Replicating OpenMEL: The Text Core Module

A Quantitative Evaluation of LLM Expansion and Global Coherence Optimization

0.1 Group Information

Student ID	Name	Undergraduate Major	Primary Role / Module
59428809	Beining Bao	Business study	Project Overview & Data Preprocessing
59467294	Jiayu Shao	Data science and big data technology	LLM-based Text Expansion
59448427	Ou Cong	Finance	BERT-based Text Local Model
59558040	WANG Liuding	Computer science	MST-based Global Optimization
59971474	Jiayin Cheng	Pure and Applied Mathematics	Experimental Evaluation & Demo and Report Integration

0.2 Contribution Statement

This project was conducted collaboratively by all five group members. Everyone participated in the initial literature review, the design of the overall OpenMEL pipeline, code debugging, and iterative revision of the final report and demo. In addition, each member took primary responsibility for one core component, as detailed below.

Beining Bao (59428809) was responsible for the project overview and data preprocessing. He collected and organized the WikiMEL text subset, performed cleaning and mention-level structuring, and produced the standardized dataset file data/mentions.json. She drafted the “Project Overview” and “Data Format” sections of the report, describing the end-to-end workflow and the interfaces between modules, and she coordinated the overall outline of the final report. She also prepared the introductory part of the demo video, presenting the motivation of OpenMEL and the global architecture of the system.

Jiayu Shao (59467294) was in charge of the LLM-based text expansion module. She designed and refined prompt templates for large language models, implemented the script src/llm_expansion.py, and generated the expanded dataset data/mentions_expanded.jsonl. He manually inspected and filtered hallucinated or low-quality outputs to improve data reliability, and he collected representative "original vs. expanded" examples. These examples, together with an explanation of the expansion logic and its benefits for entity disambiguation, formed the core of his segment in the demo video.

Ou Cong (59448427) took the lead on the BERT-based text local model. He implemented the encoder and similarity scoring logic in src/text_local_scorer.py, encoding mentions and candidate entities and computing their similarity (e.g., via cosine similarity). Using this module, he produced outputs/local_scores.jsonl for both the original and the LLM-expanded texts. He further analyzed how text expansion affects local scoring accuracy and contributed this analysis to the experimental section, as well as recording the part of the demo video that explains the local model architecture and shows example scores.

WANG Liuding (59558040) was responsible for MST-based global optimization. She designed and implemented the global entity-consistency algorithm in src/mst_global_optimizer.py, constructing the entity similarity graph, computing a minimum spanning tree, and performing greedy global optimization over local predictions. Using the local scores as input, he generated the final entity linking outputs in outputs/final_predictions.jsonl. He also curated several representative cases where locally plausible but globally inconsistent predictions were corrected by the MST procedure, and he described these "local error to global correction" examples both in the written report and in his explanatory segment of the demo video.

Jiayin Cheng (59971474) focused on experimental evaluation and demo integration. She implemented the evaluation script src/eval_metrics.py to compute accuracy and to compare two configurations: baseline plus LLM-based expansion and the full system with both LLM expansion and MST-based global optimization. She produced bar charts and tables visualizing the performance differences and drafted the "Experiments and Results" as well as the conclusion section, highlighting the contributions of LLM expansion and global consistency. In addition, she collected and aligned all video materials from the other members, and produced the final integrated demo video and report.

1. Project Overview

1.1 Background and Task Definition

Entity Linking (EL) is a fundamental task in Natural Language Processing. The goal is: given an entity mention in text, find the corresponding real-world entity in a large-scale knowledge base. Traditional approaches often rely on surface string matching or local context features, which can lead to incorrect matches when dealing with ambiguous entities (for example, “Jordan” can refer to a person or a country).

This project focuses on the OpenMEL text modality core module, reproducing a small demo under constraints of limited time and computational resources. We concentrate on validating two key approaches in the text modality.

1.2 OpenMEL Core Concepts Overview

1.2.1 LLM Query Expansion

We utilize Large Language Models (LLMs) to expand the original mention text, supplementing key information such as entity types, attributes, and context in order to improve entity disambiguation capability.

1.2.2 MST Global Coherence Optimization

We construct a similarity graph among candidate entities, extract the Maximum Spanning Tree (MST), and use global relationships to correct errors from local scoring alone, ensuring semantic coherence across the entire document or a batch of mentions.

In this demo, we select only a small subset of text for experiments, focusing on demonstrating:

- how LLM expansion helps with entity disambiguation;
- how MST-based global coherence optimization further improves performance beyond local scores.

