# Unsupervised Domain Adaptation for Entity Blocking Leveraging Large Language Models

Yaoshu Wang
*Fundamental Research Department*
*Shenzhen Institute of Computing Sciences*
Shenzhen, China
yaoshuw@sics.ac.cn

Mengyi Yan
*Computer Science Department*
*Beihang University*
Beijing, China
yanmy@act.buaa.edu.cn

2024 IEEE International Conference on Big Data (BigData) | 979-8-3503-6248-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/BigData62323.2024.10825234

***Abstract*—Entity blocking, which aims to find all potentially matched tuple pairs in large-scale data, is an important step for entity resolution. It is non-trivial because it needs to consider both of the effectiveness and efficiency, and the emergence of representation learning has made it possible. Although there exist existing representation learning models for entity blocking, all of them require self-curated training instances in the target domain, which limits their capabilities for unseen data. In this paper, we propose UDAEB, a framework for Unsupervised Domain Adaptation for Entity Blocking that is fine-tuned between the source and target domains using contrastive learning by leveraging the capabilities of LLMs. UDAEB first adopts the adversarial learning strategy to reduce the distribution discrepency between source and target domains as the warmup step. Based on the initially learned representations, UDAEB involves pre-trained LLMs to enrich robust and distinguishable attributes for source and target domains. Furthermore, we propose an iterative step to fine-tune entity blocking model by selecting high-quality training instances with pseudo-labels by leveraging LLMs. Finally we conduct comprehensive experiments to show UDAEB has the superior performance against the state-of-the-art algorithms with aspects of the pair completeness (PC), pair quality (PQ) and the candidate set size ratio (CSSR).**

***Index Terms*—Entity Blocking, Unsupervised Domain Adaptation, Large Language Models**

## I. Introduction

Entity resolution (ER), also called de-duplication and record linkage, aims to retrieve and identify all matched tuple pairs in collections of tuples. It is an important component for data cleaning, information integration, and data processing pipelines for training machine learning models. When encountering large number of tuples, entity resolution often adopts entity blocking as the filtering step to filter unmatched tuple pairs so that the Cartesian product operation can be avoided. Thus, entity blocking, as a vital step of ER, needs to achieve (1) fast speed so that candidate tuple pairs can be efficiently retrieved; (2) high recall so that no matched pairs are missed; and (3) high precision so that the following entity matching step does not need to evaluate a large number of candidates.

With the emergence of representation learning based on neural networks, a promising research direction for entity blocking has been developed. All tuples are transformed into dense embeddings, and $K$ nearest neighbor search is executed to retrieve the top-$K$ most similar ones for each tuple, where $K$ is a pre-defined hyper-parameter.

Due to the insufficient manual annotations, recent studies have shown that self-supervised learning has significantly improved entity blocking. Most existing entity blocking models, such as DeepBlocker [1] and Sudowoodo [2], automatically generate training tuple pairs using data augmentation techniques, where positive tuples are generated from one tuple by randomly inserting, deleting, or replacing its tokens. However, none of the existing works consider using data from other domains to enhance their performance, i.e., unsupervised domain adaptation. Although there are works like DADER [3] and MFSN [4] that propose cross-domain entity resolution, they all focus on entity matching, a binary classification task, which is different from entity blocking, *i.e.,* a ranking task. Furthermore, as Large Language Models (LLMs) have recently shown significant performance improvements, entity matching based on LLMs, such as JellyFish [5] and MELD [6], have been proposed. However, there are no works that leverage the background knowledge of LLMs to enhance entity blocking.

In this paper, we design an unsupervised domain adaptation framework for entity blocking (UDAEB), leveraging the capabilities of LLMs. We address this by proposing a framework consisting of three steps: warmup, enrichment, and iteration. The warmup step aims to initialize the parameters of the representation model by transforming the embeddings of two tuples into similarity space and aligning their similarity vectors between the source and target domains. Next, we leverage the prior knowledge from LLMs to separately enrich tuples from the source and target domains, reducing both the source empirical risk and the distribution discrepancy between the source and target domains. Once tuples have been enriched with more robust and distinguishable features and the representation model is well-initialized, we execute the iteration step. This step gradually fine-tunes the representation model using both self-supervised and supervised contrastive learning by selecting and generating high-quality training instances with pseudo-labels from datasets and LLMs.

Comprehensive experiments have been conducted on benchmark datasets, and the experimental results show the effectiveness and efficiency of UDAEB, verifying that leveraging the capabilities of LLMs and designing the proposed framework to boost the performance of entity blocking is a promising direction. Specifically, UDAEB outperforms the state-of-the-



979-8-3503-6248-0/24/$31.00 ©2024 IEEE

art entity blocking systems using metrics of PC, PQ and CSSR.

**Contribution.** The main contributions are summarized as follows.

1) We propose a framework called Unsupervised Domain Adaptation Entity Blocking that integrates implicit feature alignment across domains, enrichment by LLMs and contrastive learning strategy.
2) Because of no training tuple pairs in $\mathsf{T}$, we propose a training data selection strategy that automatically selects high-quality training data for model fine-tuning.
3) We design a schema enrichment mechanism to enrich optimized sets of features for both source and target domain such that they could promote the process of feature alignments.
4) We propose an iterative fine-tuning approach based on self-supervised and supervised contrastive learning that leverages the capabilities of LLMs and data selection.
5) We conduct comprehensive experiments to verify that UDAEB outperforms the existing baselines in several benchmark datasets.

