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

***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 so that both of distribution discrepancy and source empirical risk are reduced. 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, *e.g.,* its pair completeness (PC) and pair quality (PQ) are 7.26% and 13.54% higher than others on average, respectively. The codes and datasets are available[1].**

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

I. Introduction

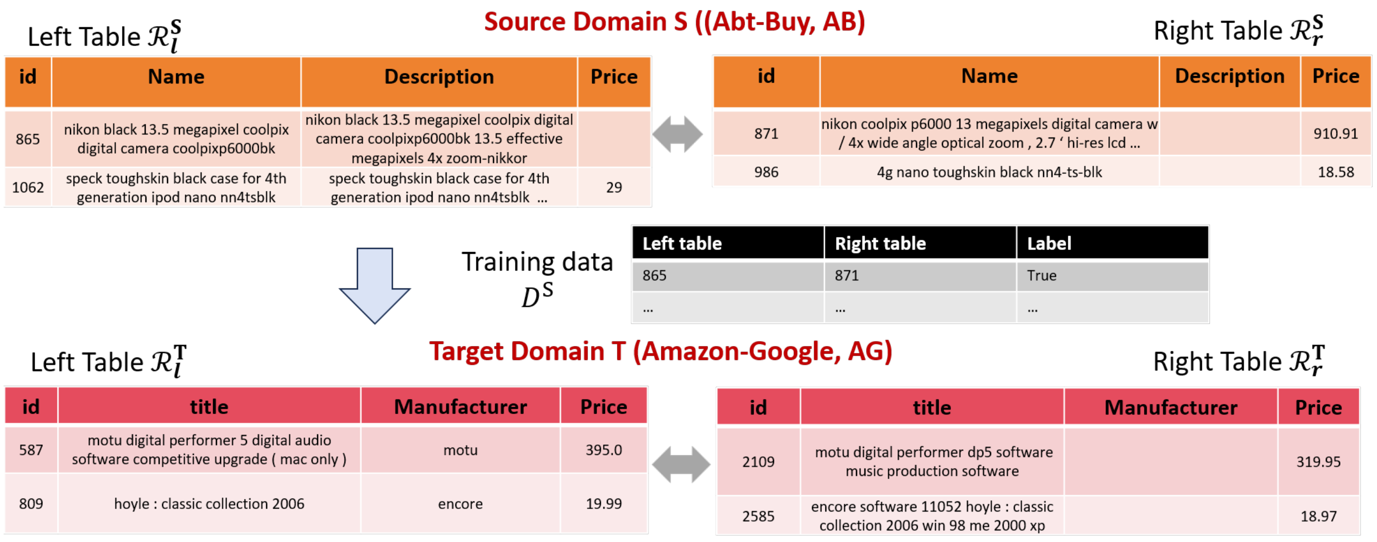
Entity resolution (ER), also called de-duplication and record linkage, aims to retrieve and identify all matched tuple pairs in collections of tuples. It is an important component for data cleaning, information integration, and data processing pipelines for training machine learning models. When encountering 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. The representation models are often implemented using an encoder of Transformer [1], such as BERT based models, to transform tuples into vectors. Specific loss functions, such as contrastive loss, are designed to pull embeddings of tuples that refer to the same entities closer together and push apart those of non-matched ones. After tuples are transformed into their dense semantic vectors, efficient vector retrieval tools could be adopted, such as Faiss [2], to find the top-K similar results as the candidate set.

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 [3] and Sudowoodo [4], 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 [5] and MFSN [6] 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 [7] and MELD [8], 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

[1]https://anonymous.4open.science/r/Transfer-ER-Blocking-58F1/


Source Domain S ((Abt-Buy, AB))

Left Table $\mathcal{R}_l^S$

id	Name	Description	Price
865	nikon black 13.5 megapixel coolpix digital camera coolpixp6000bk	nikon black 13.5 megapixel coolpix digital camera coolpixp6000bk 13.5 effective megapixels 4x zoom-nikkor	
1062	speck toughskin black case for 4th generation ipod nano nn4tsblk	speck toughskin black case for 4th generation ipod nano nn4tsblk ...	29

Right Table $\mathcal{R}_r^S$

id	Name	Description	Price
871	nikon coolpix p6000 13 megapixels digital camera w / 4x wide angle optical zoom , 2.7 ' hi-res lcd ...		910.91
986	4g nano toughskin black nn4-ts-blk		18.58

Training data D^S

Left table	Right table	Label
865	871	True
...	...	...

Target Domain T (Amazon-Google, AG)

Left Table $\mathcal{R}_l^T$

id	title	Manufacturer	Price
587	motu digital performer 5 digital audio software competitive upgrade (mac only)	motu	395.0
809	hoyle : classic collection 2006	encore	19.99

Right Table $\mathcal{R}_r^T$

id	title	Manufacturer	Price
2109	motu digital performer dp5 software music production software		319.95
2585	encore software 11052 hoyle : classic collection 2006 win 98 me 2000 xp		18.97

Fig. 1: Examples of Unsupervised Domain Adaptation for Entity Blocking

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-art entity blocking systems, *e.g.,* its PC and PQ are 7.26% and 13.54% higher than other baselines on average.

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 with pseudo-labeled training instances.
2) 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.
3) We propose an iterative fine-tuning approach based on self-supervised and supervised contrastive learning that leverages the capabilities of LLMs and data selection.
4) 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 S and 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^{S} and b_i^{S} (resp. a_i^{T} and b_i^{T}) represent $\bar{A}_{\mathsf{S}}$-attribute (resp. $\bar{A}_{\mathsf{T}}$-attribute) tuples in S (resp. T).