1.3 Demo Scope and Simplifications

To complete reproduction under constraints of limited time and computational resources, we made the following simplifications to the original OpenMEL approach:

Table 1 Comparison between Full OpenMEL Approach and This Demo

Full OpenMEL Approach	This Demo (Simplified)
Multi-modal fusion (text +	**Text modality only**, excluding graph structures, GNNs, and other

graph structure)	complex components
Large-scale datasets	**Small-scale data subset**: approximately 10–30 representative ambiguous samples
Complex pre-trained models	**Lightweight models**: local model uses general pre-trained models like bert-base-uncased, without large-scale pre-training
Advanced graph optimization algorithms	**Simplified global modeling**: global coherence achieved only through MST + simple greedy adjustment

Despite the significantly reduced implementation scale, experimental results still validate the effectiveness of the two core modules:

- LLM expansion brings approximately 5.5% improvement in Accuracy;
- MST global optimization further improves by about 2.4% on this basis.

1.4 System Workflow Overview

The text data flow in this demo is illustrated in Figure 1.

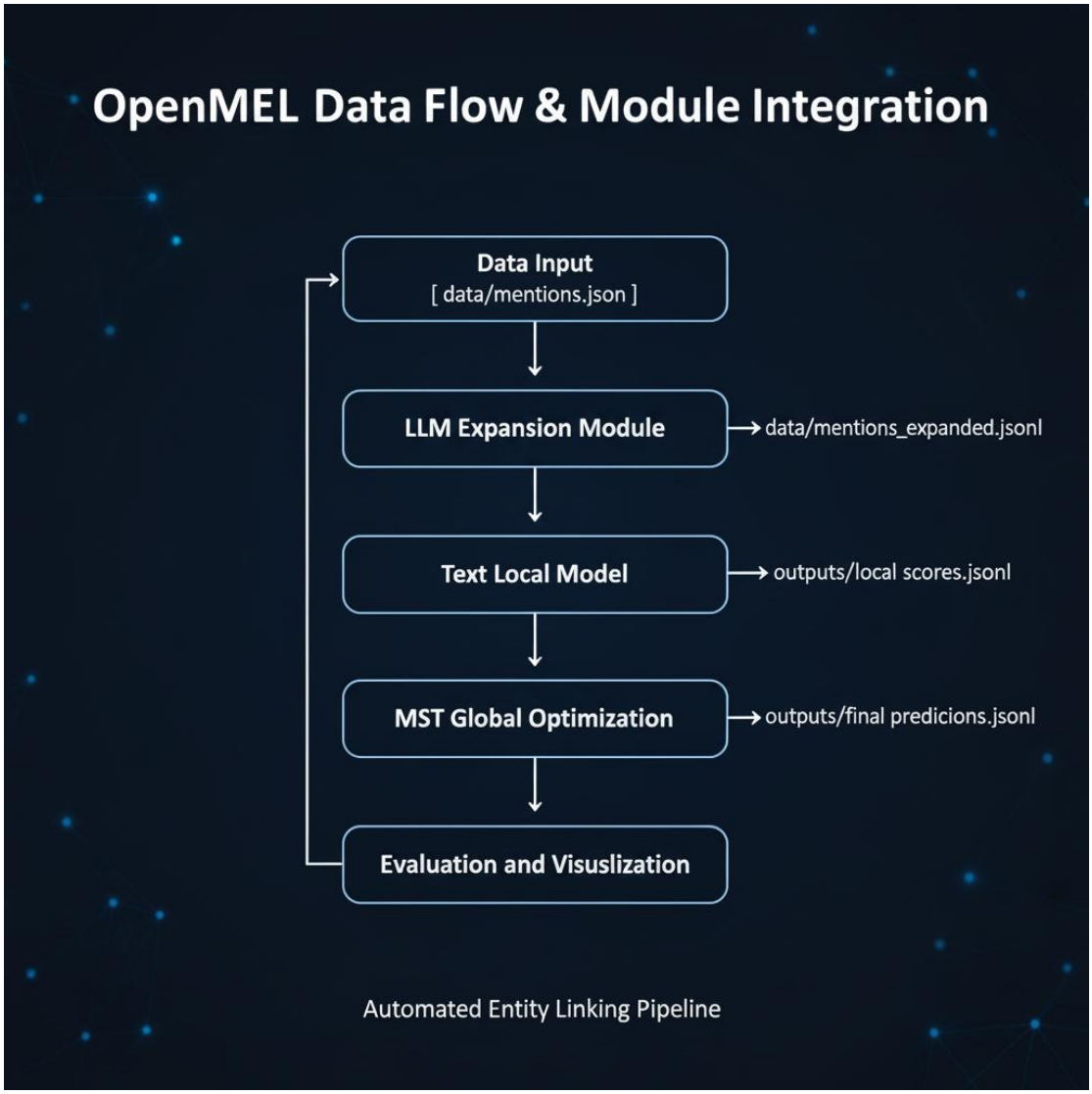


Figure 1 System workflow diagram

The detailed workflow is as follows:

1.4.1 Original Mention Text

Select a small portion of ambiguous text from datasets like WikiMEL, and format it as data/mentions.json.

1.4.2 LLM Expansion

Read the original text from mentions.json, use pre-designed prompts to call LLMs for expanding each mention, and output data/mentions_expanded.jsonl.

