PUER: Boosting Few-shot Positive-Unlabeled Entity Resolution with Reinforcement Learning

Yaoshu Wang[1], Mengyi Yan[2]*, Wei Wang[3]

[1]Shenzhen Institute of Computing Sciences, [2]Shandong University
[3]The Hong Kong University of Science and Technology, Guangzhou
[1]yaoshuw@sics.ac.cn, [2]yanmy@sdu.edu.cn, [3]weiwcs@ust.hk

Abstract

Entity resolution is a fundamental problem in data management that aims to identify all duplicate entries within collections of multi-attribute tuples. Most existing works focus on supervised learning, relying on large amounts of high-quality labeled data, including both positive and negative tuple pairs that are meticulously prepared. However, in reality, the manual annotation process is labor-intensive; in particular, selecting high-quality negative data for labeling is both important and challenging. In this paper, we propose an end-to-end ER solution, PUER, to address low-resource entity resolution (ER) by leveraging Large Language Models (LLMs) in a Positive-Unlabeled (PU) learning setting, where only a small number of positively labeled examples, *e.g.,* 50, and unlabeled data are provided. Unlike directly fine-tuning LLMs in a supervised manner, we solve the entity matching task using reinforcement learning and propose a self-adaptive reward function in the process of RL. To enhance performance, we design an iterative workflow based on the co-training mechanism that fully utilizes entity blocking component to assist the entity matching. This workflow aims to improve the robustness and quality of pseudo-labels so that the performance of entity matching is improved. Comprehensive experimental results on various benchmark datasets demonstrate the superiority of PUER. Full version and code are available[1].

1 Introduction

Entity resolution (ER) aims to identifying all tuple pairs from two relational tables that refer to the same entities, making it a key components of data cleaning with the goal of deduplicating records in datasets. Traditionally, the ER task consists of two components, entity blocking (EB) and entity matching (EM). Entity blocking efficiently retrieves potentially matched tuple pairs, while entity matching verifies whether these tuple pairs refer to the same entities.

Traditionally, the entity resolution has been extensively studied and most of solutions reply on a sufficient number of annotated tuple pairs to achieve good performance. However manual annotation is costly, as demonstrated by methods like Ditto (Li et al., 2020b) and DeepMatcher (Mudgal et al., 2018). To address it, a few existing EM approaches focus on unsupervised learning, semi-supervised learning and active learning. For instance, TDmatch (Ahmadi et al., 2022) is an unsupervised ER approach based on graph creation and random walk, while only relying on the data distribution cannot have very high accuracy due to the extreme class imbalance. PromptEM (Wang et al., 2022) generates pseudo-labels for low-resource ER in the semi-supervised learning. While it partially alleviates the annotation cost, selecting and labeling high-quality positive and negative tuple pairs from large datasets remains challenging. Active learning approaches, *e.g.,* (Arasu et al., 2010), select ambiguous tuple pairs for user labeling, but this also incurs significant manual annotation costs. In this work, we focus on the few-shot Positive-Unlabeled (PU) learning, where only a small number of labeled positive tuple pairs are provided along with two relational tables. To the best of our knowledge, we are the first to explore the ER task in the few-shot PU learning context.

Entity resolution, which typically replies on both positive and negative training data, is particularly relevant to few-shot PU learning (Bekker and Davis, 2020). In practice, only users who detect duplicate issues in their datasets often invest resources to find and integrate these duplicates, so that these detected duplicates instances are treated as labeled positive data. Negative tuple pairs are generally not provided. Additionally, in a search engine scenario, users might ask a question and receive multiple semantically identical re-

* Corresponding author
[1]https://github.com/authurlord/PUER

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 24567–24579
November 4-9, 2025

sponses (Niu et al., 2016). These responses are considered positive data. While negative tuple pairs can be relatively easy to collect, acquiring high-quality negative pairs in large-scale datasets is non-trivial and requires manual annotation and verification, presenting a challenge for annotators (Wang et al., 2024b). Building on these observations, we investigate the ER task within the framework of few-shot PU learning to address these challenges effectively with minimal labelling cost, while aligning with human preference.

However, existing ER methods based on pre-trained Language Models (e.g., RoBERTa) primarily learn distribution and decision boundaries from large sets of annotated samples. This approach results in inefficiencies in labeled data utilization, placing these methods at a disadvantage in few-shot scenarios. Furthermore, they lack the capability to generalize well, and cannot achieve high-performance EM tasks based solely on a limited number of positive sample annotations. Recently, as the advanced performance of large language models (LLMs), they have been explored in the entity matching task. JellyFish (Zhang et al., 2024) addresses various data pre-processing tasks, including entity matching, by leveraging LLMs in the instruction-tuning and reasoning manner. TableGPT (Li et al., 2024b) employs and fine-tunes online GPT-3.5 on various data pre-processing tasks. Considering the data privacy concerns, we focus on using local LLMs that are open-sourced and can be fine-tuned in local environments.

Motivated by the above considerations, we explore the ER problem within the framework of few-shot PU learning by harnessing the capabilities of local LLMs. Our objective is to fully utilize entity blocking to assist the entity matching process and to develop an LLM-based model that can efficiently and effectively retrieve all matched tuple pairs from limited labeled data. Unlike traditional methods that treat entity matching purely as a binary classification task, our approach is the first to formulate entity matching as a reinforcement learning problem while simultaneously fine-tuning the model using both Supervised Fine-Tuning (SFT) and reinforcement learning. Furthermore, to integrate entity blocking with entity matching, we introduce an iterative workflow that progressively generates high-quality pseudo-labels, facilitating mutual learning among these components.

Our contributions are as follows:

- Beyond binary classification, we are the first to employ **reinforcement learning** to solve the entity matching, and design a self-adaptive reward function to enhance the convergence speed.
- We propose an **end-to-end** entity resolution workflow that iteratively make full use of the entity blocking model to select high-quality training data and jointly fine-tunes two entity matching models through a co-training mechanism.
- We conduct **comprehensive experiments** to evaluate the efficiency and effectiveness of our approach, demonstrating its superiority over existing methods.

2 Related Work

We classify ER into entity blocking and matching.

Entity blocking. We classify entity blocking as (1) rules, *e.g.,* handcrafted rules (Papadakis et al., 2020, 2014; Fan et al., 2009; Kejriwal and Miranker, 2015), and learned rules (Michelson and Knoblock, 2006; Kejriwal and Miranker, 2015; Singh et al., 2017a; Paulsen et al., 2023), (2) traditional ML, *e.g.,* (C. et al., 2018; Efthymiou et al., 2015), and (3) deep learning, *e.g.,* (Thirumuruganathan et al., 2021; Brinkmann et al., 2024; Wang et al., 2023; Reimers and Gurevych, 2019; Wu et al., 2023; Wang et al., 2024a), which retrieve potentially matched tuple pairs from large-scale datasets.

Entity matching. There are host of works on entity matching, including rule-based methods (Guo et al., 2010; Fan et al., 2011; Whang and Garcia-Molina, 2013; Singh et al., 2017b), ML-based methods (Konda et al., 2016; Bilenko and Mooney, 2003; Wu et al., 2020) and deep learning-based methods (Li et al., 2020b; Mudgal et al., 2018; Ebraheem et al., 2018; Zhao and He, 2019; Li et al., 2020a; Fu et al., 2019). Recently low resource entity matching based on deep learning models has been paid attention, including (1) active learning ER, *e.g.,* (Qian et al., 2017; Meduri et al., 2020; Kasai et al., 2019; Nafa et al., 2022), (2) data augmentation ER, *e.g.,*Rotom (Miao et al., 2021), (3) unsupervised learning ER, *e.g.,* (Zeng et al., 2024; Ahmadi et al., 2022), (4) transfer learning ER, *e.g.,* (Kirielle et al., 2022; Tu et al., 2022; Sun et al., 2024; Loster et al., 2021), (5) semi-supervised learning ER, *e.g.,*PromptEM (Wang et al., 2022), (6) multi-task learning, *e.g.,*Unicorn (Fan et al., 2024a). and (7) information fusion (Yao et al.,

2021). There are also works to combine the entity blocking and matching models for mutual learning, *e.g.,* (Wu et al., 2023; Wang et al., 2023; Li et al., 2021), and works by leveraging local LLMs (Zhang et al., 2024; Wadhwa et al., 2024) and online LLMs (Li et al., 2024b; Wang et al., 2025; Li et al., 2024a; Fan et al., 2024b). However, none of the above works address few-shot the Positive-Unlabeled setting, such that only as small number of positive instances are given, which is more practical in real-life.

