Multilingual Room Matching with Fuzzy Logic and XGBoost

Ryoji – Room Match ML API Project

April 2025

Contents

1 Introduction

This project builds a multilingual machine learning API for matching hotel room listings, inspired by Cupid's Room Match API. The system accepts POST requests with structured room data from both suppliers and a reference catalog, and returns probabilistic room match predictions. It supports mixed-language inputs (e.g., English, Arabic, Korean) and uses fuzzy logic, language detection, and machine learning classification.

2 Project Structure and API Setup

- `Room_Match/` (project root)
  - `README.md` *Instructions and architecture overview*
  - `requirements.txt` *Dependencies*
  - `app.py` *Flask server for POST API*
  - `matcher.py` *Core logic: fuzzy matching, ML inference*
  - `models/` *Includes:*
    - `model.pkl` (XGBoost model)
    - `lid.176.bin` (fastText language model)
  - `sample_request.json` *Example POST input*
  - `test_post.py` *Simple test script*
  - `notebooks/room_match_dev.ipynb` *EDA and training*
  - `report.pdf` *Technical report summarizing the matching system and evaluation results*

Running the API:

1. `pip install -r requirements.txt`
2. `FLASK_APP=app.py flask run --host=0.0.0.0 --port=5050`
3. Send a test request:

```
curl -X POST http://127.0.0.1:5050/room_match \
  -H 'Content-Type: application/json' \
  -d @sample_request.json
```

4. Or run `python test_post.py`

3 Methodology

3.1 Input Format

The input to the API is a JSON object with supplier and reference rooms:

```
{
  "inputCatalog": [
    {
      "supplierId": "nuitee",
      "supplierRoomInfo": [
        {"supplierRoomId": "2", "supplierRoomName": "Classic Room - Olympic Queen Bed - ROOM ONLY"}
      ]
    }
  ],
  "referenceCatalog": [
    {
      "propertyId": "5122906",
      "propertyName": "Pestana Park Avenue",
      "referenceRoomInfo": [
        {"roomId": "512290602", "roomName": "Classic Room"},
        {"roomId": "512290608", "roomName": "Classic Room - Disability Access"}
      ]
    }
  ]
}
```


3.2 Candidate Matching Strategy

1. **ID Filtering:**
   - Supplier IDs are checked against reference `room_id`, `lp_id`, `core_room_id`, etc.
2. **Fuzzy Matching:**
   - Normalize strings (lowercase, remove accents/punctuation)
   - Compute similarity using `rapidfuzz.partial_ratio`
3. **Language Detection:**
   - Uses `fastText` model to annotate room names for language context
4. **Feature Extraction:**
   - `lp_id_match`, `hotel_id_match`, `room_id_match`, `fuzzy_score`

3.3 Model Training

- Label = 1 if fuzzy score ≥ 0.85 AND ID match
- Model: XGBoost classifier
- Tuning: Optuna
- Metrics: F1, AUC, Confusion Matrix

3.4 Multilingual Handling

- `fastText` supports 100+ languages.
- Can detect Arabic, Korean, Japanese, etc. — but only the dominant language.
- Mixed-language strings may produce partial results.
- Example: `Deluxe Room (デラックスルーム)` may be detected as Japanese or English depending on structure.

Limitation: `fastText` cannot detect or translate multiple languages in one string. It returns only the dominant language.

Recommendation: Use `SentenceTransformer (MiniLM-L12-v2)` with GPU for better cross-lingual semantic understanding.

4 Results

- **F1-score:** 99.6%
- **ROC AUC:** High
- **Confusion Matrix:** Few false positives/negatives

5 Sample Output


```
{
  "supplierRoomId": "2",
  "supplierRoomName": "Classic Room - Olympic Queen Bed
  "refRoomId": "512290602",
  "refRoomName": "Classic Room",
  "fuzzy_score": 1.0,
  "match_score": 0.9991,
  "lang_supplier": "en",
```

```
  "lang_ref": "en"
}
```

6 Figures

Figure 1: Confusion Matrix

Figure 2: ROC AUC Curve

Figure 3: XGBoost Feature Importance

7 Limitations and Future Work

- Only one supplier — needs extension to multiple.
- Current model uses only name-based features.
- Future versions should add:
  - Room view, floor, amenities
  - Descriptions and full metadata

7.1 Deployment Notes

- Docker for reproducibility
- CI/CD with Jenkins or GitHub Actions
- Hosting via FastAPI or TorchServe

7.2 LLM Potential

- Fine-tuning `MiniLM-L12-v2` with LoRA
- Use of RAG + embeddings for richer room description grounding
- Large LLMs for summarization and inference