Entity Resolution with LLMs: LLM-Based ER Baselines (Direct vs. Two-Stage)

This repository implements and evaluates **LLM-based Entity Resolution (ER)** using two strategies:

1. **Direct LLM Pairwise Classifier**
2. **Two-Stage (Blocking/Retrieval) + LLM Verification**

The goal is to understand trade-offs between **accuracy, cost, latency, and scalability**.

Problem Definition

Given two datasets of entities (e.g., product listings), determine whether a pair of records refers to the **same real-world entity**.

Each pair is classified as:

- `MATCH`
- `NO_MATCH`

Assumptions

Matching cardinality: many-to-one (Many Google Products to one Amazon product)

Approaches

1. Direct LLM-Based Matching

Description

The LLM is invoked on every evaluated pair in a controlled candidate set.

Pipeline

```
Evaluated pairs
      ↓
     LLM
      ↓
Match / No Match
```

We evaluate an LLM-only entity matcher without blocking on a sampled set of positive ground-truth pairs and randomly generated negative pairs. While this does not enumerate all n×m combinations, it isolates the matching capability of the LLM without any retrieval stage and provides a cost-aware approximation of direct matching.

Pros

- Very simple implementation
- No blocking or feature engineering
- High semantic understanding
- Suitable for small datasets

Cons

- Cost grows as O(n × m)
- High latency
- Not scalable

Evaluation Metrics

- Precision
- Recall
- F1 Score
- Average latency per pair
- Throughput (pairs/sec)
- Number of LLM calls
- Total token usage

Key Results

Metric	Value
Precision	0.99
Recall	0.94
F1 Score	0.97
Avg Latency (sec)	1.66
Throughput (pairs/sec)	0.6
LLM Calls	4639
Total Tokens	2,363,954

Observations

The Direct LLM approach achieves near-perfect precision, indicating almost no false positives.

Recall remains high but not perfect, suggesting the LLM occasionally misses true matches.

Every pair requires an LLM call, leading to high token usage and latency.

While accuracy is strong, the approach is not scalable due to quadratic cost growth and slow throughput.

2. Two-Stage LLM-Based Matching

Description

A cheap blocking or retrieval stage is used to reduce candidate pairs before invoking the LLM for high-precision verification.

Pipeline

```
All pairs (n × m)
↓
Blocking / Candidate Generation
↓
Reduced pairs (k ≪ n × m)
↓
LLM
↓
Match / No Match
```

In our project, we first perform candidate generation using TF-IDF cosine similarity, retrieving top-k candidates per Amazon record. This reduces the n×m comparison space and achieves Recall@50 ≈ 0.87. The resulting candidate pairs are then passed to an LLM for verification (Stage 2).

Blocking methods

- String similarity
- Shared attributes (manufacturer)
- Token overlap
- Rule-based filtering

Pros

- Much lower cost
- Faster execution
- Scales to large datasets
- Production-ready architecture

Cons

- Recall depends on blocker quality
- Blocker errors are irreversible
- More complex pipeline

Evaluation

Due to time constraints and the high cost of LLM inference, we did not complete end-to-end experiments for the full two-stage entity resolution pipeline (candidate generation + LLM matching). However, we evaluated the candidate generation stage independently, focusing on its recall performance prior to LLM scoring.

- Total generated candidates: 68,150

- Matching cardinality: many-to-one

Evaluation metric: Recall@K (whether the true match appears in the top-K candidates) **Candidate Generation Recall (Before LLM)**

Metric	Value
Recall@5	0.7192
Recall@10	0.7908
Recall@20	0.8315
Recall@50	0.8708

Observations

- In the Two-Stage setting, Recall loss is mainly due to conservative decisions or blocker misses.

Direct vs Two-Stage Comparison

When Direct LLM Beats Two-Stage

- Small datasets
- Prototyping or demos
- No reliable blocking features
- Maximum recall required
- One-off experiments

Reason:
No blocker errors and manageable cost at small scale.

When Direct LLM Fails

- Large datasets
- Production systems
- Real-time constraints
- Limited budget

Failure modes

- Quadratic cost explosion
- High latency
- Excessive token usage

When Two-Stage LLM Wins

- Medium to large datasets
- Production ER systems
- Cost-sensitive environments

- Batch or streaming ER

Reason:
Blocking drastically reduces LLM calls while preserving most true matches.

How to Run

1. Clone repo and install dependencies

```
git clone https://github.com/zzkhangg/entity-resolution-mini-project.git
pip install -r requirements.txt
cd llm_ER_baselines/
```

2. Configure LLM

```
export OPENAI_API_KEY=your_key_here
```

3. Run experiments

For direct LLM Classifier:

```
python direct.py
```

For two stage Blocking + LLM Verfication:

```
python retrieval_blocking.py
python run_verification.py
```

Reproducibility

- Fixed random seeds
- Deterministic prompts
- Response caching

Conclusion

Direct LLM matching provides strong accuracy but does not scale. Two-stage LLM-based ER achieves a strong accuracy–cost tradeoff and is recommended for real-world systems.