3 Preliminaries

In this section, we first present the ER problem, and then introduce the entity blocking and matching.

3.1 Problem Formulation

Given two relational tables of multi-attribute tuples, the goal of entity resolution (ER) is to identify pairs of tuples that refer to the same entity. The ER task generally consists of two main components: entity blocking and entity matching. The entity blocking component efficiently retrieves a candidate set of potentially matching tuple from large tables, thereby avoiding the quadratic time complexity of comparing all tuple pairs between relational tables. The entity matching component then predicts whether tuple pairs in the candidate set are matches.

Definition 1: (ER under the few-shot positive-unlabeled setting.) Given two relational tables of multi-attribute tuples $\mathcal{R}_l$ and $\mathcal{R}_r$, and a set $\mathcal{P}$ consists of a small number of positive tuple pairs, the objective of few shot PU entity resolution (ER) is to identify all matching tuple pairs from $\mathcal{R}_l \times \mathcal{R}_r$. □

In this paper, we mainly focus on addressing entity matching task of ER. We fully utilize existing entity blocking techniques to enhance the efficiency and effectiveness of the matching process.

3.2 Entity Resolution

Following previous work (Wu et al., 2023), we decompose entity resolution into entity blocking and entity matching.

Entity blocking. The entity blocking Blocker, primarily utilizes the SentenceBert model, denoted as $\mathcal{F}_{\mathsf{RAG}}$, to transform each tuple t into an embedding vector (Thirumuruganathan et al., 2021; Wang et al., 2023; Wu et al., 2023; Li et al., 2020b). Given the training data $(t, \mathcal{P}_t, \mathcal{N}_t)$, where $\mathcal{P}_t$ and $\mathcal{N}_t$ represent the positive and negative sets of tuples that match and mismatch with t, we fine-tune $\mathcal{F}_{\mathsf{RAG}}$ via contrastive learning (Oord et al., 2018).

Entity matching. Previous work (Wang et al., 2025) formulates the entity matching into two subtasks: Matcher $\mathcal{F}^{\mathsf{M}}_{\mathsf{EM}}$ and Selector $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$.

Matcher. Given a tuple pair (t, s) and a domain-specific prompt pt_m as EM instruction, we could query LLM to transform (t, s) to a binary decision $p_m \in \{\mathsf{Yes}, \mathsf{No}\}$, *s.t.* $p_m = \mathsf{LLM}(\mathsf{pt}_m, (r, s_i))$. The Matcher $\mathcal{F}^{\mathsf{M}}_{\mathsf{EM}}$ is consistent with all existing entity matching works, *e.g.,* JellyFish (Zhang et al., 2024), Ditto (Li et al., 2020b), aiming to identifying whether a tuple pair is matched or not. Matcher is supervised fine-tuned (SFT) with LoRA and aims to provide domain-specific decision boundary.

Selector. Selector takes a pivotal tuple t, a list of candidate tuples $\mathcal{C}_s(t) = \{s_1, \ldots, s_{|\mathcal{C}_s|}\}$, and a prompt pt_s as inputs, and outputs a list of positive ones in $\mathcal{C}_s(t)$. It lets LLMs check more examples so that they make correct decision. Selector targets at re-ranking $\mathcal{C}_s$ by simulating human preference.

The Matcher subtask is mainly used in most EM approaches, *e.g.,* JellyFish, while Selector has not been as extensively studied. Although (Wang et al., 2025) introduced it to address the EM, they did not further fine-tune it to improve its performance.

4 RL-based Entity Matching

As discussed above, the entity matching task is formulated as two sub-tasks, Matcher $\mathcal{F}^{\mathsf{M}}_{\mathsf{EM}}$ and Selector $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$. In this section, we focus on how to fine-tune the Selector to make policies from a list of candidate tuples. Here we assume that $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ is fine-tuned using a (pseudo-)labeled training dataset $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}} = \{(t, \mathcal{C}_s(t), \mathbf{L}_t)\}$, where $\mathbf{L}_t$ is the label of the pivotal tuple t. The process of generating $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ will be discussed in Section 5.

RL-based Selector. To select matching tuples from a pivotal tuple, we employ the Group Relative Policy Optimization (GRPO) (DeepSeek-AI et al., 2025) to fine-tune the Selector so that it can better adapt to the dynamic changes and improve its accuracy. However, the number of positive tuples in the candidate list is very small, resulting in a continuously low reward value during the learning process of GRPO. To address this issue, we design a self-adaptive reward model.

GRPO. Given each pivotal tuple t from $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ that follows the distribution P, we adopt GRPO (DeepSeek-AI et al., 2025) with the following loss function.

$$\mathcal{J}_{\mathsf{GRPO}} = \mathbb{E}[t \sim P(\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\mathsf{old}}}(O|t)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|t)}{\pi_{\theta_{\mathsf{old}}}(o_i|t)}A_i, \mathsf{clip}\left(\frac{\pi_\theta(o_i|t)}{\pi_{\theta_{\mathsf{old}}}(o_i|t)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \eta\mathbb{D}_{\mathsf{KL}}(\pi_\theta||\pi_{\mathsf{ref}})\right)$$

where A_i is the advantage function computed within a group of rewards. The Selector $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ is fine-tuned in two stages. First, to address the cold start problem, we initialize $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ using SFT, which enables it to adapt to the selection task. Subsequently, $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ is further fine-tuned using GRPO with the $\mathcal{J}_{\mathsf{GRPO}}$ loss function.

Input and Output Formats of $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$. Given a handcrafted instruction pt_s, a tuple t, and a candidate list $\mathcal{C}_s(t)$, we first formulate them into a final prompt following (Wang et al., 2025). We then feed this prompt into the function $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$. Then $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ subsequently produces the response r.

$r =$ <positive>[...]</positive><negative>[...]</negative>

Here, the lists of positive and negative tuple IDs from $\mathcal{C}_s(t)$ are enclosed within the markers <positive> and <negative>, respectively. Here notice that we also let $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ return negative tuples to make sure that it also focuses on negative ones.

Self-adaptive Reward Function. Given a tuple t, the candidates $\mathcal{C}_s(t)$, and the label vector $\mathbf{L}_t$, we design a reward function R that returns a scalar reward value for RL, where $\mathbf{L}_t \in \{0,1\}^{|\mathcal{C}_s(t)|}$ is a binary vector indicating whether each candidate in $\mathcal{C}_s(t)$, *e.g.,* the i-th element in $\mathcal{C}_s(t)$, is a true match ($\mathbf{L}_t[i]=1$) or not ($\mathbf{L}_t[i]=0$). The reward function R contains the following steps.

(1) Step 1: Answer Extraction. We handcraft the regular expression to extract the list L_{pos} of positive tuple IDs and the list L_{neg} of negative ones from the response r of $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$. We return a zero reward if L_{pos} or L_{neg} cannot be parsed successfully from r, if the tuple IDs in L_{pos} or L_{neg} are not within the range $[1, |\mathcal{C}_s(t)|]$, or if $|L_{\mathsf{pos}}| + |L_{\mathsf{neg}}| \neq |\mathcal{C}_s(t)|$. In other words, the answer extracted from the response must be valid. If this condition is met, we proceed to Step 2; otherwise, a zero reward is returned.