Additionally, we have a set D^{S} of labeled training data from S, and D^{T} of unlabeled training data from 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^{S} of labeled training data from S, and D^{T} of unlabeled training data from 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.

Example 1: We present an example of entity blocking in Figure 1. The left table, $\mathcal{R}_l^{\mathsf{S}}$, and the right table, $\mathcal{R}_r^{\mathsf{S}}$, consist of two tuples from the source domain S, *i.e.*,, Abt-Buy, with attributes id, Name, Description, and Price as $\bar{A}_{\mathsf{S}}$. Additionally, we have a set of labeled training data from S, denoted as D^{S}. For example, {865, 871, True } indicates that the 865th tuple in $\mathcal{R}_l^{\mathsf{S}}$ and the 871st tuple in $\mathcal{R}_r^{\mathsf{S}}$ are a match.

We are also provided with the left table, $\mathcal{R}_l^{\mathsf{T}}$, and the right table, $\mathcal{R}_r^{\mathsf{T}}$, from the target domain T of schema $\bar{A}_{\mathsf{T}}$, *i.e.*,, Amazon-Google, as well as unlabeled training data D^{T}. The goal of unsupervised domain adaptation for entity blocking is to efficiently retrieve a set of matched tuple pairs, such as $\{587, 2109\}$ and $\{809, 2585\}$, from $\mathcal{R}_l^{\mathsf{T}} \times \mathcal{R}_r^{\mathsf{T}}$, by leveraging $\mathcal{R}_l^{\mathsf{S}}$, $\mathcal{R}_r^{\mathsf{S}}$, D^{S}, and D^{T}.

□

Solving the entity blocking task is non-trivial due to several challenges as follows. (a) Compared with cross-domain entity matching, *e.g.,*DADER [5], 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 3, 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.

III. Feature Alignment with Adversarial Learning

In this section, we introduce how to learn the representation model and align distributions across domains using source labeled D^{S} and unlabeled target D^{T}.

A. Representation Learning

We adopt the SentenceBert model [9] as the backbone model for $\mathcal{M}$ to transform each tuple t 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 [10]. Then we fed $\mathsf{serial}(t)$ into $\mathcal{M}$ and return its embedding, s.t., $\mathbf{emb}_t = \mathcal{M}(\mathsf{serial}(t))$.

To fine-tune $\mathcal{M}$ in S, we adopt the contrastive learning loss [11]. Given the source labeled training data D^{S}, we transform it into a set CL^{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. In other words, for each tuple $a \in \mathcal{R}_l^{\mathsf{S}}$, we scan the right tuples in $\mathcal{R}_r^{\mathsf{S}}$ to find those that match and mismatch with a, resulting in pairs (a, b_1, True) and (a, b_2, False). After obtaining CL^{S} by aggragating all tuples like a, we fine-tune $\mathcal{M}$ using contrastive learning for each tuple a as follows.

$$\mathcal{L}_{\mathsf{CL}} = \sum_{b \in \mathcal{P}_a} -\log \frac{\exp(\langle \mathbf{emb}_a, \mathbf{emb}_b \rangle / \tau)}{\sum_{b' \in \mathcal{P}_a \cup \mathcal{N}_a} \exp(\langle \mathbf{emb}_a, \mathbf{emb}_{b'} \rangle) / \tau} \quad (1)$$

where τ is the temperature parameter to control the level of smoothness.

B. Cross-domain Representation Learning

Although we can adopt contrastive learning to learn from S, we still need to transfer the knowledge to the target domain T. To address this, we first design a simple yet effective method to aggregate embeddings of tuples into a similarity representation. We then use an adversarial learning strategy to align the similarity representations between the source and target domains.

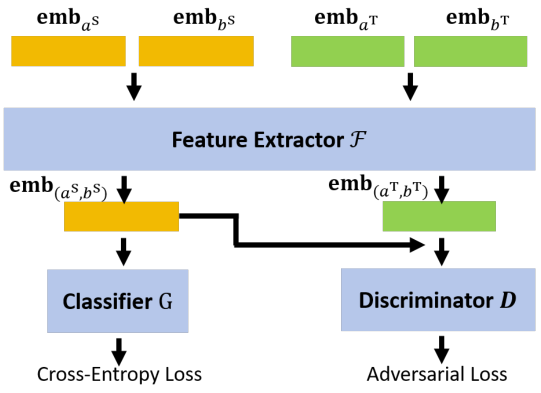

$\mathbf{emb}_{a^{\mathsf{S}}}$
$\mathbf{emb}_{b^{\mathsf{S}}}$
$\mathbf{emb}_{a^{\mathsf{T}}}$
$\mathbf{emb}_{b^{\mathsf{T}}}$
Feature Extractor $\mathcal{F}$
$\mathbf{emb}_{(a^{\mathsf{S}},b^{\mathsf{S}})}$
$\mathbf{emb}_{(a^{\mathsf{T}},b^{\mathsf{T}})}$
Classifier G
Discriminator D
Cross-Entropy Loss
Adversarial Loss