## II. Problem Definition

We explore unsupervised domain adaptation for entity blocking, leveraging sufficient labeled training data in the source domain to transfer domain-invariant features to the target domain that only has unlabeled training data. Let the source and target domains be denoted by $\mathsf{S}$ and $\mathsf{T}$, respectively. Each domain has left and right tables of tuples, denoted as $\mathcal{R}_l^{\mathsf{S}} = \{a_i^{\mathsf{S}}\}_{i=1}^{|\mathcal{R}_l^{\mathsf{S}}|}$ and $\mathcal{R}_r^{\mathsf{S}} = \{b_i^{\mathsf{S}}\}_{i=1}^{|\mathcal{R}_r^{\mathsf{S}}|}$ based on attributes $\bar{A}_{\mathsf{S}}$ for the source domain, and $\mathcal{R}_l^{\mathsf{T}} = \{a_i^{\mathsf{T}}\}_{i=1}^{|\mathcal{R}_l^{\mathsf{T}}|}$ and $\mathcal{R}_r^{\mathsf{T}} = \{b_i^{\mathsf{T}}\}_{i=1}^{|\mathcal{R}_r^{\mathsf{T}}|}$ based on attributes $\bar{A}_{\mathsf{T}}$ for the target domain. Here, $a_i^{\mathsf{S}}$ and $b_i^{\mathsf{S}}$ (resp. $a_i^{\mathsf{T}}$ and $b_i^{\mathsf{T}}$) represent $\bar{A}_{\mathsf{S}}$-attribute (resp. $\bar{A}_{\mathsf{T}}$-attribute) tuples in $\mathsf{S}$ (resp. $\mathsf{T}$).

Additionally, we have a set $D^{\mathsf{S}}$ of labeled training data from $\mathsf{S}$, and $D^{\mathsf{T}}$ of unlabeled training data from $\mathsf{T}$, where $D^{\mathsf{S}} = \{(a_i^{\mathsf{S}}, b_i^{\mathsf{S}}, y_i^{\mathsf{S}})\}_{i=1}^{|D^{\mathsf{S}}|}$, and $D^{\mathsf{T}} = \{(a_i^{\mathsf{T}}, b_i^{\mathsf{T}})\}_{i=1}^{|D^{\mathsf{T}}|}$. Here $y_i^{\mathsf{S}} \in \{\mathsf{False}, \mathsf{True}\}$ is the label for the $i$-th pair $(a_i^{\mathsf{S}}, b_i^{\mathsf{S}})$. Using these above notations, we define our problem.

**Unsupervised domain adaptation for entity blocking.** Given the tables of the source domain $(\mathcal{R}_l^{\mathsf{S}}, \mathcal{R}_r^{\mathsf{S}})$, the tables of the target domain $(\mathcal{R}_l^{\mathsf{T}}, \mathcal{R}_r^{\mathsf{T}})$, the set $D^{\mathsf{S}}$ of labeled training data from $\mathsf{S}$, and $D^{\mathsf{T}}$ of unlabeled training data from $\mathsf{T}$, the objective of unsupervised domain adaptation for entity blocking is to find all potentially matching tuple pairs in $\mathcal{R}_l^{\mathsf{T}} \times \mathcal{R}_r^{\mathsf{T}}$ efficiently.

Solving the entity blocking task is non-trivial due to several challenges as follows. (a) Compared with cross-domain entity matching, *e.g.,*DADER [3], that predicts whether two tuples are matched or not, cross-domain entity blocking aims to efficiently retrieve all potentially matching tuple pairs from large-scale data with high precision and recall; (b) It is challenging to achieve knowledge transfer between domains while maintaining fast running times. In Figure 1, we propose a framework of entity blocking, called UDAEB to solve these issues, which consists of three components, feature alignment with adversarial learning, data enrichment with LLMs and iterative training using contrastive learning.

**Representation learning.** We adopt the SentenceBert model [7] as the backbone model for $\mathcal{M}$ such that each tuple $t$ is transformed to its high-dimensional embedding vector. Specifically, we serialize $t$ to a sequence as $\mathsf{serial}(t) = [\mathsf{COL}]A_1[\mathsf{VAL}]t[A_1]\ldots[\mathsf{COL}]A_m[\mathsf{VAL}]t[A_m]$, where $A_1, \ldots, A_m$ are $m$ attributes of $t$ and [COL] and [VAL] are special tokens [8]. Then we fed $\mathsf{serial}(t)$ into $\mathcal{M}$ and return its embedding, s.t., $\mathbf{emb}_t = \mathcal{M}(\mathsf{serial}(t))$.

