Supplementary Material of High-Order Consistency-Guided User Identity Linkage with Large Language Model

Appendix A. Problem Definition

A platform network consists of user nodes, inter-user relationships, and user-generated content, and can be represented as an attributed graph structure $G = (V, E, A)$, where V denotes the set of users, E represents user relationships, and A represents user attributes (e.g., textual content). In a cross-platform scenario, the same user exists in different networks and is regarded as an anchor user. The objective of this task is to infer all unknown matching user identity pairs given a set of known anchor user pairs. The main notations and their definitions used in this paper are listed in Table 1.

Table 1: Notations and Descriptions

Notation	**Description**
G_X, G_Y	Source network and target network
v	Node in the network, representing a user
(v_i^X, v_j^Y)	Cross-network anchor node pair, where $v_i^X \in G_X$, $v_j^Y \in G_Y$
A_t	Textual attribute of a node (i.e., user-generated text content)
$\mathcal{N}_v$	Set of neighboring nodes of v
$\mathcal{W}_v$	Random walk sequence of node v
w_1, w_2	Weight parameters controlling the influence of node degree and attribute length in random walks
d_u	Degree of neighboring node u
T_v^a	Attribute text chain of node v
T_v^n	Neighbor text chain of node v
T_v^r	Random walk text chain of node v
Q_a	Topic of attribute text chain T_v^a
Q_n	Topic of neighbor text chain T_v^n
Q_r	Topic of random walk text chain T_v^r
$\mathcal{L}_{\text{finetune}}$	Cross-entropy loss function for fine-tuning the BERT model
$\mathbf{e}_v^a$	Raw attribute embedding of node v
$\mathbf{h}_v^a$	Topic embedding of attribute text chain
$\mathbf{h}_v^{\text{attr}}$	Final attribute embedding of node v
$\mathbf{h}_v^n, \mathbf{h}_v^r$	Topic embeddings of neighbor and random walk text chains, respectively
$\mathbf{h}_v^{\text{str}}$	Fused embedding combining attribute and structural information of node v
$\mathbf{h}_v$	Final embedding of node v
$\mathbf{h}_i^X, \mathbf{h}_j^Y$	Final embeddings of node i in G_X and node j in G_Y, respectively
$D(\mathbf{h}_i^X, \mathbf{h}_j^Y)$	Distance metric between anchor nodes (v_i^X, v_j^Y)

Appendix B. Algorithm

The overall training and implementation process of the model is shown in Algorithm 1.

Algorithm 1 UIL-HC-MV Algorithm

Input: Networks G_X and G_Y with node attributes and structural information, anchor node pairs (v_i^X, v_j^Y), parameters of the large-language model and the BERT model, etc.
Output: Cross-network user identity matching results

```
1:  Step 1: Text Chain Design
2:  for Each node $v$ do
3:      Generate the attribute text chain $T_v^a$, the neighbor text chain $T_v^n$, and the random walk text chain $T_v^r$
4:  end for
5:  Step 2: Topic Generation of Text Chains
6:  for Each node $v$ do
7:      Obtain the topic representations $Q_a$, $Q_n$, and $Q_r$ of $T_v^a$, $T_v^n$, and $T_v^r$ respectively through the prompts of the large-language model
8:  end for
9:  Step 3: BERT Fine-tuning Strategy Based on Text Chain Topics
10: Use the three types of text chain topics of anchor node pairs as positive samples and the topics of random non-anchor node pairs as negative samples to fine-tune the BERT encoder based on the cross-entropy loss function
11: Step 4: Topic Representation Learning
12: for Each node $v$ do
13:     Encode $Q_a$, $Q_n$, and $Q_r$ through the fine-tuned BERT to obtain $\mathbf{h}_v^a$, $\mathbf{h}_v^n$, and $\mathbf{h}_v^r$ respectively
14:     Add the original node attribute vector $\mathbf{e}_v^a$ and $\mathbf{h}_v^a$ to get the attribute representation $\mathbf{h}_v^{\text{attr}}$
15:     Add $\mathbf{h}_v^n$ and $\mathbf{h}_v^r$ to get the structural representation $\mathbf{h}_v^{\text{str}}$
16:     The final representation $\mathbf{h}_v = \text{AddNorm}\big(\mathbf{h}_v^{\text{attr}}, \mathbf{h}_v^{\text{str}}\big)$
17: end for
18: Step 5: Cross-network User Identity Matching
19: for Each pair of anchor nodes $(v_i^X, v_j^Y)$ do
20:     Calculate the similarity between $\mathbf{h}_{v_i^X}$ and $\mathbf{h}_{v_j^Y}$ and output the matching result
21: end for
```