Fig. 2: Cross-domain Representation Learning

In Figure 2, we show the workflow of the cross-domain representation learning module. Specifically, given $(a^{\mathsf{S}}, b^{\mathsf{S}}, y^{\mathsf{S}}) \in D^{\mathsf{S}}$ and $(a^{\mathsf{T}}, b^{\mathsf{T}}) \in D^{\mathsf{T}}$, we first transform each tuple into an embedding, resulting in $\mathbf{emb}_{a^{\mathsf{S}}}$, $\mathbf{emb}_{b^{\mathsf{S}}}$, $\mathbf{emb}_{a^{\mathsf{T}}}$, and $\mathbf{emb}_{b^{\mathsf{T}}}$. Next, we use a feature extractor $\mathcal{F}$, implemented as a MLP-based fully-connected layer NN with the activation function ReLU, to generate the similarity representation. For $(a, b) \in \{(a^{\mathsf{S}}, b^{\mathsf{S}}), (a^{\mathsf{T}}, b^{\mathsf{T}})\}$, we have $\mathbf{emb}_{(a,b)} = \mathcal{F}(\mathbf{emb}_a \oplus \mathbf{emb}_b) = \mathsf{ReLU}(\mathsf{NN}(\mathbf{emb}_a \oplus \mathbf{emb}_b))$, where $\oplus$ is the concatenation operation by row. After generating the similarity embeddings, we also adopt a binary classifier $\mathcal{G}(\cdot)$ to transfer the similarity representation to a binary decision. Finally, we have $\mathcal{G}(\mathcal{F}(\mathbf{emb}_{a^{\mathsf{S}}} \oplus \mathbf{emb}_{b^{\mathsf{S}}}))$, and $\mathcal{G}(\mathcal{F}(\mathbf{emb}_{a^{\mathsf{T}}} \oplus \mathbf{emb}_{b^{\mathsf{T}}}))$ as the logits of $(a^{\mathsf{S}}, b^{\mathsf{S}})$ and $(a^{\mathsf{T}}, b^{\mathsf{T}})$, respectively. Following CGANs [12], we design a binary discriminator D to distinguish whether the similarity representations are from the source or target domain, and then adopt the domain adversarial neural network approach as a two-player game using the following loss function that combines the adversarial loss $\mathcal{L}_{\mathsf{adv}}$ and cross-entropy loss $\mathcal{L}_{\mathsf{CE}}$:

$$\min \mathbb{E}_{(a^{\mathsf{S}},b^{\mathsf{S}},y^{\mathsf{S}})\sim D^{\mathsf{S}}} \mathcal{L}_{\mathsf{CE}}(\mathcal{G}(\mathbf{emb}_{(a^{\mathsf{S}},b^{\mathsf{S}})}), y^{\mathsf{S}}) + \lambda \cdot \Big(\mathbb{E}_{(a^{\mathsf{S}},b^{\mathsf{S}},y^{\mathsf{S}})\sim D^{\mathsf{S}}} \log[D(\mathbf{emb}_{(a^{\mathsf{S}},b^{\mathsf{S}})})] + \mathbb{E}_{(a^{\mathsf{T}},b^{\mathsf{T}})\sim D^{\mathsf{T}}} \log[1 - D(\mathbf{emb}_{(a^{\mathsf{T}},b^{\mathsf{T}})})] \Big), \quad (2)$$

$$\max \mathbb{E}_{(a^{\mathsf{S}},b^{\mathsf{S}},y^{\mathsf{S}})\sim D^{\mathsf{S}}} \log[D(\mathbf{emb}_{(a^{\mathsf{S}},b^{\mathsf{S}})})] + \mathbb{E}_{(a^{\mathsf{T}},b^{\mathsf{T}})\sim D^{\mathsf{T}}} \log[1 - D(\mathbf{emb}_{(a^{\mathsf{T}},b^{\mathsf{T}})})]$$

In summary, by employing the adversarial loss function, we can align the distribution in the similarity representation space of the source and target domains. To better learn the cross-domain representations of tuples in different domains, we initially fine-tune $\mathcal{M}$, $\mathcal{F}$, and $\mathcal{G}$ together in D^{S}. Then, we freeze the parameters of $\mathcal{F}$ and $\mathcal{G}$, and only fine-tune $\mathcal{M}$ in D^{S} and D^{T} so that $\mathcal{M}$ could generate cross-domain embeddings.