Given the source labeled training data $D^{\mathsf{S}}$, we transform it into a set $\mathsf{CL}^{\mathsf{S}}$ of triplets, *s.t.* $\mathsf{CL}^{\mathsf{S}} = \{(a, \mathcal{P}_a, \mathcal{N}_a) | \forall a, (a, b_1, \mathsf{True}) \in D^{\mathsf{S}}, (a, b_2, \mathsf{False}) \in D^{\mathsf{S}}, b_1 \in \mathcal{P}_a, b_2 \in \mathcal{N}_a\}$, where $\mathcal{P}_a$ and $\mathcal{N}_a$ are the sets of tuples that match(w.r.t. positive) and mismatch(w.r.t. negative) with $a$, respectively. After obtaining $\mathsf{CL}^{\mathsf{S}}$ by aggregating all tuples like $a$, we fine-tune $\mathcal{M}$ using the contrastive learning loss function [9] for each tuple $a$.

## III. Data Enrichment with LLMs

To generate robust representations of the source and target domains, we enrich tuples with LLMs that could generate more structural data, containing explicit domain-invariant features. Given the basic attribute set $\bar{A}_{\mathsf{S}}$ of $D^{\mathsf{S}}$ and $\bar{A}_{\mathsf{T}}$ of $D^{\mathsf{T}}$, we aim to extend these to enriched attribute sets $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$, respectively.

Consider a tuple pair $(a^{\mathsf{S}}, b^{\mathsf{S}}, y^{\mathsf{S}}) \in D^{\mathsf{S}}$. We manually create an enrichment instruction for a large language model (LLM) to generate a set of possible enriched attributes. Subsequently, we scan all tuple pairs in $D^{\mathsf{S}}$, counting the frequency of each generated attribute. Attributes with frequencies below a predefined threshold are then filtered out. Finally, we collect a full set $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$ of attributes with high frequencies to enrich. This process is similarly applied to generate enriched attributes $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$ in the target domain. After that, we manually select subsets $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$ from $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$ and $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$, respectively.

Given a set of attributes $\bar{B} \in \{\bar{B}_{\mathsf{S}}, \bar{B}_{\mathsf{T}}\}$, we further handcraft an instruction for LLMs to impute values for $\bar{B}$ in tuple pairs $(a, b)$. This transforms tuples $a$ and $b$ from $\bar{A}$-attribute tuples to $(\bar{A} \cup \bar{B})$-attribute tuples. We denote the enriched tuple pair $(a, b)$ with $\bar{B}$ attributes as $(a, b)_{\bar{B}}$ and the enriched set as $(D)_{\bar{B}}$, where $\bar{A} \in \{\bar{A}_{\mathsf{S}}, \bar{A}_{\mathsf{T}}\}$.

## IV. The Framework of Cross-domain Entity Blocking

In this section, we design a framework for cross-domain entity blocking by leveraging the capabilities of LLMs and contrastive learning. This framework integrates data enrichment, adversarial domain adaptation, and data selection.

### A. Cross-domain Training Strategy

To further adapt $\mathcal{M}$ to the target domain, we employ a two-step contrastive learning strategy: self-supervised contrastive learning leveraging LLMs and supervised contrastive learning based on pseudo-labeled training instances.

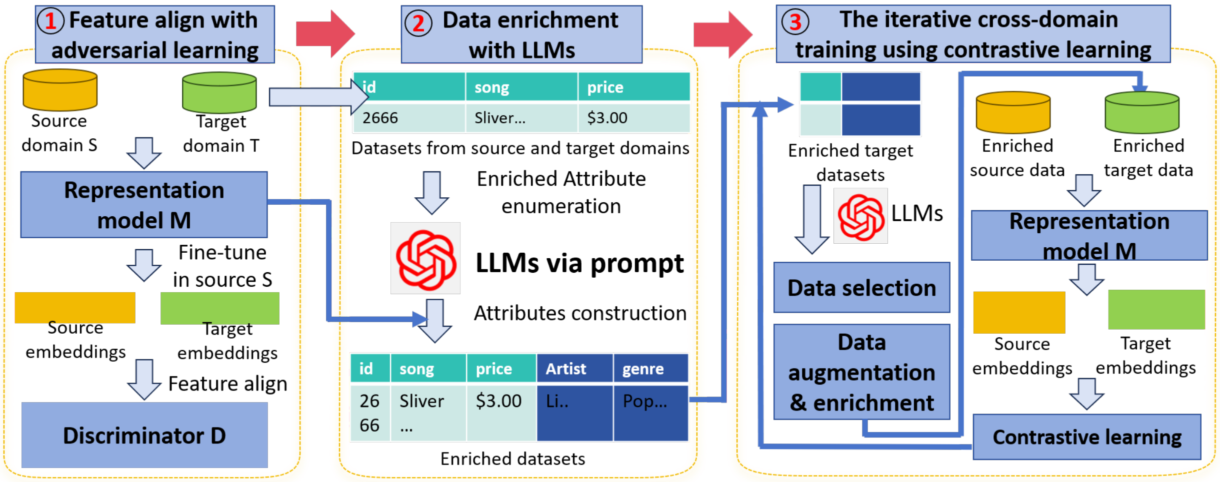



Fig. 1: The Framework of UDAEB

**(1) Self-supervised contrastive learning.** Because the target training data $D^{\mathsf{T}}$ has no labels, we combine data augmentation and enrichment by LLMs.