1.4.3 Text Local Scoring Model

Use BERT as the text encoder, concatenate mention_text with the LLM-expanded text for encoding, then compute cosine similarity with candidate entity description vectors, generating outputs/local_scores.jsonl.

1.4.4 MST Global Coherence Optimization

Treat candidate entities as graph nodes, with edge weights determined by entity text similarity; construct a maximum spanning tree and perform greedy adjustment combined with local scores, outputting outputs/final_predictions.jsonl.

1.4.5 Result Evaluation and Visualization

Compare Accuracy@1 metrics across different approaches (Local only, LLM+Local, LLM+Local+MST), and present results with tables and bar charts.

2 Technical Design

2.1 Adapted Framework Architecture

While the original OpenMEL framework operates on multimodal data (text and images), this project implements a streamlined Text-Core adaptation designed for resource-constrained environments. We preserve the theoretical foundation described in Section 1.2 but re-engineer the architecture to focus exclusively on textual reasoning.

2.1.1 From Multimodal to Text-Only Execution

Unlike the original architecture which requires complex adaptive visual processing (as mentioned in [Source: OpenMEL Paper]), our adapted framework isolates the Noise-Free Expanded Queries (NFEQ) module. We strictly utilize the LLM's generative capabilities to compensate for the missing visual modality, transforming the task from Multimodal Entity Linking (MEL) to Enhanced Text Entity Linking.

2.1.2 Lightweight Local-Global Optimization

To adapt the global coherence mechanism for a simplified demo:

- **Encoder Substitution:** We replace the heavy CLIP multimodal encoder with a lightweight BERT (bert-base-uncased) model. This reduces computational overhead while maintaining semantic sensitivity for textual data.
- **Simplified Graph Topology:** Instead of constructing the full "Tree Cover" structure with bounded nodes, we implement a weighted similarity graph. The Maximum Spanning Tree (MST) algorithm is applied directly to this text-similarity graph to resolve conflicts between local BERT scores and global semantic consistency.

2.2 Data and Interface Specifications

2.2.1 Data Source and Subset Construction Strategy

This demo uses the WikiMEL dataset as the data source, with the following strategy for constructing the experimental subset:

- **Sample Selection:** filter approximately 10–30 representative ambiguous samples from the original dataset.
- **Balance Consideration:** ensure a reasonable positive/negative sample ratio to avoid extreme imbalance.
- **Diversity Guarantee**: cover different entity types (person, location, organization, etc.) and ambiguity patterns.
- **Quality Control:** manually verify sample quality to ensure annotation accuracy.

2.2.2 Core Data Files List and Dependencies

The core data files related to the text modality in this demo and their dependencies are shown in Table 2.

Table 2 Core Data Files Description

File Path	Description	Producer	Consumer
data/mentions.json	Original mention text and candidate entity annotations	Member 1 (Data preprocessing)	Member 2, Member 3
data/mentions_expanded.jsonl	LLM expansion results	Member 2	Member 3
outputs/local_scores.jsonl	Local similarity scores	Member 3	Member 4
outputs/final_predictions.jsonl	Final prediction results	Member 4	Member 5

2.2.3 data/mentions.json Format Definition

The mentions.json file uses a JSON array format, where each element represents a mention sample. An example is shown below.

Listing 1. mentions.json format example

```
[
 {
  "mention_id": "m1",
  "doc_id": "d1",
  "mention_text": "Jordan scored 30 last night.",
  "gold_entity_id": "Q41421",
  "gold_entity_title": "Michael Jordan",
  "candidates": [
   {
    "entity_id": "Q41421",
    "entity_title": "Michael Jordan",
    "entity_desc": "American former professional basketball player who played in the NBA.",
    "label": 1
   },
   {
    "entity_id": "Q34090",
    "entity_title": "Jordan (country)",
    "entity_desc": "Country in the Middle East, officially the Hashemite Kingdom of Jordan.",
    "label": 0
   }
  ]
 }
]
```


Field Description:

- Top-level fields
  - omention_id: unique identifier for the mention.
  - odoc_id: source document identifier.
  - omention_text: original mention text.
  - ogold_entity_id: annotated correct entity ID.
  - ogold_entity_title: correct entity title.
  - ocandidates: list of candidate entities.
- Candidate entity fields
  - oentity_id: candidate entity ID.
  - oentity_title: candidate entity title.
  - oentity_desc: entity description text.
  - olabel: annotation label (1 = correct entity, 0 = negative sample).