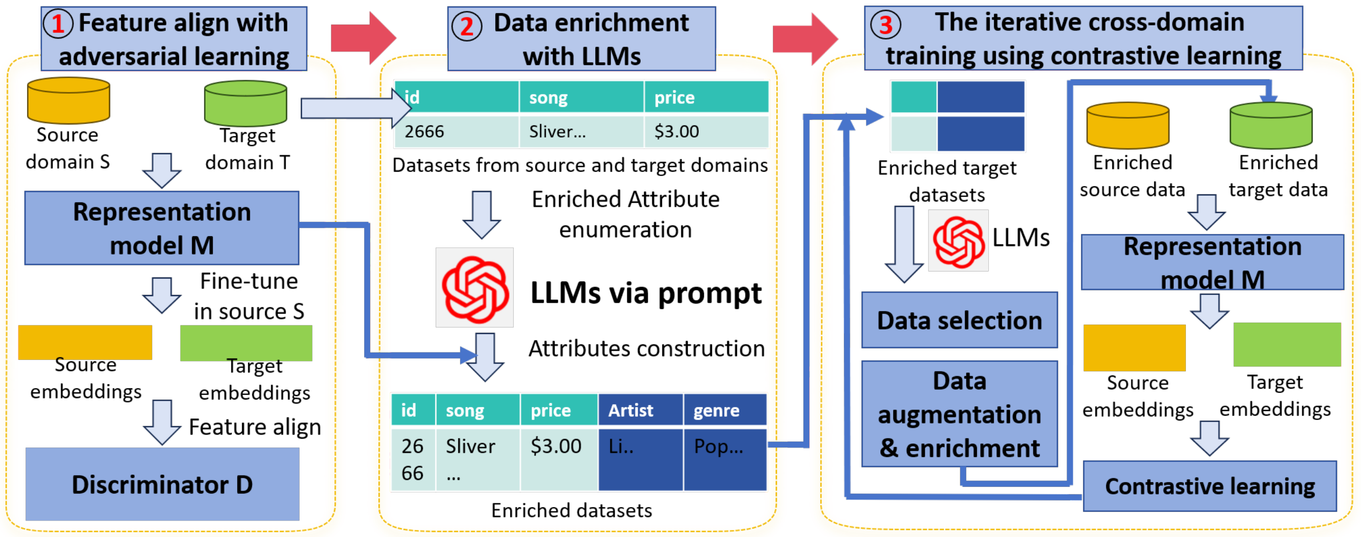


Fig. 3: The Framework of UDAEB

IV. 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^{S} and $\bar{A}_{\mathsf{T}}$ of D^{T}, we aim to extend these to enriched attribute sets $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$, respectively. We first introduce how to enumerate all possible attributes $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$ and $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$, and then discuss how to select an optimized subset from $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$. Once the enriched attributes of S are settled, we propose an effective strategy to select a subset from $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$.

(1) Enriched attribute enumeration. 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^{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.

Given a set of attributes $\bar{B} \in \{\bar{B}_{\mathsf{S}}^{\mathsf{all}}, \bar{B}_{\mathsf{T}}^{\mathsf{all}}\}$, 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 $\bar{A} \in \{\bar{A}_{\mathsf{S}}, \bar{A}_{\mathsf{T}}\}$.

(2) $\bar{B}_{\mathsf{S}}$ construction. Consider a set $\bar{B}_{\mathsf{S}} \subseteq \bar{B}_{\mathsf{S}}^{\mathsf{all}}$, we first transform D^{S} to the enriched one $(D^{\mathsf{S}})_{\bar{B}_{\mathsf{S}}}$, and then adopt the feature extractor $\mathcal{F}$ to partition it to two clusters $\mathbb{C}_{\bar{B}_{\mathsf{S}}}$ (*i.e.,* match and mis-match) according to the k-means clustering. Because D^{S} has ground truth, *i.e.,* labels of matches or mismatches, we could get the true distribution of labels, denoted by Ω. Then we adopt the normalized mutual information [13] (NMI) to calculate the correlation between $\mathbb{C}_{\bar{B}_{\mathsf{S}}}$ and Ω, *s.t.* the higher correlation indicates the effectiveness of $\bar{B}_{\mathsf{S}}$. Thus, our objective function to get $\bar{B}_{\mathsf{S}}$ is $\bar{B}_{\mathsf{S}} = \underset{\bar{B} \subseteq \bar{B}_{\mathsf{S}}^{\mathsf{all}}}{\operatorname{argmax}} \mathsf{NMI}(\mathbb{C}_{\bar{B}}, \Omega)$.
To find $\bar{B}_{\mathsf{S}}$, we adopt a simple greedy approach to gradually add attribute B from $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$ with the largest value of $\mathsf{NMI}(\mathbb{C}_{\bar{B}_{\mathsf{sel}} \cup \{B\}}, \Omega)$ until the values of NMI do not increase, where $\bar{B}_{\mathsf{sel}}$ is the currently selected attributes.

(3) $\bar{B}_{\mathsf{T}}$ construction. After $\bar{B}_{\mathsf{S}}$ has been selected, we select a subset $\bar{B}_{\mathsf{T}} \subseteq \bar{B}_{\mathsf{T}}^{\mathsf{all}}$ to minimize the distribution discrepancy between S and T. Denoted P as the distribution of $(D^{\mathsf{S}})_{\bar{B}_{\mathsf{S}}}$, and $Q_{\bar{B}_{\mathsf{T}}}$ as $(D^{\mathsf{T}})_{\bar{B}_{\mathsf{T}}}$, and we adopt Maximum Mean Discrepancy (MMD) [14] to measure their discrepancy for domain adaptation. After transforming $(D^{\mathsf{S}})_{\bar{B}_{\mathsf{S}}}$ (resp. $(D^{\mathsf{T}})_{\bar{B}_{\mathsf{T}}}$) to the embeddings $(\mathbf{emb}^{\mathsf{S}})_{\bar{B}_{\mathsf{S}}}$ (resp. $(\mathbf{emb}^{\mathsf{T}})_{\bar{B}_{\mathsf{T}}}$) from $\mathcal{F}$, We have the objective function to get $\bar{B}_{\mathsf{T}}$, *s.t.* $\bar{B}_{\mathsf{T}} = \underset{\bar{B} \subseteq \bar{B}_{\mathsf{T}}^{\mathsf{all}}}{\operatorname{argmin}} \mathsf{MMD}((\mathbf{emb}^{\mathsf{T}})_{\bar{B}}, (\mathbf{emb}^{S})_{\bar{B}_{\mathsf{S}}})$. Similar with $\bar{B}_{\mathsf{S}}$ construction, we adopt the greedy strategy to gradually select $B \in \bar{B}_{\mathsf{T}}^{\mathsf{all}}$ with the minimum MMD.