Input: a collection of training data $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}} = \{(t, \mathcal{C}_s(t), \mathbf{L}_t)\}$, the number of iteration $\mathsf{iter}_{\mathsf{max}}$, the smoothing factor α.
Output: the policy π_{EM}.

Split $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ into train/valid data $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ and $\mathcal{D}^{\mathsf{S}}_{\mathsf{valid}}$;
SFT $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ in $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ as the cold start.
$\mathsf{iter} := 0, w^{(0)}_{\mathsf{pos}} = 1, w^{(0)}_{\mathsf{neg}} = 1$;
while $\mathsf{iter} \leq \mathsf{iter}_{\mathsf{max}}$ **do**
  The reward $R = \mathcal{H}_w(\mathbf{P}_t, \mathbf{L}_t, w^{(\mathsf{iter})}_{\mathsf{pos}}, w^{(\mathsf{iter})}_{\mathsf{neg}})$;
  Fine-tune $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ in $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ via GRPO using R;
  Compute the prediction $\mathbf{P}_{\mathsf{valid}} = \mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}(\mathcal{D}^{\mathsf{S}}_{\mathsf{valid}})$;
  Compute FN and FN using $\mathbf{P}_{\mathsf{valid}}$ and $\mathbf{L}_{\mathsf{valid}}$;
  $w_{\mathsf{pos}} = \frac{\mathsf{FP}+\epsilon}{\mathsf{FN}+\mathsf{FP}+\epsilon}, w_{\mathsf{neg}} = \frac{\mathsf{FN}+\epsilon}{\mathsf{FN}+\mathsf{FP}+\epsilon}$;
  $w^{(\mathsf{iter}+1)}_{\mathsf{pos}} := (1-\alpha)w^{(\mathsf{iter})}_{\mathsf{pos}} + \alpha w_{\mathsf{pos}}$;
  $w^{(\mathsf{iter}+1)}_{\mathsf{neg}} := (1-\alpha)w^{(\mathsf{iter})}_{\mathsf{neg}} + \alpha w_{\mathsf{neg}}$;
  $\mathsf{iter} := \mathsf{iter} + 1$;
return $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$;

Figure 1: RL-based Selector

(2) Step 2: Similarity reward computation. Intuitively, we aim to measure the similarity between the answer and the ground truth. Hamming similarity is a good option.

$$R(t, \mathcal{C}_s(t), \mathbf{L}_t) = \mathcal{H}\left(\mathsf{Enc}(L_{\mathsf{pos}}, L_{\mathsf{neg}}), \mathbf{L}_t\right)$$

where Enc is a handcrafted encoding function that transforms L_{pos} and L_{neg} into a binary vector $\mathbf{P}_t$ of the same dimension as $\mathbf{L}_t$. $\mathcal{H}$ represents the Hamming similarity, *s.t.* $\mathcal{H}(\mathbf{P}_t, \mathbf{L}_t) = \frac{\sum_{i=1}^{|\mathcal{C}_s(t)|}\mathbf{1}_{\mathbf{P}_t[i]=\mathbf{L}_t[i]}}{|\mathcal{C}_s(t)|}$. A higher $\mathcal{H}(\mathbf{P}_t, \mathbf{L}_t)$ indicates better performance of $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$, while a lower value suggests suboptimal performance.

However, directly using $\mathcal{H}$ has the following drawbacks. First, the ratio of positive tuples in $\mathcal{C}_s$ is very small, and the rewards from negative tuples would dominate the exploration process, causing the feedback from the reward function to remain at a very low value. Second, the goal of entity matching is to reduce both false positives (FPs) and false negatives (FNs). When the number of FPs increases, we expect $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ to focus on reducing FPs; otherwise, it should focus on reducing FNs. The current $\mathcal{H}$ does not encourage this behavior in $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$, causing it to spend a large number of iterations exploring unseen regions.

To address these issues, we design a weighted Hamming similarity $\mathcal{H}_w$:

$$\mathcal{H}_w(\mathbf{P}_t, \mathbf{L}_t, w_{\mathsf{pos}}, w_{\mathsf{neg}}) = \frac{\sum_{i=1}^{|\mathcal{C}_s(t)|}\mathbf{1}_{\mathbf{P}_t[i]=\mathbf{L}_t[i]=1}w_{\mathsf{pos}} + \mathbf{1}_{\mathbf{P}_t[i]=\mathbf{L}_t[i]=0}w_{\mathsf{neg}}}{\sum_{i=1}^{|\mathcal{C}_s(t)|}\mathbf{1}_{\mathbf{L}_t[i]=1}w_{\mathsf{pos}} + \mathbf{1}_{\mathbf{L}_t[i]=0}w_{\mathsf{neg}}}$$

For the positive tuples in $\mathbf{L}_t$, we assign a weight w_{pos}, and for the negative tuples, we assign a

weight w_{neg}. By integrating these weights into the reward function, $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ is more inclined to focus on one side, *i.e.,* either positive or negative data.

The final problem is how to set the values of w_{pos} and w_{neg}. Our idea is that if $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ has an increasing number of false positives, we should increase the value of w_{pos} so that GRPO focuses on reducing the false positives. Otherwise, we encourage GRPO to find true positives from the negative tuples. To achieve this, we split $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$ into validation data $\mathcal{D}^{\mathsf{S}}_{\mathsf{valid}}$ and compute the false positives (FPs) and false negatives (FNs) in each iteration. Let $w^{(i)}_{\mathsf{pos}}$ and $w^{(i)}_{\mathsf{neg}}$ be the weights for the i-th iteration. We use $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ to make predictions $\mathbf{P}_{\mathsf{valid}}$ on $\mathcal{D}^{\mathsf{S}}_{\mathsf{valid}}$, and then compute FPs and FNs. The current weights w_{pos} and w_{neg} are set to the percentages of FPs and FNs, respectively. However, resetting these weights in each iteration would lead to an unstable reward function. To gradually change these values, we introduce a smoothness factor α to update the weights with small adjustments. Specifically, we set $w^{(i+1)}_{\mathsf{pos}} := (1-\alpha)w^{(i)}_{\mathsf{pos}} + \alpha w_{\mathsf{pos}}$ and $w^{(i+1)}_{\mathsf{neg}} := (1-\alpha)w^{(i)}_{\mathsf{neg}} + \alpha w_{\mathsf{neg}}$.

To further reinforce the impact of positive tuples in $\mathbf{L}_t$, we incorporate semantic similarity into the reward function as prior knowledge. This guides the RL process to find a good direction. Our final reward is as follow.

$$R(t, \mathcal{C}_s(t), \mathbf{L}_t) = \mathcal{H}_w(\mathbf{P}_t, \mathbf{L}_t, w_{\mathsf{pos}}, w_{\mathsf{neg}}) + \beta \cdot \frac{1}{|S|} \sum_{s \in S} \mathsf{Sim}_{\mathsf{cos}}(\mathsf{vec}(t), \mathsf{vec}(s))$$

where β is a hyper-parameter and 0.2 by default, S is the set of true positives in the prediction of $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$, $\mathsf{Sim}_{\mathsf{cos}}$ is the cosine similarity between two vectors, and $\mathsf{vec}()$ is the embedding returned by $\mathcal{F}_{\mathsf{RAG}}$.