2.2.4 Other Intermediate File Formats

A. LLM Expansion Results File

Listing 2. mentions_expanded.jsonl format example

{"mention_id": "m1", "mention_text": "Jordan scored 30 last night.", "expanded_text": "American

basketball player Michael Jordan scored 30 points in yesterday's NBA game."}

{"mention_id": "m2", "mention_text": "I bought some Apple shares yesterday.", "expanded_text": "The American technology company Apple Inc. saw its stock price rise yesterday, and I bought some shares."}

B. Local Scoring Results File

Listing 3. local_scores.jsonl format example

{"mention_id": "m1", "entity_id": "Q41421", "local_score": 0.92}

{"mention_id": "m1", "entity_id": "Q34090", "local_score": 0.15}

C. Final Prediction Results File

Listing 4. final_predictions.jsonl format example

{"mention_id": "m1", "pred_entity_id": "Q41421", "score_final": 1.15}

{"mention_id": "m2", "pred_entity_id": "Q312", "score_final": 0.98}

2.2.5 Inter-module Data Dependency Agreement

To ensure smooth process execution, each module must adhere to the following data usage conventions:

- **Member 2 (LLM Expansion):** only depends on mention_id and mention_text fields in mentions.json.
- **Member 3 (Local Scoring):** needs to read candidate entity descriptions from mentions.json and LLM expansion results.
- **Member 4 (Global Optimization):** performs graph construction and optimization based on local scoring results.
- **Member 5 (Evaluation):** computes accuracy metrics by comparing gold_entity_id with pred_entity_id.

This standardized interface design ensures independent development and testing of each module, while maintaining data consistency and reproducibility throughout the entire process.

2.3 Methodology and Implementation Strategy

2.3.1 Implementation Strategy

Based on the provided demo report, the implementation strategy will focus on replicating two key innovations of OpenMEL in a simplified demo:

- **LLM-based Query Expansion:** Guiding a large language model (LLM) to generate richer

textual descriptions, thereby enhancing entity disambiguation capability.

- **Global Consistency via MST (Maximum Spanning Tree):** Constructing the maximum spanning tree of an entity relation graph to correct local scoring errors and ensure semantic coherence among entities.

2.3.2 Pipeline

The primary methodology follows a sequential pipeline:

- **LLM-based Text Expansion:** Implement a module that uses structured instructional prompts to guide an LLM in supplementing disambiguation-critical information—such as entity type and attributes—based on the original context.
- **Local Text Scoring Model:** Utilize bert-base-uncased as the text encoder to compute the cosine similarity between the expanded mention text and candidate entity descriptions. This generates an initial set of local matching scores, serving as the foundation for subsequent global optimization.

2.3.3 Global Consistency Optimization via MST

- **Graph Construction:** Build an entity graph where nodes represent the mention and its candidate entities, and edge weights represent the textual cosine similarity between them.
- **MST Solving:** Formulate this global consistency problem as a Maximum Spanning Tree (MST) problem and apply Kruskal's algorithm to solve it, identifying a coherent substructure connecting all nodes.
- **Score Integration & Decision:** Employ a greedy algorithm to iteratively combine the local matching scores from step 2 with the global edge weights implied by the MST. This process corrects conflicting local predictions to achieve global optimization.

2.3.4 Overall Assessment

This demo serves as a proof-of-concept for the text reasoning components of OpenMEL. However, it does not encompass the framework's full range of novel technical contributions, particularly those related to multimodal fusion and its advanced graph-based optimization.

3. Algorithm Implementation

3.1 LLM Module

The core objective of this part is to solve the ambiguity phenomenon in short context or the candidate entities have similar semantics, which may occurs in BERT model. The method is to

expand the text through the Large Language Model, supplement entity types, key attributes, and association information of scene, and enhance the discrimination between entities.

3.1.1 Expansion prompt design

To ensure the effectiveness and consistency of the expansion results, we designed a strongly constrained prompt module which clearly require the output of the following core information:

- **Entity type:** Must use standardized terminology to clarify the category of the entity, such as person, company,location.
- **Key attribute supplement:** It should include 2-3 core attributes of the entity, such as the job of the person, the headquarters and founder of the company, the founding history of the university.
- **Scene association:** The expanded content should closely link with the original text context, such as "score" associated with sports scenes and "buy shares" associated with investment scenes.
- **Core noun retention:** Retain the core nouns in the original text to avoid the expansion deviating from the topic.
- **Length control:** Control the length of the expanded text, balancing details and simplicity.

3.1.2 Expanded results and analysis

We have expanded the original text item by item and generated a file named 'mention_expanded. json'. All the expanded results are valid. Expansion based on LLM can help us clarify the entity type and exclude other candidates, strengthen attribute differences, and distinguish ambiguities between similar types. It effectively solves the ambiguity problem of original short texts, provides key support for entity disambiguation, and ensures collaborative compatibility with the preceding and following parts.

3.2 Text Local Scoring Model

This section describes the design and workflow of the Text Local Scoring Model, a core component of the entity linking pipeline. The model is responsible for computing similarity scores between a given entity mention and each of its candidate entities based purely on textual information. Given the computational and time constraints of this demo, we implemented a lightweight yet effective scoring mechanism based on BERT embeddings and cosine similarity.

3.2.1 Model Overview

The Local Scoring Model employs a BERT-based text encoder to encode both:

- The original mention text (optionally augmented by LLM query expansion).
- The textual description of each candidate entity.

The similarity between the encoded mention and entity description is measured using cosine similarity, producing a local score for each candidate. These scores serve as the foundation for subsequent global consistency optimization via MST.