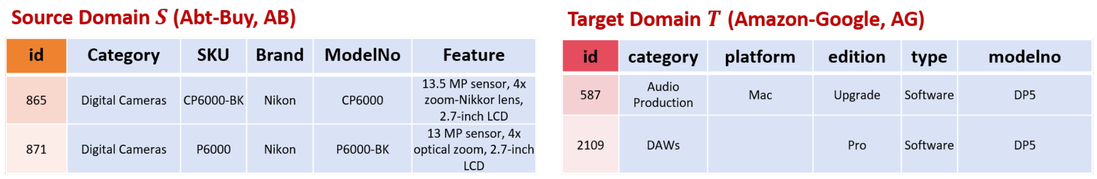
Example 2: Consider the dataset in Figure 1. We present the set of enriched attributes in Figure 4 that are gradually added, denoted as $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$, for the Abt-Buy and Amazon-Google datasets, respectively, generated by LLMs. □

V. 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

Recall that we fine-tune $\mathcal{M}$ in D^{S} using contrastive learning as described in Section III. To further adapt $\mathcal{M}$ to the target



Source Domain S (Abt-Buy, AB)

id	Category	SKU	Brand	ModelNo	Feature
865	Digital Cameras	CP6000-BK	Nikon	CP6000	13.5 MP sensor, 4x zoom-Nikkor lens, 2.7-inch LCD
871	Digital Cameras	P6000	Nikon	P6000-BK	13 MP sensor, 4x optical zoom, 2.7-inch LCD

Target Domain T (Amazon-Google, AG)

id	category	platform	edition	type	modelno
587	Audio Production	Mac	Upgrade	Software	DP5
2109	DAWs		Pro	Software	DP5

Fig. 4: Examples of LLM Enrichment with $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$ in source and target datasets

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.

(1) Self-supervised contrastive learning. Because the target training data D^{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_{aug} data augmentation strategies [15], *i.e.,* random insertion, deletion, and replacement of tokens in t, to generate L_{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 T, and we randomly sample L_{LLM} subsets from $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$ to generate a new set $\mathcal{S}$ of size L_{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_{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 [16], 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 KNN_t. Next, following [17], 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 $\frac{1}{K}$.

```
Input: The training data $D^{\mathsf{S}}$, $D^{\mathsf{T}}$, $\mathcal{R}_l^{\mathsf{T}}$, $\mathcal{R}_r^{\mathsf{T}}$, a LLM and the number
        of iteration iter.
Output: the fine-tuned representation model $\mathcal{M}$.
1.  $\mathcal{M} := \emptyset$
    // (1) warmup step
2.  Fine-tune $\mathcal{M}$ using $\mathcal{L}_{\mathsf{CL}}$ of Equation 1 and $\mathcal{L}_{\mathsf{adv}}$ of Equation 2;
    // (2) enrichment step
3.  Generate $\bar{B}_{\mathsf{S}}^{\mathsf{all}}$ and $\bar{B}_{\mathsf{T}}^{\mathsf{all}}$ using LLM and impute $\mathcal{R}_l^{\mathsf{T}}$, and $\mathcal{R}_r^{\mathsf{T}}$;
4.  Select $\bar{B}_{\mathsf{S}}$ and $\bar{B}_{\mathsf{T}}$ as the enriched attributes for S and T;
    // (3) iteration step
5.  for each $e \in 1, \ldots, \mathsf{iter}$ do
6.      Generate $\mathcal{P}_t$ and $\mathcal{N}_t$ for each tuple $t \in \mathcal{R}_l^{\mathsf{T}} \cup \mathcal{R}_r^{\mathsf{T}}$
          by leveraging the representation capabilities of LLM;
7.      Fine-tune $\mathcal{M}$ using self-supervised contrastive learning;
8.      Find a set KNN from $\mathcal{R}_l^{\mathsf{T}} \times \mathcal{R}_r^{\mathsf{T}}$ as the coreset by
          retrieving the top-$K$ results $\mathsf{KNN}_t$ for $t \in \mathcal{R}_l^{\mathsf{T}}$;
9.      Randomly sample the pseudo positive set $\mathcal{C}^{\mathsf{T}}$ in KNN
          with probabilities $p$ according to their distances;
10.     Fine-tune $\mathcal{M}$ using supervised contrastive learning
          in $D^{\mathsf{S}}$ and $\mathcal{C}^{\mathsf{T}}$ with hard negative sampling;
11.     $e := e + 1$;
12. return $\mathcal{M}$.
```

Fig. 5: The workflow of UDAEB

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 [18], 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 [18].

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.

Figure 5 shows the pseudo-code of the cross-domain entity blocking framework. With inputs of the labeled source training tuple pairs D^{S} and unlabeled target training tuple pairs D^{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 T. In the beginning we start a simple but effective warmup process as follows. We first transform D^{S} into the set of triplets CL^{S} and fine-tune $\mathcal{M}$ using the contrastive learning (Equation 1) so that $\mathcal{M}$ could be adapted in S. Next a MLP-based discriminator D is added and we adopt the adversarial learning strategy (Equation 2) to further fine-tune $\mathcal{M}$ and D so that similarity representations of S and T are aligned together (line 2).

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 $\mathcal{R}_r^{\mathsf{T}}$ (line 3). 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 simutaneously 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 $\mathcal{M}$ (line 6). 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 (line 8 and 9). Finally, we fine-tune $\mathcal{M}$ using the supervised contrastive learning in D^{S} and $\mathcal{C}^{\mathsf{T}}$ (line 10). $\mathcal{M}$ is iteratively fine-tuned until the maximum iteration iter is reached.

VI. 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. Finally, we perform ablation studies.

A. Experimental Setup

We show the experimental settings and datasets.

Baselines. We compare with the state-of-the-art entity blocking methods.