Figure 1 illustrates the RL process of $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$. In addition to $\mathcal{D}^{\mathsf{S}}_{\mathsf{train}}$, the number of iterations $\mathsf{iter}_{\mathsf{max}}$ and the smoothing factor α are added as inputs. Initially, we set $w^{(0)}_{\mathsf{pos}}$ and $w^{(0)}_{\mathsf{neg}}$ to 1, indicating the normal Hamming similarity (line 3). In each iteration, we re-formulate the reward function using $\mathcal{H}_w$ (line 5) and fine-tune $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ using GRPO with the reward function R (line 6). We then update the weights of positive and negative tuples using the gradual update rules (lines 7-11).

5 An ER Workflow

In this section, we present an ER workflow to train the entity matching models $\mathcal{F}_{\mathsf{EM}}$ with the assistance of an entity blocking model $\mathcal{F}_{\mathsf{RAG}}$. The workflow takes as input two relational tables, $\mathcal{R}_l$ and $\mathcal{R}_r$, and a set $\mathcal{P}$ of positive tuple pairs. As shown in Figure 2, the workflow consists of three main steps: data enrichment, entity blocking, and entity matching.

Step 1: Data enrichment. Enriching tuples in $\mathcal{R}_l$ and $\mathcal{R}_r$ with additional attributes, denoted as $\bar{B}$, is a common and effective method. Due to the uncertainty (Farquhar et al., 2024) inherent in LLMs, we observe that **the values of $\bar{B}$ imputed for a tuple can vary depending on its paired tuples (*w.r.t.* context)**. For each tuple $t \in \mathcal{R}_l$ and a set $S_t \subset \mathcal{R}_r$, we generate $|S_t|$ tuple pairs, i.e., $P_t = \{(t, s_1), \ldots, (t, s_{|S_t|})\}$, where $s_i \in S_t$. Given each pair $(t, s_i) \in P_t$, we query the LLM to impute the values of $\bar{B}$ as $a_i = \mathsf{LLM}((t, s_i), \bar{B}, \mathsf{pt}_{\mathsf{enr}})$. Due to LLM uncertainty, the imputed values $a_1, \ldots, a_{|P_t|}$ may differ across pairs. To leverage these variations, we enumerate all imputations and augment each tuple pair with multiple enriched versions.

Step 2: Entity Blocking. After data enrichment, we enrich the positive set $\mathcal{P}$ into an augmented set $\mathcal{P}_{\mathsf{enr}}$. We then fine-tune our entity blocking model $\mathcal{F}_{\mathsf{RAG}}$ using contrastive learning with a randomly sampled negative set, following the approach in (Wang et al., 2024a). The final output of this step is the fine-tuned $\mathcal{F}_{\mathsf{RAG}}$.

Step 3: An iterative EM workflow. Given $\mathcal{F}_{\mathsf{RAG}}$, $\mathcal{P}_{\mathsf{enr}}$, $\mathcal{R}_l$ and $\mathcal{R}_r$, we propose a progressive training workflow that fine-tunes $\mathcal{F}^{\mathsf{M}}_{\mathsf{EM}}$ and $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$.

Overview. We show the EM workflow. Given a tuple $t \in \mathcal{R}_l$, $\mathcal{F}_{\mathsf{RAG}}$ conducts similarity search by retrieving its K nearest neighbors $\mathsf{NN}_K(t)$, which forms the candidate list for t. We define two pointers: ptr_s and ptr_e, where ptr_s indicates the boundary separating positive tuples from the rest, such that all tuples in the range $[1, \mathsf{ptr}_s]$ are considered positive, while tuples in the range $[\mathsf{ptr}_e, K]$ are considered negative. Specifically, $(t, \mathsf{NN}_K(t)[i])$ are treated as positive tuple pairs for $i \in [1, \mathsf{ptr}_s]$ and $(t, \mathsf{NN}_K(t)[j])$ are negative ones for $j \in [\mathsf{ptr}_e, K]$. $[\mathsf{ptr}_s, \mathsf{ptr}_e]$ are ambiguous pairs.

In the beginning of the training procedure, ptr_s is set to 1 and ptr_e is set to K, and $\mathcal{F}_{\mathsf{RAG}}$ generates the potentially positive and negative tuple pairs $\mathcal{P}_{\mathsf{RAG}}$ and $\mathcal{N}_{\mathsf{RAG}}$ within $[1, \mathsf{ptr}_s]$ and $[\mathsf{ptr}_e, K]$, respectively. Next $\mathcal{F}_{\mathsf{RAG}}$ sends them to $\mathcal{F}_{\mathsf{EM}}$, which processes them using the co-training strategy. In the next iteration, $\mathcal{F}_{\mathsf{RAG}}$ retrieves the new $\mathsf{NN}_K(t)$ for each tuple t and adjust ptr_s and ptr_e by a step size δ, updating the pointers as $\mathsf{ptr}_s = \mathsf{ptr}_s + \delta$ and

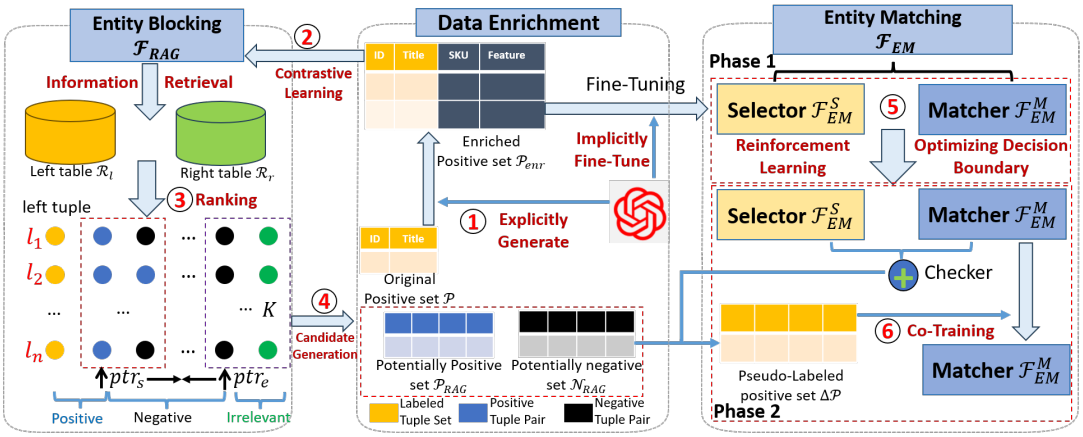

Figure 2: The end-to-end entity resolution workflow

$\text{ptr}_e = \text{ptr}_e - \delta$. The iterative process continues until ptr_s is no longer less than ptr_e.

Co-training strategy. Given potentially positive tuple pairs $\mathcal{P}_{\text{RAG}}$ and negative tuple pairs $\mathcal{N}_{\text{RAG}}$, and the augmented set $\mathcal{P}_{\text{enr}}$, we simultaneously learn Matcher $\mathcal{F}^{\text{M}}_{\text{EM}}$ and Selector $\mathcal{F}^{\text{S}}_{\text{EM}}$. Considering the extremely low ratio of positive to negative tuple pairs in $\mathcal{R}_l$ and $\mathcal{R}_r$, and the fact that existing methods, *e.g.,* (Thirumuruganathan et al., 2021), reply on random sampling for negative tuple pairs, we assume that $\mathcal{N}_{\text{RAG}}$ are more likely to be correct in the first few iteration. Thus, we introduce a warmup period during which $\mathcal{N}_{\text{RAG}}$ are initially treated as ground truth negatives and combined with $\mathcal{P}_{\text{enr}}$ to fine-tune $\mathcal{F}^{\text{M}}_{\text{EM}}$ for the first λ iterations. As ptr_e approaches ptr_s after λ iterations, $\mathcal{F}^{\text{M}}_{\text{EM}}$ is then responsible for selecting which negative tuple pairs should be included in the training data.

Specifically, we design a two-phase learning method to simultaneously train $\mathcal{F}^{\text{M}}_{\text{EM}}$ and $\mathcal{F}^{\text{S}}_{\text{EM}}$.