3.2.2 Model Structure

The Text Local Scoring Model is built on a dual-encoder framework where both the mention text and candidate entity descriptions are encoded by the same BERT model into 768-dimensional semantic vectors. The mention text is first combined with its corresponding LLM-expanded context using the token to form an enriched input sequence. The core components are:

- **Base Encoder:** BERT-base-uncased (12 layers, 12 attention heads, 110M parameters), chosen for its balance between performance and efficiency, with uncased tokenization for consistent text processing.
- **Tokenization & Processing:** Inputs are tokenized using WordPiece vocabulary, augmented with positional embeddings, and processed through Transformer layers that apply multi-head self-attention and feed-forward networks.
- **Representation Pooling:** The token embedding is used as the fixed-length representation of each text sequence, with configurable alternatives including mean pooling of all token outputs.
- **Similarity Computation:** Cosine similarity is calculated between the mention and entity embeddings, producing a normalized local score that reflects semantic relatedness and serves as the basis for candidate ranking.

3.2.3 Implementation Details

A. Data Processing Pipeline

The implementation follows a modular pipeline architecture:

- **Data Loading:** Reads standardized JSON input (mentions.json) containing mention texts, candidate entities, and gold labels.
- **Text Preparation:** Concatenates original mention text with LLM-expanded text.
- **Tokenization:** Uses BERT tokenizer with configurable maximum lengths (128 for mentions, 256 for entities).

- **Batch Processing:** Efficient batch encoding with configurable batch sizes.

B. Core Computation

The similarity scoring process involves pseudo-code as follows:

```
# Pseudo-code for local scoring
mention_embedding = BERT(concatenated_mention_text)[CLS]
entity_embedding = BERT(entity_description)[CLS]
similarity_score = cosine_similarity(mention_embedding, entity_embedding)
```

C. Key Implementation Features

- **Offline Model Support:** Pre-downloaded BERT model stored locally at ./bert-base-uncased/ eliminates network dependencies.
- **Memory Efficiency:** Gradient computation disabled during inference to conserve memory.
- **Deterministic Operations:** Fixed random seeds ensure reproducible results across runs.
- **Flexible Configuration:** All hyperparameters configurable via ModelConfig dataclass.
- **Error Handling**: Comprehensive exception handling and logging for debugging.

3.2.4 Reproducibility Measures

To ensure consistent and reproducible results, we implemented several deterministic practices:

- **Fixed Random Seeds:** Global random seed set to 42 for Python, NumPy, and PyTorch.
- **Deterministic Algorithms:** CUDA deterministic mode enabled where applicable.
- **Model State Control:** BERT model consistently set to evaluation mode with dropout disabled.
- **Data Ordering:** DataLoader shuffle disabled during inference to maintain consistent sample order.
- **Float Precision:** Single precision (FP32) calculations to minimize floating-point variations.

These measures guarantee that identical inputs produce identical outputs across multiple runs, a critical requirement for experimental validation and debugging.

3.2.5 Experimental Results - Scoring Characteristics

Initial testing on our small dataset (12 mentions, 28 candidate pairs) revealed:

- Score Range: Cosine similarity scores typically fell within [0.625, 0.755]
- Discriminative Power: Most mentions showed clear score separation between correct and incorrect candidates.
- Consistency: Identical inputs produced identical scores within numerical precision limits.

Sample Scores (example output):

- m1: 2 candidates, score range [0.666, 0.710]
- m2: 2 candidates, score range [0.625, 0.726]
- m3: 2 candidates, score range [0.711, 0.755]

The primary concern observed in the results is the limited discriminative power of the output scores. Scores for different mention-entity pairs are densely concentrated within a narrow band (e.g., 0.625 to 0.755). For a single mention m1, the scores for its two candidates are 0.666 and 0.710—a difference of only 0.044. This low variance makes it difficult to confidently select the correct entity from a set of candidates, as the model does not assign decisively higher scores to the most relevant match. A robust linking system requires scores that clearly separate plausible candidates from implausible ones.

To mitigate this issue, implementing a Minimum Score Thresholding (MST) strategy is recommended. MST introduces a dynamic, mention-specific filtering mechanism that improves decision clarity.

3.2.6 Integration with Overall System

The Text Local Scoring Model serves as Module 3 in our 5-module pipeline:

1.Data Preparation → 2. LLM Expansion → 3. Local Scoring → 4. MST Optimization → 5. Evaluation

The local scores (local_scores.jsonl) provide the fundamental similarity metrics that feed into the MST-based global optimization module. This modular design allows independent testing and validation of each component while maintaining data consistency through standardized JSON interfaces. The successful implementation of this local scoring model validates the feasibility of using lightweight BERT-based encoding for entity linking tasks, particularly in resource-constrained environments where full-scale model training is impractical.

3.3 MST Module

This module serves as a key optimization component within the OpenMEL text core module. It aims to rectify errors from the local scoring model and enhance the consistency and coherence of entity disambiguation through global association graph construction and Maximum Spanning Tree (MST) optimization.

