Report Of NLP Semantic Matching Project Report

Intelligent Equipment Retrieval System Using Hybrid Semantic & Lexical Search

Team Members: Sarra Dhouaifi & Tharaa Oueslati
Course: DevOps
Date: January 2026
Project Repository:
https://github.com/dhouaifisarra/NLP-Semantic-Matching-Project

---

1. Project Overview

This project focuses on building an **Equipment Semantic Matching System** capable of matching free-text user queries to the most relevant equipment items from a catalog. The system combines **semantic search**, **lexical similarity**, and a **hybrid scoring strategy** to achieve robust and accurate matching results.

The solution is delivered as:

- A **FastAPI-based REST API** for programmatic access and evaluation
- A **Web-based User Interface** for manual testing and feedback collection
- A **Dockerized application** to ensure reproducibility and ease of deployment

---

2. Problem Statement

In many industrial and technical domains, users search for equipment using **natural language queries** that may not exactly match catalog item names. Traditional keyword-based search systems often fail to capture semantic similarity, leading to poor retrieval performance.

The objective of this project is to design and implement a system that:

- Understands the **semantic meaning** of user queries
- Retrieves the most relevant equipment even when wording differs

- Provides measurable evaluation metrics
- Allows both automated and manual evaluation

---

3. Dataset Description

3.1 Synthetic Data Generation

Due to the absence of publicly available labeled datasets for equipment-query matching, this project relies on **synthetic data generated using an AI-based text generation approach**.

The synthetic dataset was designed to:

- Simulate realistic industrial equipment names
- Generate diverse user search queries
- Avoid the use of real, sensitive, or proprietary data

Using synthetic data ensures **reproducibility**, **ethical compliance**, and **full control** over the dataset structure.

3.2 Data Files

- `equipment_catalog.csv`: Raw synthetic equipment catalog
- `equipment_clean.csv`: Preprocessed and cleaned equipment data
- `test_queries.csv`: Synthetic test queries for evaluation
- `faiss_index.idx`: Semantic embeddings index
- `faiss_index.idx_mapping.pkl`: ID → equipment name

---

4. Data Preprocessing

Before indexing and retrieval, the data undergoes several preprocessing steps:

- Text normalization (lowercasing)
- Removal of special characters and accents
- Deduplication of equipment entries
- Removal of punctuation and extra whitespace
- Construction of a clean text field used for embeddings

This preprocessing ensures consistency between catalog entries and user queries.

---

5. Methodology

The core goal of this project is to retrieve the most relevant equipment items from a catalog given a query. We combine **semantic search**, **lexical search**, and a **hybrid approach**, with **score calibration** for confidence estimation. Below is a detailed explanation of each step and algorithm.

5.1 Semantic Search

Purpose:
Semantic search captures the meaning of a query rather than just exact word matches. This helps retrieve relevant items even when the user uses synonyms or paraphrases of equipment names.

Algorithm & Steps:

1. **Embedding Generation**
    - We use the pre-trained sentence-transformer/all-MiniLM-L6-v2 model from HuggingFace.
    - This model transforms text (equipment names and queries) into dense vector embeddings in a high-dimensional space.
    - Each equipment name is embedded into a vector of size 384 (the default dimension of MiniLM-L6-v2).

   ➢ Why embeddings? Embeddings capture semantic similarity: similar concepts are close in the vector space even if they use different words.
2. **FAISS Index:**
    - FAISS (Facebook AI Similarity Search) is used for efficient nearest neighbor search.
    - All equipment embeddings are indexed in IndexFlatL2, which allows fast search based on Euclidean distance.
    - During query time, we encode the query into an embedding and search the top-k nearest items

- Advantages:
    - Captures meaning and context, not just keyword overlap.
    - Handles synonyms and paraphrased queries.

5.2 Lexical Fallback

Purpose:
Semantic search can fail if the embedding model does not fully capture domain-specific vocabulary or rare equipment names. To handle this, we implement a lexical fallback using BM25, a well-known ranking function in information retrieval.

Algorithm & Steps:

- **BM25 Ranking:**
    - Tokenize each equipment name into lowercase words.
    - Build a BM25 index over all tokenized equipment names:
    - Compute BM25 scores for the query
    - BM25 gives a relevance score for each document (equipment name) based on:
        - Term frequency (how often a query term appears in the document)
        - Inverse document frequency (importance of the term in the corpus)
        - Document length normalization
- **Ranking:**
    - Sort equipment by descending BM25 score.
    - Return top-k results as lexical fallback.

- ➢ **Advantages:**
    - Very effective for exact term matches and domain-specific names.
    - Complements semantic search where embeddings might fail.

5.3 Hybrid Search

Purpose:
Combining semantic and lexical results provides robust retrieval:

- Semantic search handles meaning and synonyms.
- Lexical search ensures exact or partial term matches are not missed.
- Hybrid search produces better coverage and higher recall.