Phase 1. In the first phase, we add $\mathcal{P}_{\text{enr}}$ into the training data $\mathcal{D}_{\text{train}}$. If the current iteration is less than λ, we generate the training data $\mathcal{D}_{\text{train}}$ as $\mathcal{D}_{\text{train}} = \mathcal{P}_{\text{enr}} \cup \mathcal{N}_{\text{RAG}}$. For iterations beyond λ, we have a checker step by using $\mathcal{F}^{\text{M}}_{\text{EM}}$ to verify whether the labels of tuple pairs in $\mathcal{N}_{\text{RAG}}$ are consistent with its predictions, The training data $\mathcal{D}_{\text{train}}$ is then updated to $\mathcal{D}_{\text{train}} = \mathcal{P}_{\text{enr}} \cup \{(t,s) | \mathcal{F}^{\text{M}}_{\text{EM}}(t,s) = \text{No}, (t,s) \in \mathcal{N}_{\text{RAG}}\}$.

After generating $\mathcal{D}_{\text{train}}$, we proceed to generate the training data $\mathcal{D}^{\text{S}}_{\text{train}}$ for the Selector. For each pivot tuple t, we retrieve all tuple pairs $(t, s_1), \ldots, (t, s_L)$ from $\mathcal{D}_{\text{train}}$ where t appears on the left-hand side of the tuple pairs. We then define $\mathcal{C}_s(t) = \{s_1, \ldots, s_L\}$, and the label $\mathbf{L}_t$ is an L-dimensional vector, where $\mathbf{L}_t[i] = 1$ if (t, s_i) is a matched tuple, and $\mathbf{L}_t[i] = 0$ otherwise. We generate a triplet $(t, \mathcal{C}_s(t), \mathbf{L}_t)$ as an element of $\mathcal{D}^{\text{S}}_{\text{train}}$. Finally, we fine-tune $\mathcal{F}^{\text{M}}_{\text{EM}}$ using SFT on $\mathcal{D}_{\text{train}}$ and $\mathcal{F}^{\text{S}}_{\text{EM}}$ using GRPO on $\mathcal{D}^{\text{S}}_{\text{train}}$, respectively.

Phase 2. Once $\mathcal{F}^{\text{S}}_{\text{EM}}$ and $\mathcal{F}^{\text{M}}_{\text{RM}}$ are well fine-tuned, we adopt $\mathcal{F}^{\text{S}}_{\text{EM}}$ to generate more positive training data with pseudo-labels. For each tuple $t \in \mathcal{R}_l$, we first generate m augmented tuples, denoted by $t_1, \ldots, t_m$, and query $\mathcal{F}^{\text{S}}_{\text{EM}}$ with the list $\mathcal{C}_s(t_i)$ for t_i for $i \in [1, m]$. This process yields m lists of positive tuples $R_1, \ldots, R_m$. To enhance the accuracy of these positive instances, we use $\mathcal{F}^{\text{M}}_{\text{EM}}$ to make inferences, obtaining additional positive training data as $\Delta\mathcal{P} = \{(t,s) | \mathcal{F}^{\text{M}}_{\text{EM}}(t,s) = \text{Yes}, (t,s) \in R_1 \cup \cdots \cup R_m\}$. Finally, we incorporate $\Delta\mathcal{P}$ into $\mathcal{D}_{\text{train}}$, and $\mathcal{F}^{\text{M}}_{\text{EM}}$ is continuously fine-tuned using SFT to further enhance its performance.

The ER workflow that relies on RL training can be time-consuming; nevertheless, they are usually carried out in an offline preprocessing phase. For example, deduplication of corpora executed offline ensures that search engines do not return duplicate or valueless results when an online query is given. Similarly, when a new data source is introduced, data integration is usually executed offline, allowing the newly integrated data to be used for downstream online business processes. Consequently, users mainly focus on the accuracy of detecting duplicates, and the workflow remains worthwhile as long as it does not introduce prohibitive delays.

6 Experimental Results

In this section, we empirically evaluated our method, PUER using benchmark datasets on (1) the effectiveness and efficiency of entity matching, (2) the ablation study, and (3) a comparative analysis with online LLMs, particular ComEM (Wang et al., 2025) with GPT-4o-mini. More experimental

results are provided in the supplemental material.

Experimental settings. We start with our settings.

Datasets. We conducted experiments using 11 benchmark datasets from the ER Benchmark datasets (Köpcke et al., 2010), the Magellan data repository (The Magellan Data Repository) and WDC product data corpus (Primpeli et al., 2019) used for evaluating Ditto (Li et al., 2020b), These datasets include Amazon-Google (AG), Walmart-Amazon (WA), Abt-Buy (AB), DBLP-ACM (DA), DBLP-Scholar (DS), Company (CO), Cameras (CA), Computers (COM), Shoes (SH), Watch (WAT) and WDC-All-Small (WS). Following Ditto, we randomly sample 50 positive tuple pairs as labeled training data and retained the left and right relational tables as all unlabeled data. The statistics of all datasets are summarized in the supplementary material.

Baselines. We implemented PUER in Python and used the following baselines. (1) Ditto (Li et al., 2020b), an entity matching model based on BERT; (2) Rotom (Miao et al., 2021), an entity matching model leveraging language models and data augmentation through RL; (3) PromptEM (Wang et al., 2022), a prompt-tuning model based on pretrained language models; (4) Unicorn (Fan et al., 2024a), a multi-task data matching model using a mixture of experts; (5) CLER (Wu et al., 2023), a low-resource entity resolution model that integrates entity blocking and matching; (6) JellyFish (Zhang et al., 2024), an LLM based entity matching model using LoRA-based instruction-tuning; (7) Sudowoodo (Wang et al., 2023), an entity resolution framework based on contrastive representation learning. We also compared with the following online LLMs: (8) BatchER (Fan et al., 2024b), a cost-effective batch prompting to ER based on online LLMs, and (9) ComEM (Wang et al., 2025), an LLM-based ER model using Matcher, Comparer and Selecter.

For a fair comparison, all baselines are provided with the same set of 50 positive tuple pairs (denoted as $\mathcal{P}$), all labeled negative pairs from the training set of the benchmark, and the same relational left and right tables. In contrast, PUER is provided with $\mathcal{P}$ and relational left and right tables but without the labeled negative pairs, representing a few-shot PU (positive-unlabeled) setting. Notably, Unicorn, and JellyFish were also pretrained on additional labeled entity matching corpus.

Measures. We report precision (P), recall (R) and F1 (F) score for entity matching following (Li et al., 2020b). All results are reported in 100-scale.

Configuration. We select Qwen-2.5-7B-instruct (Yang et al., 2024) as the backbone of $\mathcal{F}_{\mathsf{EM}}$ and `bge-large-en` as the pre-trained model for $\mathcal{F}_{\mathsf{RAG}}$. We set K as 20, δ as 5,and τ as 0.02 by default and adopt the AdamW optimizer with the learning rate of 1e-4 and 1e-5 for $\mathcal{F}^{\mathsf{M}}_{\mathsf{EM}}$ and $\mathcal{F}_{\mathsf{RAG}}$, respectively. In GRPO of $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$, we set the training batch size as 16, the length of input prompt as 1024, the maximum output length as 64, the mini-batch as 16, the learning rate of the actor model as 1e-6, and the coefficient η of KL loss as 0.001. We adopt verl (Sheng et al., 2025), a RL training framework to fine-tune $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ and remains other hyper-parameters by default. For all baselines, we use their default settings. We conduct our experiment on a single machine powered by 1.5TB RAM and 128 processors with Intel(R) Xeon(R) Platinum 8358 CPU @2.60GHz and 4 NVIDIA A800 GPUs. Each experiment was conducted twice, averaging the results reported here.

Experimental results. We next report our findings.