Appendix C. Implementation Details in the Experiment setup

Implementation Details Regarding experimental parameter settings: In DBLP_1, 70% of the anchor nodes were selected as the training set for BERT fine-tuning, while both DBLP_1 and DBLP_2 used 30% of the anchor nodes for testing. This chapter maintains a unified representation dimension of 768 for all generated text chains. During the design of text chains and the use of large language models (LLMs) for high-order information extraction, the total token limit for each text chain was set to 4096 tokens. Specifically, the termination length for random walk text chains was fixed at 4096 tokens, whereas for neighbor text chains, if the raw collaboration data fell short of 4096 tokens, neighbor attribute texts were appended to meet the length requirement (each supplementary text did not exceed 800 tokens). For random walk text chains, the weight parameters controlling node

Table 2: Runtime Comparison Across Methods (minutes)

Dataset	Method						
	UIL-HC-MV	Grad-Align	CENALP	GAlign	NeXtAlign	NetTrans	MAUIL
DBLP_1	40	45	360	8	140	43	24
DBLP_2	18	35	255	4	160	30	30

transition probabilities, w_1 and w_2, were set to 0.7 and 0.3, respectively. Regarding the parameter configuration for LLM responses, the output text length was capped at 512 tokens, aligning with the maximum input length for BERT fine-tuning. To ensure fair comparisons, all baseline methods in this section were configured according to their original papers, prioritizing efficient and reasonable parameter choices.

Appendix D. Computational Efficiency Analysis

Table 2 compares runtime (training + inference) under uniform hardware (Intel I7-12700H, 40GB RAM, NVIDIA RTX 3070Ti 8GB).

UIL-HC-MV achieves balanced performance-efficiency tradeoffs. Notably: - DBLP_1 runtime includes BERT fine-tuning - DBLP_2 uses pre-fine-tuned BERT from DBLP_1, reducing runtime by 55% This demonstrates that transferred BERT models eliminate redundant computation, enhancing practical applicability in multi-dataset scenarios.

Appendix E. Heterogeneity Mitigation Analysis

We use K-Means clustering (k=2) with Davies-Bouldin Index (DBI) and Silhouette Coefficient to evaluates heterogeneity reduction. Lower DBI and higher Silhouette values (range: [-1,1]) indicate better clustering.

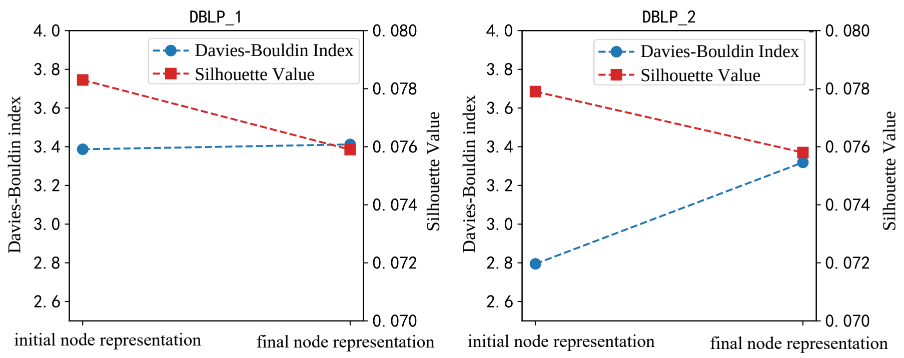

Figure 1: Cross-network Heterogeneity Mitigation Analysis

Figure 1 reveals increased DBI and decreased Silhouette values for the final representations (fused attribute-structure higher-order information) versus initial attribute representations across both datasets, demonstrating effective cross-network heterogeneity reduction through our model.