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 Room Matching Strategy

To develop the backend ML model, I first loaded and explored the datasets:

```
df_rooms: pdated_core_rooms.csv
df_ref: referance_rooms-1737378184366.csv
```

Exploratory Data Analysis (EDA) included inspecting schema with `df.info()`, removing records where `room_name` is `NaN`, and understanding key identifier relationships like `lp_id`, `hotel_id`, `room_id`, and `core_room_id`. The `room_id` typically acts as a foreign key while `core_room_id` reflects internal indexing.

Figures Figure 1 summarizes the ID matching counts of

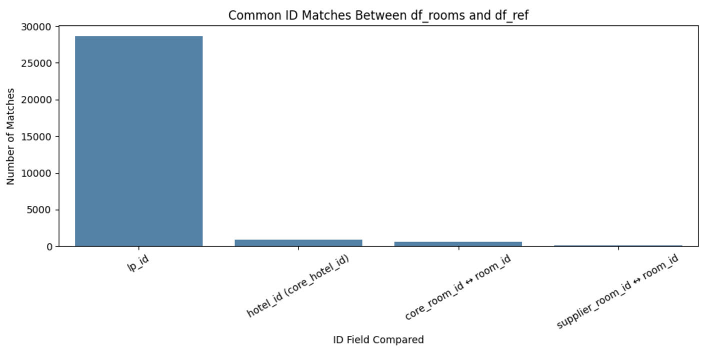

Figure 1: Common ID Matches Between core rooms and reference rooms

`lp_id`, `hotel_id`, `room_id`, and `core_room_id`.

Language detection was performed using `fastText` to annotate room names for multilingual handling.

Figure 2 displays the ID matching and fuzzy score (≥

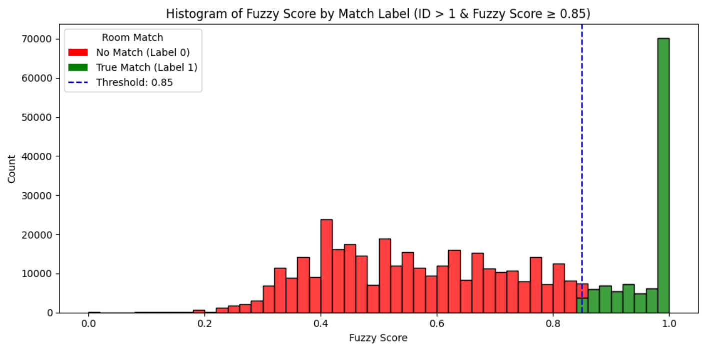

Figure 2: Histogram of Fuzzy Score by Match Label (ID>1 & Fuzzy Score ≥ 0.85)

0.85[1] and the rest of counts. It is worth noting that the fuzzy_score distribution is heavily dominated by values close to 1.0, and the resulting 0/1 label distribution is approximately 1:5. While a stricter threshold such as 1.0 could be considered.

For supervised model training:

- Matching candidates were created when `(lp_id, hotel_id, room_id)` matched more than once.
- Similarity scores were computed using fastText embedding similarity.
- The dataset was labeled and split accordingly.
- A tuned `XGBoost` classifier was trained on features including ID match booleans and text-based similarity.

[1]I tested thresholds at 0.75, 0.85, and 0.95. The label distribution ratios 01 were approximately 1.8, 3.1, and 4.7 respectively. Despite increasing imbalance, the XGBoost classifier achieved high performance across all thresholds, with F1-scores near 1.0.

Evaluation metrics:

- **Confusion Matrix** to identify true/false positives and negatives
- **F1-Score** to balance precision and recall
- **ROC Curve** for threshold-independent classification performance

Figures below show the confusion matrix and ROC curve.

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:** ~ 100 % .
- **ROC AUC:** High, shown in Fig. 3.
- **Confusion Matrix:** Small false negatives, shown in Fig. 4.

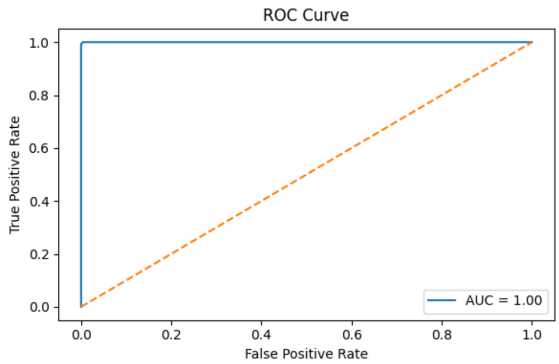

Figure 3: ROC AUC Curve

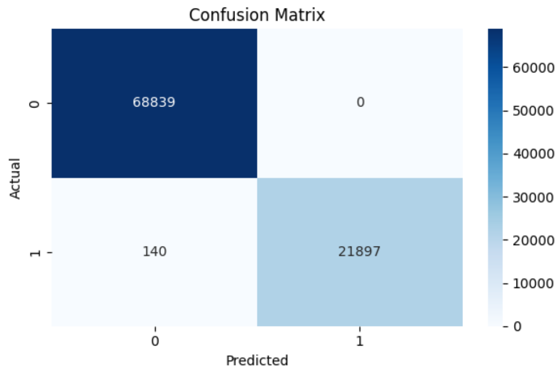

Figure 4: Confusion Matrix

5 Sample Output


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


6 Limitations and Future Work

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

6.1 Deployment Notes

- Docker for reproducibility

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

6.2 LLM Potential

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