Exp-1: Entity matching. We evaluate the effectiveness of PUER in comparison to other baselines with aspect to entity matching. Table 1 shows the performance of all baselines. PUER consistently outperforms all other baselines across all 11 datasets in terms of precision, recall and F1-score, achieving average improvements of 26.31%, 36.87% and 40.19%, respectively, and up to 63.21%, 50.09% and 53.42%. This verifies that the co-training strategy between $\mathcal{F}^{\mathsf{S}}_{\mathsf{EM}}$ and $\mathcal{F}^{\mathsf{M}}_{\mathsf{EM}}$ and interaction between $\mathcal{F}_{\mathsf{EM}}$ and $\mathcal{F}_{\mathsf{RAG}}$ are effective, where the Selector and the RAG blocker enhance the performance of the entity matching component. Furthermore, PUER exhibits greater robustness compared to other baselines and is less not sensitive with data distribution. Specifically, PUER shows superior performance in 10 out of 11 datasets in terms of F1-score, *e.g.,* at least 23.01%, 14.82% and 47.74% improvement in WS, DS and Company dataset across different domains. This highlights the stability of PUER and its effectiveness independent of specific data distribution.

Compared with pre-trained baselines, PUER also outperforms JellyFish, an LLM-based entity matching model using handcraft prompts and LoRA tuning, *e.g.,* 16.63% F1-score improvement

Datasets	PUER (Ours)			Ditto			Rotom			Unicorn			PromptEM			JellyFish			Sudowoodo		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
AG	84.83	76.49	**80.45**	39.28	4.70	8.39	19.50	59.01	29.30	90.66	11.53	20.14	64.62	17.95	28.09	92.03	44.44	59.94	55.85	52.99	54.88
WA	93.95	88.60	**91.20**	78.43	20.72	32.78	11.80	73.10	20.30	89.99	60.62	72.44	93.55	30.05	45.49	80.09	93.78	86.39	46.41	50.25	48.25
AB	90.04	87.86	**88.94**	97.24	51.45	67.3	14.60	42.70	21.70	97.11	49.02	65.16	98.04	48.54	64.94	99.38	78.15	87.5	42.30	32.03	36.46
DA	95.47	99.77	97.57	99.5	89.63	94.31	80.9	97.1	88.2	99.29	95.27	97.24	100	86.04	92.49	99.76	97.07	**98.40**	78.55	98.19	87.28
DS	99.31	94.85	**97.03**	98.2	30.65	46.72	65.6	94.7	77.5	98.81	70.18	82.07	98.73	58.04	73.1	99.7	64.01	77.97	73.33	93.55	82.21
CO	98.56	79.18	**87.82**	25.06	100	40.08	n/a	n/a	n/a	87.49	3.84	7.37	n/a	n/a	n/a	96.43	22.07	35.92	30.62	25.28	27.69
CA	93.20	100.00	**96.48**	70.53	27.43	39.5	40.1	35.8	37.8	96.29	36.11	52.52	89.02	25.35	39.46	96.07	68.05	79.67	57.58	44.79	50.39
COM	98.67	99.33	**99.00**	65.64	28.76	39.99	29.3	50.2	37	92.82	69.23	79.31	80.57	74.58	77.57	97.5	78.26	86.82	37.08	48.49	42.02
SH	98.48	88.13	**93.02**	74.7	43.05	54.62	26.7	55.9	36.1	80.47	58.64	67.84	50.00	1.00	1.97	94.44	51.86	66.95	34.38	36.94	35.62
WAT	97.89	85.03	**91.01**	27.36	27.09	27.22	26.9	65.9	38.2	93.71	49.83	65.06	42.86	1.00	1.96	89.45	77.37	82.97	31.49	51.50	39.08
WS	87.67	95.57	**91.45**	71.89	20.28	31.64	27.4	75.00	40.2	95.97	43.73	60.09	91.55	28.05	42.94	96.64	52.92	68.39	40.99	63.10	49.70
Average	94.37	90.43	**92.17**	67.98	40.34	43.87	31.16	59.04	38.75	92.96	49.82	60.84	73.54	51.69	42.55	94.68	66.18	75.54	48.05	54.28	50.33

Table 1: Entity matching performance in comparison to baselines, n/a means the method is terminated within 10 hours

Methods/Model	AB	AG	DA	DS	WA
PUER (Ours)	**88.94**	**80.45**	**97.57**	**97.03**	**91.20**
CLER	75.86	47.56	80.04	55.96	70.02
BatchER (GPT-4)	85.22	64.06	96.04	89.48	81.22
ComEM (GPT-3.5-turbo)	87.62	69.63	90.85	84.68	86.37
ComEM (GPT-4o-mini)	88.24	71.47	90.58	87.84	88.56

Table 2: Comparison with Online Model (F1-score)

Datasets	AG		AB		WA	
	Train	Predict	Train	Predict	Train	Predict
Ditto	235	23	208	20	215	22
Rotom	522	23	435	20	487	23
Unicorn	725	17	602	13	654	14
PromptEM	1420	65	1533	42	1365	55
JellyFish	1243	45	1010	30	1190	42
PUER (Ours)	3561	208	4323	255	4555	339

Table 3: The Efficiency of Entity Matching (in seconds)

on average. This underscores the effectiveness of our end-to-end iterative framework. While Unicorn achieves relatively good performance among the baselines due to its mixture-of-expert architecture, it struggles to attain high recall because of the small set of positive instances.

To evaluate the ER framework that integrates entity blocking and matching, we compare with CLER and Sudowoodo in Table 1 and 2. PUER is 10.85%, 29.73% and 23.63% more accuracy in aspects of precision, recall and F1-score than CLER on average (Table 2), which indicates that the proposed workflow that interacts $\mathcal{F}_{\mathsf{RAG}}$ and $\mathcal{F}_{\mathsf{EM}}$ could be beneficial to both of them. Due to the few-shot PU setting, Sudowoodo struggles to generate high-quality pseudo-labels relying solely on similarity threshold and positive ratio, and the number of high-quality pseudo-labels it generates is insufficient (Table 1). In contrast, PUER employs the GRPO algorithm with a carefully designed reward function, which enables it to tolerate noisy data with better generalization ability.

In Table 2, we compared PUER with the online BatchER (Fan et al., 2024b) and ComEM (Wang et al., 2025) using GPT-3.5, GPT-4 and GPT-4o-mini as the backbones. The result shows that PUER achieves up to 15.5% higher F1-score, indicating the effectiveness of the fine-tuning workflow.

Methods	AG			WA			AB		
	P	R	F	P	R	F	P	R	F
PUER	84.83	76.49	**80.45**	93.95	88.60	**91.20**	90.04	87.86	88.94
w.o. Selector	33.43	97.86	49.83	11.42	100.00	20.51	42.20	94.66	58.38
w.o. enrich	65.92	76.06	70.63	81.42	88.60	84.86	88.62	90.77	**89.68**
w.o. co-train	63.66	84.61	72.66	75.73	93.78	83.79	85.30	87.37	86.33

Table 4: Ablation Study for Entity Matching

Exp-2: Efficiency of entity matching. We present the fine-tuning time (Train) and inference time (Predict) of $\mathcal{F}_{\mathsf{EM}}$ in Table 3. Since $\mathcal{F}_{\mathsf{EM}}$ employs co-training of the Selector and Matcher, as well as an iterative workflow to gradually generate more training data with pseudo-labels, PUER requires significantly more time to fine-tune its entity matching models and perform inferences compared to other methods. Although PUER is notably slower, its high accuracy in entity matching, as shown in Table 1, justifies the use of RL and co-training strategy despite the increased computational cost.