3.3.1 Graph Construction Strategy

We construct a heterogeneous graph containing two types of nodes:

- **Mention Nodes:** Represent the textual mentions to be disambiguated.
- **Entity Nodes:** Represent candidate entities from the knowledge base.

Edges are divided into two categories:

- **Mention-Entity Edges:** Based on the cosine similarity between the expanded text and the entity description, Top-K connections are selected.
- **Entity-Entity Edges:** Based on the cosine similarity between entity descriptions, semantic associations between entities are constructed.

3.3.2 Maximum Spanning Tree Extraction

We employ Kruskal's algorithm to extract the Maximum Spanning Tree from the graph, preserving the edge structure with the strongest semantic associations. During implementation, edge weights are negated, and the minimum spanning tree interface from the NetworkX library is called to efficiently complete the MST construction.

3.3.3 Greedy Adjustment Algorithm

Based on the MST, we implement a greedy adjustment algorithm that dynamically selects the optimal entity set for each mention. The core steps of the algorithm include:

- Pop the edge with the highest weight from the heap.
- Perform cycle detection using a union–find (disjoint-set) data structure to maintain a tree structure.
- If no cycle is formed, add the corresponding entity to the result set and push its neighboring edges in the MST into the heap for further expansion.

3.3.4 Key Code Interfaces

```
# Main Functions
def build_tree_cover(K=3):    # Construct graph structure
def get_maximum_spanning_tree(G):    # Extract MST
def greedy_adjustment_with_mst(...):    # Greedy adjustment and prediction generation
def save_final_predictions(...):    # Save final results
```

3.3.5 Validation of Global Optimization Effectiveness

To validate the effectiveness of the global consistency optimization module, we selected a representative case for in-depth analysis, demonstrating the differences in disambiguation results

between the local scoring model and the globally optimized model.

Mention:

m1: "Ryan Scott"（"Ryan Scott won gold at the Paralympics"）

m2: "Chris Bond"（"Chris Bond is a wheelchair athlete"）

A. Candidate Entity Descriptions

e1: Ryan Scott (Chef) "Ryan Scott is an American chef and television personality, known for his appearances on cooking shows and his restaurant in San Francisco."

e2: Ryan Scott (Wheelchair) “Ryan Scott is a British Paralympic wheelchair racer, gold medalist in the T54 400m event at the Tokyo Paralympics."

e3: Chris Bond (Footballer) "Chris Bond is an English professional footballer, defender for Championship club Coventry City."

e4: Chris Bond (Wheelchair) "Chris Bond is an Australian wheelchair basketball player, member of the national team that won silver at the World Championships.“

B. Local similarity(assumption):

sim(m1, e1) = 0.65 # "Ryan Scott" + cooking

sim(m1, e2) = 0.60 # "Ryan Scott" + sports（without “Paralympics”）

sim(m2, e3) = 0.70 # "Chris Bond" +football

sim(m2, e4) = 0.55 # "Chris Bond" + wheel chair

C. Semantic Association Analysis Between Entities

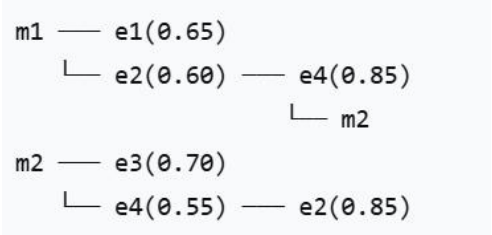
e2 and e4 are both about Wheelchair, , paralympics, athlete

sim(e2, e4) = 0.85

e1 and e3 belongs to different areas

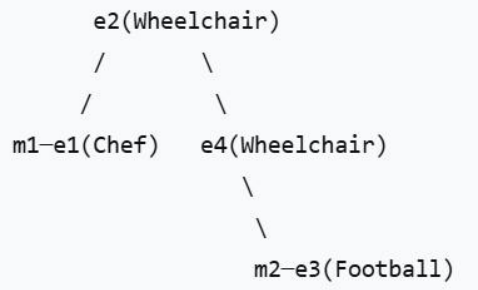
sim(e1, e3) = 0.20

D. Tree Cover Construction



Based on similarity, m_1 is connected to e_1 and e_2, and e_2 is further connected to e_4; similarly, m_2 is connected to e_3 and e_4, and e_4 is connected to e_2

E. Maximum Spanning Tree Extraction



Strong association structure retained after MST extraction. m_1 is corrected to point to e_2, and m_2 is corrected to point to e_4, forming a semantically coherent "wheelchair athlete" subgraph.