- DeepBlocker [3]. DeepBlocker is a transformer based representation model that adopts the self-supervised learning strategy and cross-entropy loss function. To fair comparison, we integrate its self-curated training data from the target domain and labeled training instances from the source domain as the training data to fine-tune DeepBlocker.
- Sudowoodo [4]. Sudowoodo is a self-supervised and contrastive learning based framework that integrates entity matching and blocking with pseudo-label training instances. Here we only adopt its entity blocking component and also use the source labeled training instances.
- STransformer [9]. STransformer is a Bert based model that transforms sentences to their dense embedding vectors with various loss function. Here we fine-tune it in the labeled source domain and adopt the contrastive loss function that is the same with UDAEB.
- UniBlocker [19]. UniBlocker is a universal dense entity blocking system that is pre-trained in GitTables [20] and can be adapted in various domains without requiring domain specific fine-tuning.

We implement UDAEB using Pytorch 2.3 and transformer-based FlagEmbedding library [21]. we use `bge-large-en` as the pre-trained model for $\mathcal{M}$ and adopt Mistral-7B [22] 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 [23] 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 methods, including PC, PQ, and CSSR.

Datasets. Table I shows the statistical information of all benchmark datasets used in the experiments, including the domains, the number of tuple pairs, matched ones, and attributes. All these seven datasets are from DeepMatcher [24] and DADER [5], covering various domains, including products, music, citations, and movies. 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 [4], DeepBlocker [3] and UniBlocker [19], 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 $|\mathcal{R}_l^{\mathsf{T}}| \times |\mathcal{R}_r^{\mathsf{T}}|$. PC and PQ are reported in 100-scale.

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 Table II, we present the accuracy of UDAEB in comparison to state-of-the-art entity blocking methods, following the evaluation standards outlined in [19]. We set the PC threshold at 90% and compare the first values of K that each baseline method can achieve. If PC of baselines remains below the threshold when $K = 20$, we report their results for $K = 20$. In other words, if a method can achieve the smallest K with a recall rate of at least 90%, we consider it effective.

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

Dataset	Domain	# All Pair	# Match Pair	# of Original Attributes	# of Attributes after Enrichment	# $\lvert\mathcal{R}_l\rvert, \lvert\mathcal{R}_r\rvert$
Abt-Buy (AB) [24]	Product	9,575	1,028	3	8	1081,1092
Walmart-Amazon (WA) [24]	Product	10,242	962	5	9	2554,22074
Amazon-Google (AG) [24]	Product	11,460	1,300	3	9	1363,3226
iTunes-Amazon (IA) [24]	Music	539	132	3	6	6907,55932
DBLP-ACM (DA) [24]	Citation	12,363	2,224	4	6	2616,2294
DBLP-Scholar (DS) [24]	Citation	28,707	5,347	4	6	2616,64263
RottenTomatoes-IMDB (RI) [5]	Movies	600	190	3	5	557,554

TABLE II: Performance Evaluation. Following UniBlocker [19], we report the first results (PC, PQ and K) of baselines when their PC exceeds the threshold (90%). If both methods have larger PC than the threshold, we evaluate K, otherwise we evaluate their PC. If their K are the same, we evaluate their PC and PQ. A higher PC, PQ and a lower K indicates better performance.

Methods	DeepBlocker		Sudowoodo		STransformer		UniBlocker		UDAEB	
Datasets	PC / PQ	K	PC / PQ	K	PC / PQ	K	PC / PQ	K	PC / PQ	K
WA-AG	85.69 / 3.67	20	90.06 / 9.80	8	91.60 / 15.69	5	90.38 / 17.24	5	**92.46 / 19.80**	**4**
AB-AG	85.69 / 3.67	20	90.83 / 11.29	7	91.52 / 13.06	6	90.38 / 17.24	5	**90.83 / 25.92**	**3**
WA-AB	75.19 / 3.57	20	90.37 / 28.73	3	74.32 / 3.53	20	93.16 / 31.53	3	**96.11 / 45.69**	**2**
AB-WA	90.12 / 3.39	10	90.54 / 8.53	4	86.38 / 1.63	20	93.33 / 14.06	3	**92.83 / 17.48**	**2**
RI-WA	90.12 / 3.39	10	44.91 / 0.92	20	77.43 / 3.68	20	93.33 / **14.06**	3	**95.43** / 11.98	3
RI-AB	75.19 / 3.58	20	71.01 / 4.95	20	77.43 / 3.68	1	93.16 / 31.53	3	**95.53 / 45.42**	**2**
IA-DA	97.21 / 82.49	1	98.92 / 84.92	1	97.03 / 82.34	1	99.19 / **84.33**	1	**99.23** / 84.21	1
IA-DS	90.14 / 18.43	10	90.24 / 12.30	15	91.17 / 26.62	7	**91.15 / 31.19**	**6**	91.19 / 26.63	7
Average	86.17 / 15.27	13.88	83.36 / 20.18	9.75	85.22 / 18.79	10.0	93.01 / 30.15	3.63	**94.20 / 34.64**	**3.0**

Furthermore, if two methods with PC values exceeding 90% have the same K, we compare their PC and PQ values, placing greater emphasis on PC. This emphasis is justified because, for the same K, the candidate set size is identical, leading to nearly equivalent running costs for the subsequent entity matching process. Additionally, a higher PC is more significant as it indicates the retrieval of more true positive results. The former 4 $\mathsf{S} \rightarrow \mathsf{T}$ pairs(*e.g.,* WA-AG) indicate similar domains, and the latter(*e.g.,* RI-WA) indicate different domains.

Among all baselines, UDAEB consistently outperforms others. For instance, its average PC and PQ are 7.26% and 13.54% higher than other baselines on average, respectively, with improvements reaching up to 10.84% and 19.37%. This underscores the effectiveness of the entity blocking framework, which integrates features such as similarity alignment between source and target domains, enrichment through LLMs, and iterative contrastive learning based on LLMs and high-quality pseudo-labels. UniBlocker achieves the second-best performance across most cases, benefiting from its pre-training in GitTables [20] and its role as a versatile blocker adaptable to various domains.