Data augmentation. We extract all single tuples $t \in \mathcal{R}_l^{\mathsf{T}} \cup \mathcal{R}_r^{\mathsf{T}}$ and adopt $L_{\mathsf{aug}}$ data augmentation strategies [10], *i.e.*, random insertion, deletion, and replacement of tokens in $t$, to generate $L_{\mathsf{aug}}$ augmented tuples as the positive set $\mathcal{P}_t^{\mathsf{aug}}$ of $t$.

Data enrichment by LLMs. To increase the robustness of target tuple representations, we further utilize the powerfulness of LLMs. Recall that $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$ is the set of all enriched attributes in $\mathsf{T}$, and we randomly sample $L_{\mathsf{LLM}}$ subsets from $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$ to generate a new set $\mathcal{S}$ of size $L_{\mathsf{LLM}}$, such that $t[C]$ is the record of attributes $C \in \mathcal{S}$ and $C \subseteq \bar{B}_{\mathsf{T}}^{\mathsf{all}}$. Thus we have $t[\bar{A}_{\mathsf{T}} \cup C_i]$ and $t[\bar{A}_{\mathsf{T}} \cup C_j]$ are the same entity for $C_i, C_j \in \mathcal{S}$. Finally we get a new positive set $\mathcal{P}_t^{\mathsf{LLM}}$ of $t$ such that $\mathcal{P}_t^{\mathsf{LLM}} = \{t[\bar{A}_{\mathsf{T}} \cup C] | C \in \mathcal{S}\}$.

Self-supervised learning. Now we could finally integrate the effectiveness of the above two techniques such that $\mathcal{P}_t = \mathcal{P}_t^{\mathsf{LLM}} \times \mathcal{P}_t^{\mathsf{aug}}$. In detail, for each enriched tuple $t' \in \mathcal{P}_t^{\mathsf{LLM}}$, we generate $L_{\mathsf{aug}}$ augmented tuples $t'_1, \ldots, t'_{L_{\mathsf{aug}}}$ as the positive ones. Finally, we generate the positive set $\mathcal{P}_t$ of $t$ with size $|\mathcal{P}_t| = L_{\mathsf{LLM}} \cdot L_{\mathsf{aug}}$, and adopt the hard negative sampling methods to find the negative set $\mathcal{N}_t$. Finally $(t, \mathcal{P}_t, \mathcal{N}_t)$ is the triplet of $t$ for self-supervised contrastive learning.

**(2) Supervised contrastive learning.** We automatically select high-quality training triplets from $\mathcal{R}_l^{\mathsf{T}} \times \mathcal{R}_r^{\mathsf{T}}$ with pseudo-labels. Unlike existing pseudo-label-based ER methods, such as PromptEM [11], we primarily focus on generating the pseudo positive set of tuples instead of directly predicting the pseudo-labels of tuple pairs, thus preventing noises from the training instances. To achieve this, we first scan each tuple $t \in \mathcal{R}_l^{\mathsf{T}}$ and retrieve its top-$K$ most similar tuples in $\mathcal{R}_r^{\mathsf{T}}$, denoting the top-$K$ set as $\mathsf{KNN}_t$. Next, following [12], we randomly sample a high-quality coreset with probabilities, *i.e.*, $\mathsf{KNN} = \{(t, s) | s \in \mathsf{KNN}_t, t \in \mathcal{R}_l^{\mathsf{T}}\}$. For each tuple pair $(t, s) \in \mathsf{KNN}$, we assign a probability $p$ to it. Their embeddings are $\mathbf{emb}_t$ and $\mathbf{emb}_s$, and we compute $p = \frac{\exp[\pi \cdot \langle \mathbf{emb}_t, \mathbf{emb}_s \rangle]}{\sum_{(a,b) \in \mathsf{KNN}} \exp[\pi \cdot \langle \mathbf{emb}_a, \mathbf{emb}_b \rangle]}$. More specifically, we assign probabilities for all tuple pairs in KNN and randomly sample a subset $\mathcal{C}^{\mathsf{T}}$ with a preset sampling ratio. Here we set the ratio as $\frac{1}{K}$.

Notice that we could simply select the top-1 similar tuple pairs in KNN instead of sampling a coreset with probabilities. However, as discussed in [13], deterministic selection with the highest similarities often results in inferior performance. Random sampling with probabilities is beneficial because (1) it involves some exploration on samples with the same probabilities, and (2) a bit of randomness in the training data is essential to achieve a high-quality solution for non-convex models such as DNNs [13].

### *B. The Cross-domain Framework*

By integrating the enrichment by LLMs, adversarial learning strategy and contrastive learning, we design a cross-domain entity blocking framework to gradually fine-tune $\mathcal{M}$ to achieve good performance.

With inputs of the labeled source training tuple pairs $D^{\mathsf{S}}$ and unlabeled target training tuple pairs $D^{\mathsf{T}}$, the target left and right tables $\mathcal{R}_l^{\mathsf{T}}$ and $\mathcal{R}_r^{\mathsf{T}}$, and a pre-trained LLM, we output the tuned embedding model $\mathcal{M}$ to adapt in the target domain $\mathsf{T}$. In the beginning we start a simple but effective warmup process as follows. We first transform $D^{\mathsf{S}}$ into the set of triplets $\mathsf{CL}^{\mathsf{S}}$ and fine-tune $\mathcal{M}$ using the contrastive learning so that $\mathcal{M}$ could be adapted in $\mathsf{S}$. Next a MLP-based discriminator $D$ is added and we adopt the adversarial learning strategy [14] to further fine-tune $\mathcal{M}$ and $D$ so that similarity representations of $\mathsf{S}$ and $\mathsf{T}$ are aligned together.