The training cost of PUER is bounded by the number of training data. Despite its higher training cost, it does not result in prohibitively high expenses in few-shot setting. Considering the significant improvement in the F1-score of PUER (e.g., at least 16.63% F1 improvement over baselines) and the unique background of the positive-unlabeled setting and ER task, the higher cost can be tolerated and is relatively negligible.

Exp-3: Quality of pseudo-labels. To further demonstrate the effectiveness of our pseudo-labeling mechanism in PUER, we conducted experiments to measure the precision and recall of pseudo-positive and pseudo-negative examples under varying values of K in Table 5 following (Wang et al., 2023). Notice that we discard tuple pairs of pseudo-labels that are not included in the ground truths provided by benchmarks so that the evaluations are accurate and totally based on benchmarks.

By varying K from 5 to 9, the average precision and recall are 88.28% and 81.53%, respec-

Dataset	K=5		K=7		K=9	
	P	R	P	R	P	R
AG	78.33	67.52	77.24	67.52	76.17	67.09
WA	95.04	87.63	95.23	87.21	95.05	87.83
AB	92.96	89.88	93.01	89.30	91.47	89.78

Table 5: Quality of pseudo-labels under different K

tively. These results indicate the high quality of our generated pseudo-labels. Furthermore, as K increases, the quality of the pseudo-labels only decreases slightly. Although it may involve more tuples for checking, our PUER is robust enough to generate correct pseudo-labels, which is critical for the further fine-tuning of the selector and matcher.

Exp-4: Flexibility of PUER. Given that our ER framework does not rely on the matcher, we adopted the EM model in Unicorn (Fan et al., 2024a), a smaller model as the matcher of PUER. Table 6 and 7 present the effectiveness and efficiency of $\text{PUER}_{\text{unicorn}}$, which utilizes the EM model of Unicorn as the matcher, respectively.

Dataset	PUER			$\text{PUER}_{\text{unicorn}}$		
	P	R	F	P	R	F
AG	84.83	76.49	80.45	80.18	76.07	78.07
WA	93.95	88.60	91.20	76.19	82.90	79.40
AB	90.04	87.86	88.94	90.28	76.69	82.93

Table 6: Effectiveness of PUER v.s. $\text{PUER}_{\text{unicorn}}$.

	Train		Predict	
Dataset	PUER	$\text{PUER}_{\text{unicorn}}$	PUER	$\text{PUER}_{\text{unicorn}}$
AG	3561	3027 (-534)	208	160 (-48)
WA	4555	3761 (-794)	339	205 (-134)
AB	4323	3618 (-705)	255	202 (-53)

Table 7: Efficiency of $\text{PUER}_{\text{unicon}}$ (in seconds).

As evidenced by Table 6 and 7, the training time is reduced to approx. 80% of the original time, but the accuracy remains largely unaffected, which indicates the effectiveness of the PUER framework, regardless of whether the Matcher is a powerful LLM or a simpler model. Under the few-shot PU setting, considering the superior accuracy achieved, the training time is deemed acceptable.

Exp-5: Ablation study. In Table 4, we present the ablation study of PUER and its variants, namely PUER without the Selector, without enrichment and PUER without co-training that the selector and matcher are trained independently. The results demonstrate that each variant achieves lower accuracy compared to the complete PUER model.

Specifically, PUER without enrichment and without the Matcher shows only a relatively small performance drop. This suggests that LLMs already possess substantial prior knowledge, and the Matcher does not contribute much in the few-shot data setting. In contrast, PUER without the Selector experiences a significant drop in accuracy, such as a 31% decrease on the AG dataset. This finding indicates that the RL component in the Selector plays a crucial role in generalization, particularly in few-shot scenarios. It enables the model to collect sufficient data from the environment to achieve robust performance.

To determine whether the pre-training process of the LLM backbone includes the benchmark datasets, we conducted an experiment of the few-shot prompting using Qwen-2.5-7B-instruct.

	CO	WA	DS	AB	AG
P	99.35	74.26	92.02	77.63	46.74
R	10.92	52.33	62.52	60.67	82.90
F	19.68 (–68.14)	61.40 (–29.80)	74.45 (–22.58)	68.11 (–20.83)	59.78 (–20.67)

Table 8: The effectivenss of Qwen-2.5-7B-instruct (P/R/F)

As illustrated in Table 8, the F1-score of PUER is at least 20.67 points higher than that of few-shot prompting and 68.14 points at most, demonstrating that only LLM backbone is not sufficient and PUER is necessary to make more accurate predictions.

Dataset	Methods	$\lvert\mathcal{P}\rvert$ =10	20	30	40	50	100
AB	PUER	**55.15**	**74.01**	**73.11**	**79.49**	**88.92**	**89.26**
	Unicorn	48.17	49.08	59.66	71.42	65.16	84.57
	DITTO	17.39	21.58	34.04	45.66	67.3	82.35
WS	PUER	**65.82**	**68.96**	**65.01**	**82.29**	**91.45**	**92.12**
	Unicorn	23.12	35.02	65.22	54.24	40.20	70.19
	DITTO	13.25	20.57	27.45	35.69	31.64	66.10

Table 9: Performance Vary Labelling Budget $\lvert\mathcal{P}\rvert$ (F1)

Exp-6: Hyper-parameter study. We vary the labeling budget $\lvert\mathcal{P}\rvert$ from 10 to 50 in Table 9. PUER demonstrates robustness with respect to the number of positive tuples, *e.g.,* only 6.9% drop when $\lvert\mathcal{P}\rvert$ decreases from 50 to 10 in WS.

7 Conclusion

In this paper, we propose, PUER, an end-to-end ER solution for few-shot PU learning. We adopt the reinforcement learning method to solve the entity matching task, and design a self-adaptive reward function. Furthermore, we introduce an iterative training workflow that fully utilizes the entity blocking model to assist the entity matching via a co-training mechanism. Finally comprehensive experiments across 11 benchmarks demonstrate the superior performance of PUER.

Limitations

Our work, PUER, introduces an end-to-end entity resolution solution tailored for few-shot positive-unlabeled (PU) learning scenarios by leveraging Large Language Models (LLMs) and reinforcement learning. We propose an iterative co-training mechanism that integrates entity blocking and entity matching, including a novel self-adaptive reward function for the reinforcement learning component, to enhance performance with minimal labeled positive data.

Despite the promising results, our approach has several limitations. Firstly, the reinforcement learning (RL) component, particularly the Selector fine-tuned with Group Relative Policy Optimization (GRPO), inherently introduces a higher level of complexity in terms of training and hyperparameter tuning compared to simpler supervised methods. Secondly, while LLMs offer powerful generative capabilities, the quality and consistency of the generated outputs (e.g., enriched attributes or pseudo-labels) can be uncertain and may occasionally require careful validation. Lastly, as indicated by our efficiency experiments, the proposed PUER framework, with its iterative workflow and co-training of multiple components including RL-based Selector and Matcher, exhibits a higher computational complexity during both training and inference compared to some traditional entity resolution methods. This increased cost is a trade-off for the achieved accuracy in low-resource settings.

Ethics Statement

The experiments were conducted on publicly available benchmark datasets and models, eliminating any data privacy concerns. To the best of our knowledge, there is no negative societal impact in this research.

Acknowledgments

This work was supported by Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No.2023B1212010007, SL2023A03J00934), Guangzhou Municipal Science and Technology Project (No. 2023A03J0003, 2023A03J0013 and 2024A03J0621).

References

Naser Ahmadi, Hansjörg Sand, and Paolo Papotti. 2022. Unsupervised matching of data and text. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 1058–1070. IEEE.

Arvind Arasu, Michaela Götz, and Raghav Kaushik. 2010. On active learning of record matching packages. In *SIGMOD*, pages 783–794.

Jessa Bekker and Jesse Davis. 2020. Learning from positive and unlabeled data: a survey. *Mach. Learn.*

Mikhail Bilenko and Raymond J. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*.