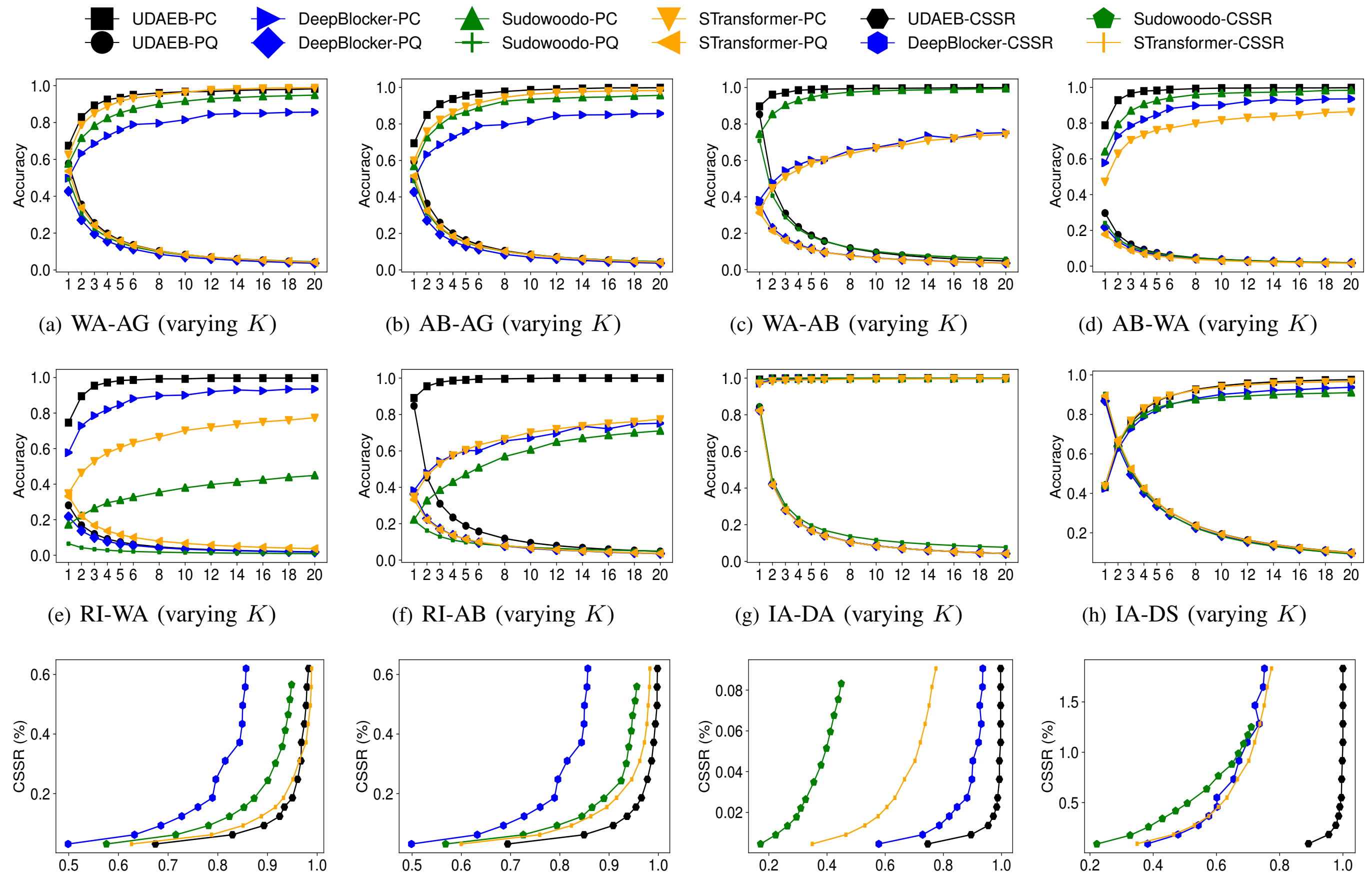
In Figure 6(a) to 6(h), where we vary K from 1 to 20, we display the PC and PQ of the baselines. The curve approaching the **upper left corner** of the figure indicates better performance. Note that UniBlocker learns from extra knowledge and requires significant pre-training costs for model fine-tuning, which other baselines do not incur. Therefore, for a fair comparison, we exclude UniBlocker from our evaluation. 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.

C. Candidate Set of Entity Blocking

In Figures 6(i) to 6(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. Ablation Study

We present the results of the ablation study in Table III, comparing three variants, *i.e.,*UDAEB without warmup, enrichment, and iteration steps in similar domain AB-AG and different domain RI-WA respectively. The results reveal that all variants exhibit lower accuracy in terms of PC and PQ in most cases, underscoring the effectiveness of each component. Notably, the iteration step demonstrates the largest impact, with PC decreasing from 74.64 to 61.35 in its absence. This further supports the assertions made in this paper regarding the benefits of cross-domain entity blocking, leveraging both self-supervised and supervised contrastive learning based on pseudo-labels and LLMs.