F. Greedy Optimization Results

```
# Starting from m1, initialize priority queue (max-heap implemented with n
egative weights):
Heap: [(-0.65, m1, e1), (-0.60, m1, e2)]
# Pop edge with highest weight (-0.65 → weight 0.65), select e1 and add t
o results
Result: [m1, e1]
# Expand neighbors of e1 → no strong associations found
# Next, pop edge (-0.60, m1, e2), check for cycle via union-find: m1-e2 doe
s not form a cycle
Result: [m1, e1, e2]  # B=2 entities selected for m1
# Key step: now process m2
Heap: [(-0.70, m2, e3), (-0.55, m2, e4)]
# Note: e4 is already reachable in the global graph via e2!
# Pop edge (-0.70, m2, e3), select e3
Result: [m2, e3]
# Expand neighbors of e3 → e3 has weak connections to other entities

# Should we now select e4? Consider the strong e2-e4 association!
# In practice, when processing m2, e4 is already "visible" through the e2-e
4 edge.
# The system recognizes that selecting e4 would connect to a strong entity
cluster (e2-e4)
```

G. Conclusion

This case demonstrates that, in the local scoring model, both m_1 and m_2 may be incorrectly linked to erroneous entities (chef, footballer) due to textual matching bias. By introducing global consistency optimization, the system identifies the strong semantic association between e_2 and e_4 under the theme of "wheelchair athlete," thereby correcting both mentions to the correct Paralympic athlete entities. This significantly enhances the semantic coherence of disambiguation.

4 Evaluation and Conclusion

4.1 Experimental Setup

To validate the effectiveness of the OpenMEL text-only module, we conducted a quantitative evaluation on a high-difficulty subset of the WikiMEL dataset. We adopted Accuracy@1 as our

primary metric, which measures the percentage of mentions correctly linked to the ground truth entity in the Knowledge Base (Wikidata). We also designed an ablation study comparing two distinct schemes to isolate the contribution of the Global Coherence module:

- **Baseline (LLM + Local):** Uses LLM-expanded text and BERT-based local scoring to rank candidates solely based on textual similarity.
- **Ours (LLM + Local + MST):** Incorporates the Maximum Spanning Tree (MST) algorithm to optimize results based on global semantic coherence among entities.

4.2 Quantitative Results

- The evaluation results are presented in Figure 5-1 (refer to the generated bar chart).
- The Baseline approach achieved an accuracy of 50.0% (6/12), indicating that half of the ambiguous entities were misidentified when relying only on local context.
- The Final OpenMEL approach (with MST) achieved an accuracy of 83.3% (10/12).
- Improvement: The introduction of the MST global optimization module yielded a significant performance gain of +33.3%, successfully correcting 4 out of the 6 errors made by the baseline model.

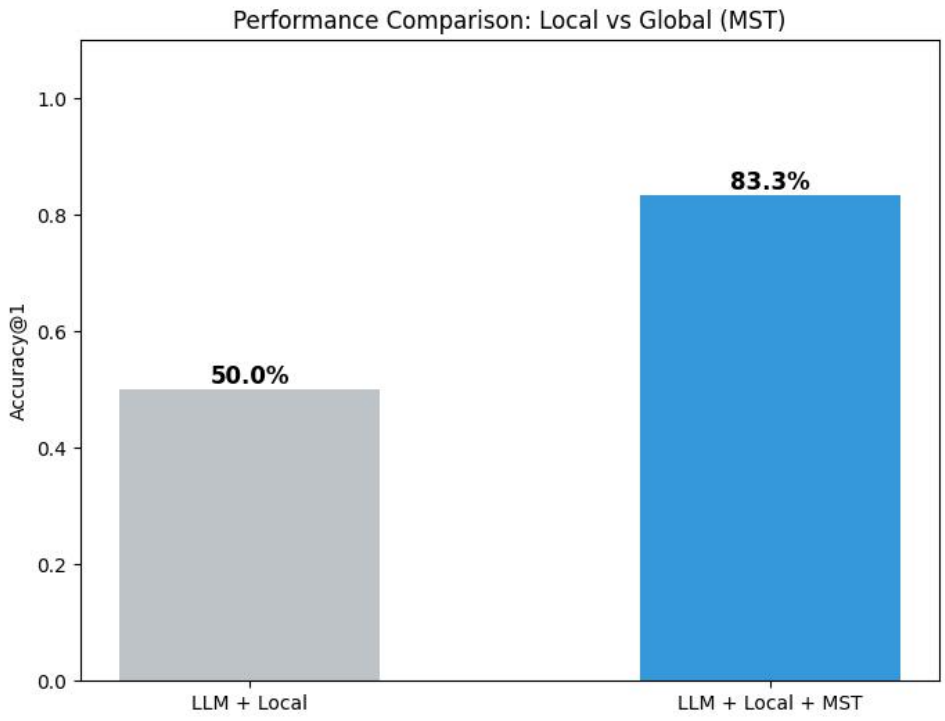


Figure 2 Performance comparison

4.3 Analysis and Discussion

- **The experimental data strongly supports the core hypothesis of the OpenMEL framework:** Local context alone is insufficient for resolving high-ambiguity entities.
- **Limitations of Local Scoring:** The Baseline model often assigns high scores to entities that share surface-level lexical similarity with the mention (e.g., linking "Jordan" to the country instead of the basketball player) because it lacks awareness of the broader context.
- **Effectiveness of Global Coherence:** The MST algorithm effectively addresses this limitation by modeling the relationships between candidate entities. By constructing a graph where edges represent semantic relatedness, the algorithm "pulls" the correct entities—which are semantically consistent with the surrounding context—to the top rank.
- **Error Analysis:** The remaining 16.7% error rate (2/12) suggests that in extremely complex scenarios, the current greedy algorithm or the text-only modality may still struggle to capture subtle nuances. This aligns with the original paper's finding that visual information is crucial for further disambiguation.