After the warmup step, we fix parameters of $\mathcal{M}$ and $D$, and initialize the enrichment step. We first retrieve all enriched attributes $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$ and $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$ for the source and target domains by referencing LLM, respectively. Next we adopt LLM to fill in values of enriched attributes of all tuples in $\mathcal{R}_l^{\mathsf{T}}$ and $\mathcal{R}_r^{\mathsf{T}}$. Then we select subsets $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$ as the final enriched features for tuple pairs in S and T. Now we start our iteration step to simultaneously fine-tune $\mathcal{M}$ in S and T. First we generate the positive tuple set for each tuple in $\mathcal{R}_l^{\mathsf{T}}$ and $\mathcal{R}_r^{\mathsf{T}}$ by using the capability of LLM and data augmentation strategies. Next we adopt the self-supervised contrastive learning in $\mathcal{R}_l^{\mathsf{T}}$ and $\mathcal{R}_r^{\mathsf{T}}$ to fine-tune $\mathcal{M}$.After self-supervised contrastive learning, we further adopt $\mathcal{M}$ to execute $K$ nearest neighbour search in $\mathcal{R}_l^{\mathsf{T}} \times \mathcal{R}_r^{\mathsf{T}}$, and adaptively select high-quality tuple pairs $\mathcal{C}^{\mathsf{T}}$ with pseudo-labels as the new training data. Finally, we fine-tune $\mathcal{M}$ using the supervised contrastive learning in $D^{\mathsf{S}}$ and $\mathcal{C}^{\mathsf{T}}$. $\mathcal{M}$ is iteratively fine-tuned until the maximum iteration iter is reached.

## V. Experimental Study

In this section, we conduct comprehensive experiments to evaluate the performance of UDAEB. First, we evaluate the accuracy of UDAEB with different $K$ values across several cross-domain benchmarks. Then, we assess UDAEB using the candidate set size ratio (CSSR) to demonstrate the quality of the candidate set retrieved by UDAEB.

### A. Experimental Setup

We show the experimental settings and datasets.

**Baselines**. We compare with the state-of-the-art entity blocking methods, including DeepBlocker [1], Sudowoodo [2] and STransformer [7]. We implement UDAEB using Pytorch 2.3 and transformer-based FlagEmbedding library [15]. we use `bge-large-en` as the pre-trained model for $\mathcal{M}$ and adopt Mistral-7B [16] as the LLM for enrichment. We adopt the AdamW optimizer with the learning rate of 1e-5, and the batch size of 16 for fine-tuning. The number of iteration steps is 5 for all datasets. For data enrichment, we adopt vLLM [17] to accelerate the inference process.

For all baselines, we adopt their default implementations and settings. For fair comparison, we also incorporate the labeled training instances from the source domain into their generated training data. We use the same dimensions of embeddings for all baselines and do not compare their efficiency in $K$ nearest neighbor search, as all transform tuples into embeddings with the same dimension sizes. Thus, in the remaining part of the experiment, we mainly focus on the effectiveness of entity blocking, including PC, PQ, and CSSR.

**Datasets**. Table I shows the statistical information of all used benchmark datasets. Here we focus on unsupervised domain adaptation and denote an entity blocking task by $\mathsf{S} \rightarrow \mathsf{T}$, where S and T are the source and target domains.

**Metrics**. Following Sudowoodo [2] and DeepBlocker [1], we adopt three evaluation metrics: (a) pair completeness (PC), also known as **Recall**, which is the fraction of true matched tuple pairs identified in the golden groundtruth; (b) pair quality (PQ), also known as **Precision**, which is the fraction of true matched tuple pairs in the candidate tuple pairs; and (c) the candidate set size ratio (CSSR), which is the fraction of candidate size in $|\mathcal{R}_l^{\mathsf{T}}| \times |\mathcal{R}_r^{\mathsf{T}}|$.

**Configuration**. We conducted the experiments on a machine powered by 256GB RAM and 32 processors with Intel(R) Xeon(R) Gold 5320 CPU @2.20GHz and 2 NVIDIA GeForce A800 GPUs. Each experiments was run 3 times and the average is reported.

### B. Comparison of Effectiveness

In Figure 2(a) to 2(h), where we vary $K$ from 1 to 20, we display the PC and PQ of the baselines. The curve approaching the **upper left corner** of the figure indicates better performance. UDAEB consistently demonstrates higher accuracy than other baselines across PC and PQ metrics, particularly noticeable for smaller values of $K$, *e.g.,* when $K = 1$, UDAEB achieves PC and PQ values that are 50.9% and 48.3% higher than the best performing baseline in RI-AB, respectively. This underscores UDAEB's capability to retrieve all matching results with a small $K$ by integrating similarity feature alignment between the source and target domains, enrichment by LLMs and iterative contrastive learning based on LLMs and high-quality pseudo-labels.