Alexander Brinkmann, Roee Shraga, and Christina Bizer. 2024. Sc-block: Supervised contrastive blocking within entity resolution pipelines. In *ESWC*.

Paul Suganthan G. C., Adel Ardalan, AnHai Doan, and Aditya Akella. 2018. Smurf: Self-service string matching using random forests. *Proc. VLDB Endow.*

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *PVLDB*, 16(8):1944–1957.

Vasilis Efthymiou, George Papadakis, George Papastefanatos, Kostas Stefanidis, and Themis Palpanas.

2015. Parallel meta-blocking: Realizing scalable entity resolution over large, heterogeneous data. In *IEEE BigData*.

Ju Fan, Jianhong Tu, Guoliang Li, Peng Wang, Xiaoyong Du, Xiaofeng Jia, Song Gao, and Nan Tang. 2024a. Unicorn: A unified multi-tasking matching model. *SIGMOD Rec.*

Meihao Fan, Xiaoyue Han, Ju Fan, Chengliang Chai, Nan Tang, Guoliang Li, and Xiaoyong Du. 2024b. Cost-effective in-context learning for entity resolution: A design space exploration. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*.

Wenfei Fan, Hong Gao, Xibei Jia, Jianzhong Li, and Shuai Ma. 2011. Dynamic constraints for record matching. *VLDB J.*, 20(4):495–520.

Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about record matching rules. *PVLDB*, 2(1):407–418.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. 2024. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630(8017):625–630.

Cheng Fu, Xianpei Han, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 2019. End-to-end multi-perspective matching for entity resolution. In *IJCAI*, pages 4961–4967.

Songtao Guo, Xin Luna Dong, Divesh Srivastava, and Remi Zajac. 2010. Record linkage with uniqueness constraints and erroneous values. *PVLDB*, 3(1):417–428.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *ACL*, pages 5851–5861.

Mayank Kejriwal and Daniel P. Miranker. 2015. A DNF blocking scheme learner for heterogeneous datasets. *CoRR*, abs/1501.01694.

Nishadi Kirielle, Peter Christen, and Thilina Ranbaduge. 2022. Transer: Homogeneous transfer learning for entity resolution. In *EDBT*.

Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward building entity matching management systems. *PVLDB*, 9(12):1197–1208.

Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.*, 3(1):484–493.

Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for bert-based entity resolution. In *AAAI*.

Bing Li, Wei Wang, Yifang Sun, Linhan Zhang, Muhammad Asif Ali, and Yi Wang. 2020a. Grapher: Token-centric entity resolution with graph convolutional neural networks. In *AAAI*, pages 8172–8179.

Huahang Li, Shuangyin Li, Fei Hao, Chen Jason Zhang, Yuanfeng Song, and Lei Chen. 2024a. Booster: leveraging large language models for enhancing entity resolution. In *WWW*, pages 1043–1046.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024b. Table-gpt: Table fine-tuned GPT for diverse table tasks. *Proc. ACM Manag. Data*.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020b. Deep entity matching with pre-trained language models. *PVLDB*, 14(1):50–60.

Michael Loster, Ioannis K. Koumarelas, and Felix Naumann. 2021. Knowledge transfer for entity resolution with siamese neural networks. *ACM J. Data Inf. Qual.*

Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A comprehensive benchmark framework for active learning methods in entity matching. In *SIGMOD*.

Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In *SIGMOD*, pages 1303–1316. ACM.

Matthew Michelson and Craig A. Knoblock. 2006. Learning blocking schemes for record linkage. In *AAAI*.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *SIGMOD*, pages 19–34.

Youcef Nafa, Qun Chen, Zhaoqiang Chen, Xingyu Lu, Haiyang He, Tianyi Duan, and Zhanhuai Li. 2022. Active deep learning on entity resolution by risk sampling. *Knowl. Based Syst.*

Gang Niu, Marthinus Christoffel du Plessis, Tomoya Sakai, Yao Ma, and Masashi Sugiyama. 2016. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *NeurIPS*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

George Papadakis, Georgia Koutrika, Themis Palpanas, and Wolfgang Nejdl. 2014. Meta-blocking: Taking entity resolutionto the next level. *IEEE Trans. Knowl. Data Eng.*

George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42.

Derek Paulsen, Yash Govind, and AnHai Doan. 2023. Sparkly: A simple yet surprisingly strong TF/IDF blocker for entity matching. *Proc. VLDB Endow.*

Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC training dataset and gold standard for large-scale product matching. In *WWW*.

Kun Qian, Lucian Popa, and Prithviraj Sen. 2017. Active learning for large-scale entity resolution. In *CIKM*, pages 1379–1388.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*.

Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017a. Synthesizing entity matching rules by examples. *Proc. VLDB Endow.*

Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed K. Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017b. Synthesizing entity matching rules by examples. *PVLDB*.

Chenchen Sun, Yang Xu, Derong Shen, and Tiezheng Nie. 2024. Matching feature separation network for domain adaptation in entity matching. In *WWW*, pages 1975–1985. ACM.

The Magellan Data Repository. Sanjib das, anhai doan, paul suganthan g. c., chaitanya gokhale, pradap konda, yash govind, and derek paulsen. https://sites.google.com/site/anhaidgroup/projects/data.

Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: A design space exploration. *Proc. VLDB Endow.*, 14(11):2459–2472.

Jianhong Tu, Xiaoyue Han, Ju Fan, Nan Tang, Chengliang Chai, Guoliang Li, and Xiaoyong Du. 2022. DADER: hands-off entity resolution with domain adaptation. *PVLDB*, 15(12):3666–3669.

Somin Wadhwa, Adit Krishnan, Runhui Wang, Byron C. Wallace, and Luyang Kong. 2024. Learning from natural language explanations for generalizable entity matching. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*. Association for Computational Linguistics.

Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. Promptem: Prompt-tuning for low-resource generalized entity matching. *PVLDB*.

Runhui Wang, Yuliang Li, and Jin Wang. 2023. Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation. In *ICDE*.

Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xuanang Chen, Xianpei Han, Le Sun, Hao Wang, and Zhenyu Zeng. 2025. Match, compare, or select? an investigation of large language models for entity matching. In *ACL*, pages 96–109.

Tianshu Wang, Hongyu Lin, Xianpei Han, Xiaoyang Chen, Boxi Cao, and Le Sun. 2024a. Towards universal dense blocking for entity resolution. *CoRR*, abs/2404.14831.

Ye Wang, Huazheng Pan, Tao Zhang, Wen Wu, and Wenxin Hu. 2024b. A positive-unlabeled metric learning framework for document-level relation extraction with incomplete labeling. In *AAAI*.

Steven Euijong Whang and Hector Garcia-Molina. 2013. Joint entity resolution on multiple datasets. *VLDB J.*, 22(6):773–795.

Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity resolution using zero labeled examples. In *SIGMOD*, pages 1149–1164.

Shiwen Wu, Qiyu Wu, Honghua Dong, Wen Hua, and Xiaofang Zhou. 2023. Blocker and matcher can mutually benefit: A co-learning framework for low-resource entity resolution. *Proc. VLDB Endow.*

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Zijun Yao, Chengjiang Li, Tiansi Dong, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Yichi Zhang, and Zelin Dai. 2021. Interpretable and low-resource entity matching via decoupling feature learning from decision making. In *Proceedings of the 59th Annual Meeting of*

the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021.

Xiaocan Zeng, Pengfei Wang, Yuren Mao, Lu Chen, Xiaoze Liu, and Yunjun Gao. 2024. Multiem: Efficient and effective unsupervised multi-table entity matching. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*.

Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024. Jellyfish: Instruction-tuning local large language models for data preprocessing. In *EMNLP*, pages 8754–8782, Miami, Florida, USA. Association for Computational Linguistics.

Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *WWW*, pages 2413–2424.