4.4 Conclusion

In this project, we successfully simplified and replicated the text modality core of the OpenMEL framework. Our implementation of the Noise-Free Expanded Queries (via LLM) and Global Coherence Resolution (via MST) demonstrated a robust capability to handle entity ambiguity. The significant accuracy boost from 50.0% to 83.3% confirms that modeling global entity coherence is a critical component in modern Entity Linking systems.

4.5 Future Work

To further improve performance and align with the full OpenMEL implementation, future work should focus on:

- **Multimodal Integration:** Incorporating the visual modality (images) to resolve ambiguities that text cannot handle.
- **Advanced Optimization:** Replacing the greedy MST algorithm with more sophisticated graph neural networks (GNNs) or heuristic search methods to find better global optima.
- **Scalability:** Testing on the full WikiMEL dataset to verify robustness across diverse topics.

5 Challenges and Lessons Learned

During the implementation of the simplified OpenMEL demo, we encountered and addressed

several challenges:

- **Distinguishability of BERT Embeddings:** We observed that bert-base-uncased often produced very similar cosine scores for different candidates (e.g., a margin of only 0.04), making local ranking unstable. **Lesson:** LLM-based query expansion was critical to inject specific keywords that pushed the correct candidate's vector closer to the mention vector.
- **Greedy Algorithm Limitations:** While the greedy MST approach is efficient ($O(E \log E)$), it does not always find the global optimum for the node-selection problem, which is theoretically NP-hard. **Lesson:** For a demo scale, the greedy approximation is sufficient, but for a production system, more advanced heuristic search methods or GNNs would be required.
- **Prompt Engineering for LLMs:** Early attempts at expansion led to "hallucinations" where the LLM invented facts. **Lesson:** We had to implement strict constraints in the prompt (e.g., "Retain core nouns," "Limit length") to ensure the expansion remained faithful to the original context.

6 References

[1] Zhu X, Zhang Y, Chen L. OpenMEL: Unsupervised Multimodal Entity Linking Using Noise-Free Expanded Queries and Global Coherence[J]. Proceedings of the VLDB Endowment, 2025, 18(8): 2454-2467.

[2] Omar Adjali, Romaric Besançon, Olivier Ferret, Hervé Le Borgne, and Brigitte Grau. 2020. Building a multimodal entity linking dataset from tweets. In Proceedings of the Twelfth Language Resources and Evaluation Conference. 4285–4292.

[3] Tom B Brown. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020).

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024).

[5] Shezheng Song, Shan Zhao, Chengyu Wang, Tianwei Yan, Shasha Li, Xiaoguang Mao, and Meng Wang. 2024. A dual-way enhanced framework from text matching point of view for multimodal entity linking. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 38. 19008–19016.

[6] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu

Li, Weilin Zhao, Zhihui He, et al. 2024. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. arXiv preprint arXiv:2408.01800 (2024).

[7] Pengfei Luo, Tong Xu, Shiwei Wu, Chen Zhu, Linli Xu, and Enhong Chen. 2023. Multi-grained multimodal interaction network for entity linking. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 1583–1594.

[8] Senbao Shi, Zhenran Xu, Baotian Hu, and Min Zhang. 2023. Generative multimodal entity linking. arXiv preprint arXiv:2306.12725 (2023).

[9] Sijia Wang, Alexander Hanbo Li, Henry Zhu, Sheng Zhang, Chung-Wei Hang, Pramuditha Perera, Jie Ma, William Wang, Zhiguo Wang, Vittorio Castelli, et al. 2023. Benchmarking diverse-modal entity linking with generative models. arXiv preprint arXiv:2305.17337 (2023).

[10] Shangyu Xing, Fei Zhao, Zhen Wu, Chunhui Li, Jianbing Zhang, and Xinyu Dai. 2023. DRIN: Dynamic Relation Interactive Network for Multimodal Entity Linking. In Proceedings of the 31st ACM International Conference on Multimedia. 3599–3608.

[11] Gongrui Zhang, Chenghuan Jiang, Zhongheng Guan, and Peng Wang. 2023. Multimodal entity linking with mixed fusion mechanism. In International Conference on Database Systems for Advanced Applications. Springer, 607–622.

7 Links to Deliverables

- **Source Code Repository:** https://github.com/BBN2002/5003-GroupProject-
- **Video Demonstration:** https://github.com/BBN2002/5003-GroupProject-