### C. Candidate Set of Entity Blocking

In Figures 2(i) to 2(l), we present the CSSR values while varying the PC of the baselines. A smaller CSSR and a larger PC (w.r.t. the curve approaching the **lower right corner** of the figure) indicate better performance. UDAEB consistently outperforms other baselines, suggesting that it returns a smaller set of candidates for downstream entity matching processes compared to others, even when they are tasked with retrieving the same number of matching tuples. Additionally, the entity resolution pipeline involving UDAEB demonstrates greater efficiency than others, as UDAEB achieves similar performance in entity resolution using smaller values of $K$.

### D. The Training Cost

We evaluate the training time of different baselines in Table II. Although UDAEB adopts LLMs for data enrichment and an iterative process for fine-tuning $\mathcal{M}$, its training time is not high, *e.g.,* 521s and 274s of data enrichment and the iterative process of learning $\mathcal{M}$ on RI-AB, respectively.

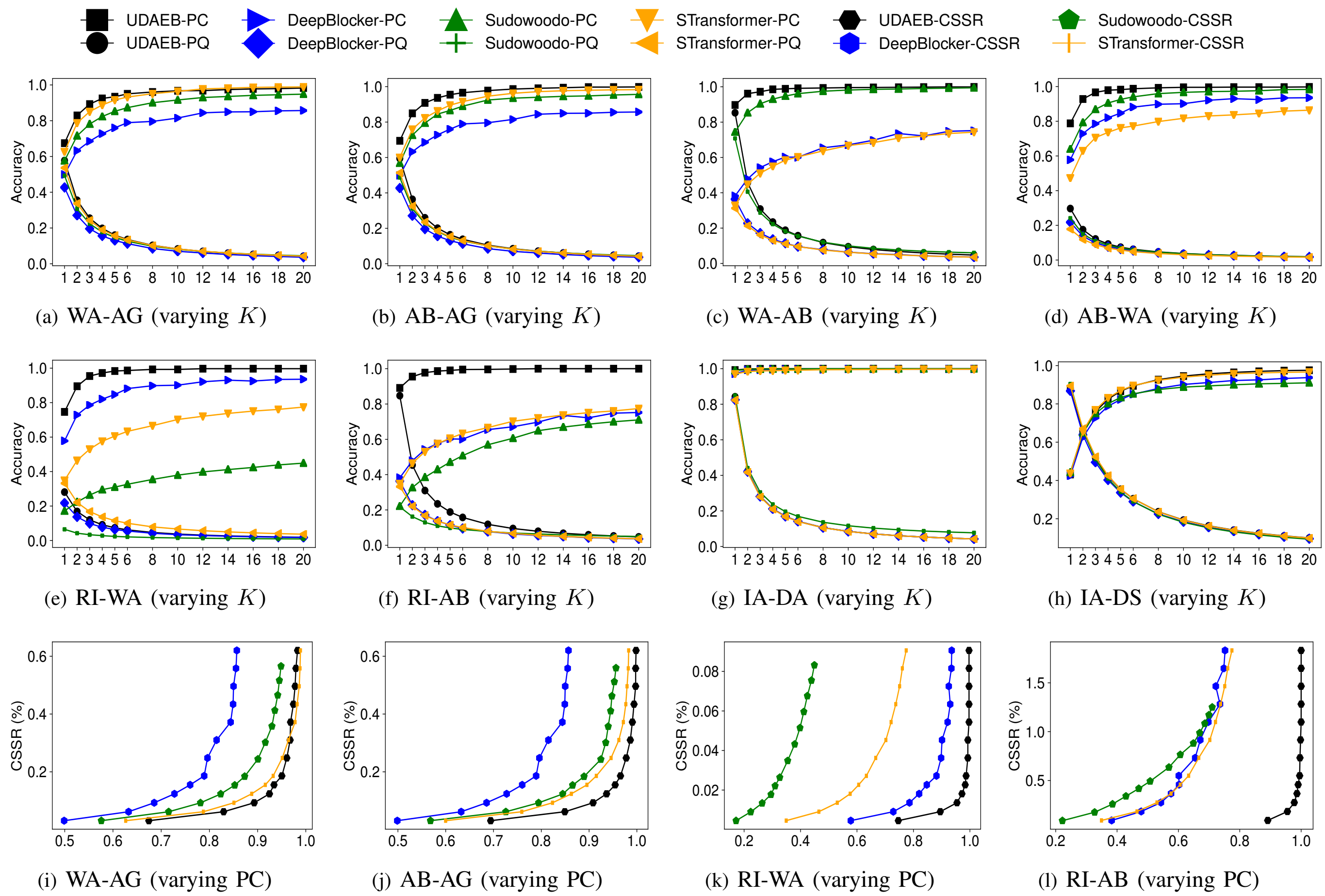
## VI. Related Work

### A. Entity Blocking

We classify entity blocking into rule-based methods, traditional ML-based methods, and deep learning-based models. (1) **Rule-based methods**. These non-learning methods adopt hash-based, sort-based, size-based, and similarity-based techniques that require handcrafted rules by experts to retrieve tuple pairs from datasets [19]. More effective methods have been proposed, including meta-blocking [20], matching dependencies [21], and DNF-based methods [22], which fully consider the correlations among tuples and attributes. Due to the limitations of these approaches, learning rules have been introduced to discover rules based on predefined predicates, such as ApproxDNF [23], BSL [24], and EM-GBF [25]. Sparkly [26] employs the tf/idf blocking technique to achieve