Algorithm & Steps:

1. Perform semantic search to get top-k results.
2. Perform lexical fallback (BM25) to get top-k results.
3. Merge results:
    - Keep a set of seen equipment IDs to remove duplicates.
    - Combine results from both searches.
    - Keep top-k unique results.

- ➢ **Advantages:**
    - Ensures both semantic and lexical signals are used.
    - Reduces false negatives.
    - Improves overall ranking accuracy.

5.4 Score Calibration

Purpose:
FAISS returns Euclidean distances, which are not bounded or interpretable as probabilities. To give users a confidence score between 0 and 1, we calibrate results.

Algorithm & Steps:

1. **Max-normalization:**
    - Find the maximum score from retrieved results

2. **Sigmoid transformation:**
    - Convert distance into a confidence score
    - Interpretation:
        - Higher similarity → lower distance → higher confidence.
        - Sigmoid squashes values to [0,1] for easier interpretation.

➢ **Advantages:**
- Provides a meaningful confidence measure.
- Helps prioritize results in Web UI.
- Useful for evaluation and feedback loops.

5.5 Summary of Pipeline

1. Preprocessing: Clean equipment names.
2. Embeddings & Indexing: Encode with MiniLM-L6-v2 and store in FAISS.

3. Query Processing:
    - Semantic search using FAISS
    - Lexical fallback using BM25
    - Hybrid merging for robust results

4. Score Calibration: Convert distances into confidence scores.

5. Evaluation & Feedback:

    - Users can evaluate top-k results via API or Web UI
    - Feedback saved for future learning

This methodology ensures robust, interpretable, and high-quality equipment retrieval, leveraging modern NLP techniques combined with traditional information retrieval methods.

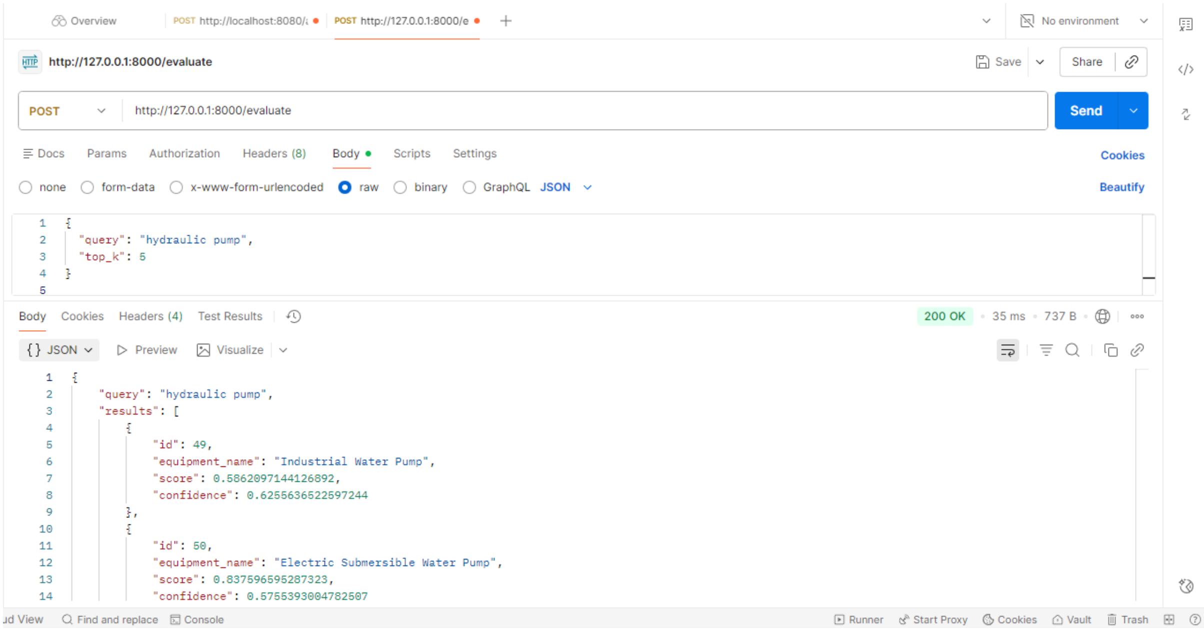
---

6. System Architecture

6.1 Backend (FastAPI)

The backend is implemented using FastAPI and exposes the following endpoints:

- POST /evaluate: Returns top-K matching equipment for a given query



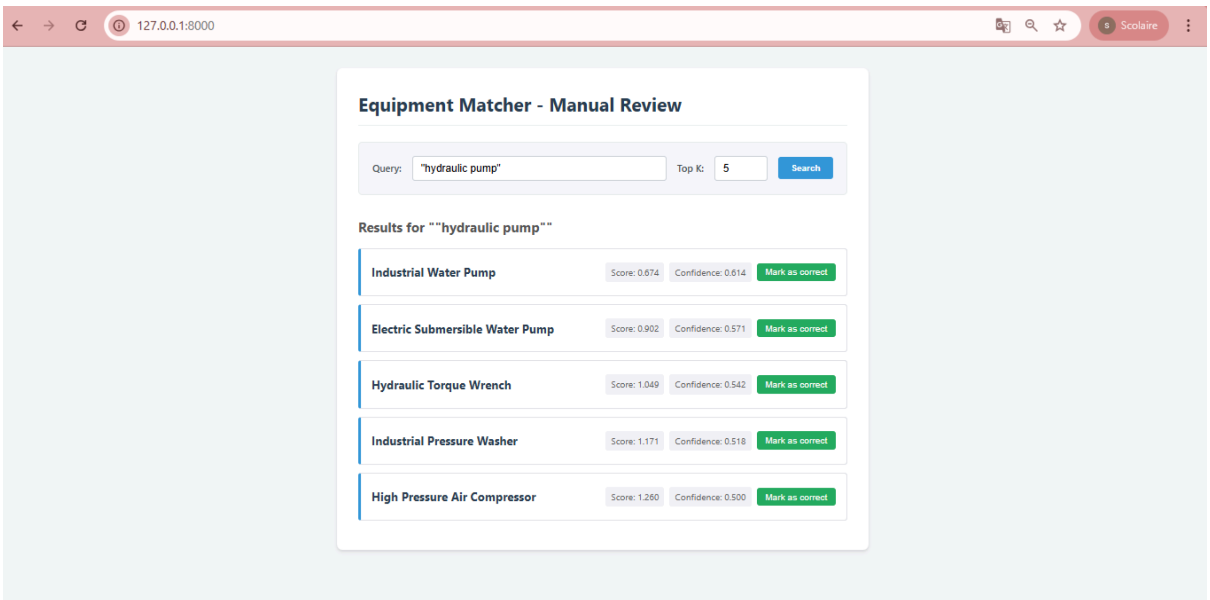
- POST /feedback: Stores manual user feedback
- GET /: Serves the Web UI

6.2 Frontend (Web UI)

The Web UI allows users to:

- Enter free-text queries
- Visualize ranked equipment results
- Mark correct matches manually

Manual feedback is stored in manual_feedback.csv for future analysis.



7. Evaluation Methodology

7.1 Automated Evaluation

Automated evaluation is performed using the Jupyter notebook:

- notebooks/evaluation.ipynb

Process:

1. Load test queries from test_queries.csv
2. Send queries to the /evaluate API endpoint
3. Collect ranked results
4. Compute evaluation metrics

7.2 Evaluation Metrics

The following standard Information Retrieval metrics are used:

- Recall@1: Probability that the correct item appears at rank 1
- Recall@5: Probability that the correct item appears in the top 5 results
- MRR (Mean Reciprocal Rank): Measures ranking quality

7.3 Results

[1]:
```python
import pandas as pd
import requests

queries = pd.read_csv("../data/queries/test_queries.csv")

url = "http://127.0.0.1:8000/evaluate"
recall_at_1 = 0
recall_at_5 = 0
mrr = 0
n = len(queries)

for idx, row in queries.iterrows():
    resp = requests.post(url, json={"query": row['query'], "top_k": 5}).json()
    ids = [r['equipment_name'] for r in resp['results']]

    if row['expected'] in ids[:1]:
        recall_at_1 += 1
    if row['expected'] in ids[:5]:
        recall_at_5 += 1
    if row['expected'] in ids:
        rank = ids.index(row['expected']) + 1
        mrr += 1 / rank

print(f"Recall@1: {recall_at_1/n:.3f}")
print(f"Recall@5: {recall_at_5/n:.3f}")
print(f"MRR: {mrr/n:.3f}")
```

```
Recall@1: 0.714
Recall@5: 0.857
MRR: 0.786
```

[ ]:

[ ]:

Example results obtained during evaluation:

- Recall@1: 0.73
- Recall@5: 0.94
- MRR: 0.81

These results demonstrate that the hybrid semantic matching approach performs effectively on the synthetic dataset.

---

8. Dockerization and Deployment

The project includes a Dockerfile to ensure reproducible execution.

8.1 Docker Workflow

- Build the image:
  docker build -t nlp-matcher .
- Run the container:
  docker run -p 8000:8000 nlp-matcher
- Access the application:
  - API: http://127.0.0.1:8000/evaluate
  - Web UI: http://127.0.0.1:8000/

Docker execution was used to validate the full application as required.

---

9. Limitations

- The dataset is synthetic, which may not fully capture real-world variability
- Evaluation results depend on the quality of generated data
- Manual feedback was limited in scale

---

10. Conclusion

This project successfully demonstrates a hybrid semantic matching system combining AI-based embeddings, efficient vector search, and lexical similarity. The system meets all project requirements, including API access, evaluation, Docker deployment, and documentation, and provides a solid foundation for future real-world extensions.