.618758398065,
      "A2": 74.34903687802168,
      "A3": 365.74470573780053,
      "A1_A3_total": 2544.353863270565,
      "A4": 0,
      "A5": 0
    }
  },
  "matched_products": [
    {
      "product_info": {
        "product_names": ["Framery O"],
        "product_description": ["Framery O pod is a sound-isolated..."],
        "product_ids": [],
        "A_values": {
          "A1": 1240, "A2": 0, "A3": 97,
          "A1_A3_total": 1337, "A4": 0, "A5": 0
        }
      },
      "similarity": 0.9993903636932373
    },
```

```
    {
      "product_info": {
        "product_names": ["Framery 2Q"],
        "product_description": ["Framery 2Q is a sound-isolated..."],
        "product_ids": [],
        "A_values": {
          "A1": 2670, "A2": 223, "A3": 1000,
          "A1_A3_total": 3890, "A4": 0, "A5": 0
        }
      },
      "similarity": 0.9993659853935242
    },
    {
      "product_info": {
        "product_names": ["Framery Q"],
        "product_description": ["Framery Q pod is a sound-isolated..."],
        "product_ids": [],
        "A_values": {
          "A1": null, "A2": null, "A3": null,
          "A1_A3_total": 2406, "A4": null, "A5": null
        }
      },
      "similarity": 0.9987083673477173
    }
  ]
}
```


Fields Explained

- `message` – Status or logic explanation.
- `score_type` – One of `top_match`, `weighted_average`, or `None`.
- `similarity_scores` – Cosine or cross-encoder similarity values.
- `impact` – Computed or exact A1–A5 values and unit.
- `matched_products` – Array of top matching product metadata with scores.

9 Code and Data Availability

https://github.com/Volition-labs/CPER

10 Conclusion

This hybrid search system efficiently retrieves semantically relevant EPDs using a combination of FAISS indexing and transformer-based re-ranking. The integration of threshold-based logic ensures high precision in selecting top matches or computing weighted impact estimates.