TABLE I: Datasets used in our experiments, # means Number of

| Dataset | Domain | # All Pair | # Match Pair | # of Original Attributes | # of Attributes after Enrichment | # $\|\mathcal{R}_l\|, \|\mathcal{R}_r\|$ |
|---|---|---|---|---|---|---|
| Abt-Buy (AB) [18] | Product | 9,575 | 1,028 | 3 | 8 | 1081,1092 |
| Walmart-Amazon (WA) [18] | Product | 10,242 | 962 | 5 | 9 | 2554,22074 |
| Amazon-Google (AG) [18] | Product | 11,460 | 1,300 | 3 | 9 | 1363,3226 |
| iTunes-Amazon (IA) [18] | Music | 539 | 132 | 3 | 6 | 6907,55932 |
| DBLP-ACM (DA) [18] | Citation | 12,363 | 2,224 | 4 | 6 | 2616,2294 |
| DBLP-Scholar (DS) [18] | Citation | 28,707 | 5,347 | 4 | 6 | 2616,64263 |
| RottenTomatoes-IMDB (RI) [3] | Movies | 600 | 190 | 3 | 5 | 557,554 |




(a) WA-AG (varying $K$)

(b) AB-AG (varying $K$)

(c) WA-AB (varying $K$)

(d) AB-WA (varying $K$)

(e) RI-WA (varying $K$)

(f) RI-AB (varying $K$)

(g) IA-DA (varying $K$)

(h) IA-DS (varying $K$)

(i) WA-AG (varying PC)

(j) AB-AG (varying PC)

(k) RI-WA (varying PC)

(l) RI-AB (varying PC)

Fig. 2: Effectiveness evaluation. For figures varying $K$, the curve near **upper left corner** for PC, and near **upper right corner** for PQ indicate better performance. For figures varying PC, the curve near **lower right corner** indicates better result.

TABLE II: The Training Time of Different Baselines (in seconds). For UDAEB, we split it as the enrichment time ($T_{enrich}$) and fine-tuning time $T_{FT}$.

| Datasets | UDAEB ($T_{enrich}$ + $T_{FT}$) | DeepBlocker | Sudowoodo |
|---|---|---|---|
| RI-AB | 521 + 274 | 73 | 137 |
| RI-WA | 437 + 309 | 594 | 137 |
| WA-AG | 434 + 291 | 116 | 383 |
| AB-AG | 476 + 313 | 116 | 235 |

high efficiency and effectiveness. (2) **ML-based models**. These methods learn ML classifiers to make inferences efficiently, such as Smurf [27] and supervised meta-block [28]. Most of them focus on active learning and accelerating the inference of ML models. (3) **DL-based models**. With the advent of neural networks, state-of-the-art entity blocking models are based on deep learning, including DeepER [29], STransformer [7], DeepBlocker [1], SC-Blocker [30], Sudowoodo [2] and UniBlocker [31] that design embedding models with contrastive learning.

Compared with existing works, UDAEB focuses on cross-domain entity blocking and proposes a framework that leverages the capabilities of LLMs for enrichment. Furthermore, we enhance contrastive learning in entity blocking by using self-supervised methods based on LLMs and supervised methods utilizing generated pseudo-labeled training instances.

### B. Unsupervised Domain Adaptation

In the ML community, unsupervised domain adaptation techniques are well-studied and can be broadly classified into five categories: (1) feature-centric methods, *e.g.,*AE-SCL [32]; (2) loss-centric methods, *e.g.,* CGANS [14]; (3) pseudo-labeling techniques, *e.g.,* [33]; (4) data selection methods, *e.g.,* [34]; and (5) pre-training methods, *e.g.,* [35].

In entity resolution, unsupervised domain adaptation has been developed for entity matching, including DADER [3], MFSN [4], TL-ER [36], TransER [37] and MELD [6]. Unlike entity matching, which is typically framed as a binary classification task, entity blocking is a ranking task. Therefore, we employ different methods for cross-domain entity blocking that is effective and efficient.

## VII. Conclusion

This paper introduces UDAEB, an unsupervised domain adaptation framework for entity blocking. UDAEB comprises three main steps: warmup, which aligns feature representations in the similarity space between source and target domains; enrichment, leveraging LLMs to enhance robust attributes for both domains; and iteration, integrating self-supervised and supervised contrastive learning using LLMs and pseudo-labeled training instances. We conduct comprehensive experiments across seven benchmarks to show the superior performance of UDAEB against the state-of-the-art methods using three standard metrics, *i.e.,* PC, PQ, and CSSR.

## Acknowledgments

This work was supported by Longhua Science and Technology Innovation Bureau 10162A20220720B12AB12.

## References

[1] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, and A. Doan, "Deep learning for blocking in entity matching: A design space exploration," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2459–2472, 2021.

[2] R. Wang, Y. Li, and J. Wang, "Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation," in *ICDE*, 2023.

[3] J. Tu, X. Han, J. Fan, N. Tang, C. Chai, G. Li, and X. Du, "DADER: hands-off entity resolution with domain adaptation," *PVLDB*, vol. 15, no. 12, pp. 3666–3669, 2022.

[4] C. Sun, Y. Xu, D. Shen, and T. Nie, "Matching feature separation network for domain adaptation in entity matching," in *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, T. Chua, C. Ngo, R. Kumar, H. W. Lauw, and R. K. Lee, Eds. ACM, 2024, pp. 1975–1985.

[5] H. Zhang, Y. Dong, C. Xiao, and M. Oyamada, "Jellyfish: A large language model for data preprocessing," *CoRR*, 2023.

[6] M. Yan, Y. Wang, K. Pang, M. Xie, and J. Li, "Efficient mixture of experts based on large language models for low-resource data preprocessing," 2024.

[7] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP*, 11 2019.

[8] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan, "Deep entity matching with pre-trained language models," *PVLDB*, vol. 14, no. 1, pp. 50–60, 2020.

[9] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[10] Z. Miao, Y. Li, and X. Wang, "Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond," in *SIGMOD*. ACM, 2021, pp. 1303–1316.

[11] P. Wang, X. Zeng, L. Chen, F. Ye, Y. Mao, J. Zhu, and Y. Gao, "Promptem: Prompt-tuning for low-resource generalized entity matching," *PVLDB*, 2022.

[12] T. Zhou, S. Wang, and J. A. Bilmes, "Robust curriculum learning: from clean label detection to noisy label self-correction," in *ICLR*, 2021.

[13] ——, "Curriculum learning by dynamic instance hardness," in *NeurIPS*, 2020.

[14] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 1647–1657.

[15] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J. Nie, "Retrieve anything to augment large language models," *CoRR*, 2023.

[16] A. Q. J. et. al., "Mistral 7b," *CoRR*, 2023.

[17] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[18] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *SIGMOD*, 2018, pp. 19–34.

[19] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas, "A survey of blocking and filtering techniques for entity resolution," *CoRR*, 2019.

[20] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl, "Meta-blocking: Taking entity resolutionto the next level," *IEEE Trans. Knowl. Data Eng.*, 2014.

[21] W. Fan, X. Jia, J. Li, and S. Ma, "Reasoning about record matching rules," *PVLDB*, vol. 2, no. 1, pp. 407–418, 2009.

[22] M. Kejriwal and D. P. Miranker, "A DNF blocking scheme learner for heterogeneous datasets," *CoRR*, vol. abs/1501.01694, 2015.

[23] M. Bilenko, B. Kamath, and R. J. Mooney, "Adaptive blocking: Learning to scale up record linkage," in *ICDM*, 2006.

[24] M. Michelson and C. A. Knoblock, "Learning blocking schemes for record linkage," in *AAAI*, 2006.

[25] R. Singh, V. V. Meduri, A. K. Elmagarmid, S. Madden, P. Papotti, J. Quiané-Ruiz, A. Solar-Lezama, and N. Tang, "Synthesizing entity matching rules by examples," *Proc. VLDB Endow.*, 2017.

[26] D. Paulsen, Y. Govind, and A. Doan, "Sparkly: A simple yet surprisingly strong TF/IDF blocker for entity matching," *Proc. VLDB Endow.*, 2023.

[27] P. S. G. C., A. Ardalan, A. Doan, and A. Akella, "Smurf: Self-service string matching using random forests," *Proc. VLDB Endow.*, 2018.

[28] L. Gagliardelli, G. Papadakis, G. Simonini, S. Bergamaschi, and T. Palpanas, "GSM: A generalized approach to supervised meta-blocking for scalable entity resolution," *Inf. Syst.*, 2024.

[29] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, and N. Tang, "Distributed representations of tuples for entity resolution," *PVLDB*, vol. 16, no. 8, pp. 1944–1957, 2018.

[30] A. Brinkmann, R. Shraga, and C. Bizer, "Sc-block: Supervised contrastive blocking within entity resolution pipelines," in *ESWC*, 2024.

[31] T. Wang, H. Lin, X. Han, X. Chen, B. Cao, and L. Sun, "Towards universal dense blocking for entity resolution," *CoRR*, vol. abs/2404.14831, 2024.

[32] Y. Ziser and R. Reichart, "Neural structural correspondence learning for domain adaptation," in *CoNLL*, R. Levy and L. Specia, Eds., 2017.

[33] X. Cui and D. Bollegala, "Self-adaptation for unsupervised domain adaptation," in *RANLP*, R. Mitkov and G. Angelova, Eds., 2019.

[34] R. Aharoni and Y. Goldberg, "Unsupervised domain clusters in pretrained language models," in *ACL*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds., 2020.

[35] A. Ladkat, A. Miyajiwala, S. Jagadale, R. Kulkarni, and R. Joshi, "Towards simple and efficient task-adaptive pre-training for text classification," in *AACL/IJCNLP*, 2022.

[36] J. Kasai, K. Qian, S. Gurajada, Y. Li, and L. Popa, "Low-resource deep entity resolution with transfer and active learning," in *ACL*, 2019.

[37] N. Kirielle, P. Christen, and T. Ranbaduge, "Transer: Homogeneous transfer learning for entity resolution," in *EDBT*, 2022.