(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. 6: 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 III: Ablation Study

Datasets	Methods	PC / PQ ($K = 1$)	PC / PQ ($K = 3$)	PC / PQ ($K = 5$)	PC / PQ ($K = 7$)
AB-AG	UDAEB	**69.41 / 59.43**	**90.83 / 25.92**	**95.46 / 16.35**	**97.78 / 11.92**
	w.o. warmup	67.95 / 58.18	89.46 / 25.53	93.83 / 16.07	95.97 / 11.74
	w.o. enrichment	67.87 / 58.11	88.69 / 25.31	94.17 / 16.13	96.83 / 11.84
	w.o. iteration	68.38 / 58.55	90.23 / 25.75	94.94 / 16.26	96.66 / 11.82
RI-WA	UDAEB	74.64 / 28.11	**95.43 / 11.98**	**98.34 / 7.41**	**99.06 / 5.33**
	w.o. warmup	69.54 / 26.19	91.89 / 11.53	95.84 / 7.22	97.61 / 5.25
	w.o. enrichment	**74.74 / 28.15**	93.66 / 11.76	98.02 / 7.38	98.54 / 5.30
	w.o. iteration	61.35 / 23.14	83.89 / 10.53	89.81 / 6.77	93.24 / 5.02

E. *Varying Different Models*

We vary different representation models $\mathcal{M}$ and LLMs to evaluate the accuracy of UDAEB. In this part, we only report PC at $K = 1$, *i.e.,* top-1 recall, since top-1 recall is crucial for user satisfaction. (Here M means million and B for billion.)

We vary different parameter size of embedding models in Table IV, including `bge-small` [25], `all-mpnet` [26], `bge-large` [27], and `bge-m3` [28] of 33.4M, 109M, 335M and 561M parameters, respectively. Also we adopt different LLMs for data enrichment (Table V), including `Llama-3.1-8B` [29] and `Phi-3.5-3.8B` [30].

As shown in Table IV and V, UDAEB is not very sensitive with different $\mathcal{M}$, *e.g.,* 70.48%, 71.00%, 76.51% and 79.00% PC for `bge-small`, `all-mpnet`, `bge-large` and `bge-m3`, respectively, verifying the effectiveness of the unsupervised domain adaptation entity blocking framework. The design of the domain adaptation module effectively compensates for the limitations in representational capability of small embedding and enrichment models, such as `bge-small` with only 33.5M parameters. This approach ensures that UDAEB is robust and feasible for real-world scenario deployment, offering high inference speed with minimal performance loss.

This highlights the efficacy of adopting a domain adaptation module in our methodology, enabling efficient deployment applications even with small models in practical settings.

When adopting different LLMs for enrichment, UDAEB has slightly higher accuracy with smaller models, *e.g.,*77.75% top-1 PC in `Phi-3.5` on RI-WA. This is contributed to the denser parameter distribution and extended context length input in these new models. However, we argue that by applying KV cache and batched query strategy with integrating vLLM [23], selecting `Mistral-7B` achieves a trade-off in enrichment performance and generation speed in UDAEB. *e.g.,*, it can enrich 10,000 tuple pairs in 437 seconds, 38.96% and 48.89% faster than `Phi-3.5` and `Llama-3.1-8B` in same settings respectively.

TABLE IV: Varying Different $\mathcal{M}$ (PC / PQ, $K = 1$)

Datasets	`bge-small`	`all-mpnet`	`bge-large`	`bge-m3`
RI-AB	85.12 / 80.94	86.28 / 82.05	**89.10 / 84.73**	86.38 / 82.15
RI-WA	70.48 / 26.55	71.00 / 26.74	76.51 / 28.82	**79.00 / 29.76**
WA-AG	66.58 / 57.01	65.12 / 55.76	**67.27 / 57.59**	66.67 / 57.08
AB-AG	66.50 / 56.93	66.24 / 56.71	66.84 / 57.23	**68.55 / 58.69**

TABLE V: Varying Different LLMs (PC / PQ, $K = 1$)

Datasets	`Mistral-7B`	`Llama-3.1-8B`	`Phi-3.5-3.8B`
RI-WA	76.51 / 28.82	76.30 / 28.74	**77.75 / 29.29**
RI-AB	**89.10 / 84.73**	79.77 / 75.86	79.47 / 75.58
WA-AG	67.27 / 57.59	67.69 / 57.96	**68.29 / 58.47**
AB-AG	**69.40 / 59.42**	67.69 / 57.96	66.84 / 57.23
AB-WA	78.79/ 29.67	79.11 / 29.80	**79.42 / 29.91**
WA-AB	89.68/ 85.27	**90.76 / 86.31**	90.08 / 85.66

F. The Training Cost

We evaluate the training time of different baselines in Table VI. Since UniBlocker requires huge pre-training cost, we do not report their training cost here. 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.

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

Datasets	UDAEB ($\mathsf{T}_{\mathsf{enrich}}$ + 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

VII. Related Work

A. Entity Blocking

Entity blocking is an important step of entity resolution that aims to reduce the number of potential comparisons in large-scale datasets. We classify it 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 [31]. More effective methods have been proposed, including meta-blocking [32], matching dependencies [33], and DNF-based methods [34], 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 [35], BSL [36], and EM-GBF [37]. Sparkly [38] employs the tf/idf blocking technique to achieve high efficiency and effectiveness. (2) **ML-based models**. These methods learn ML classifiers to make inferences efficiently, such as Smurf [39] and supervised meta-block [40]. 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. DeepER [41] learns tuple representations with cross-entropy loss and uses LSH for blocking. STransformer [9] is a Siamese Bert model that generates tuple embeddings with various loss functions. DeepBlocker [3] proposed several training data generation strategies and defined the network architecture for entity blocking. Recently, representation models fine-tuned with contrastive loss have achieved significant performance improvements, *e.g.,* simCLR [42] and SuperCon [43]. Building on these, several entity blocking models have been proposed, such as SC-Block [44], Sudowoodo [4], STransformer [9] with contrastive loss, and UniBlocker [19], which design entity blocking models with contrastive learning and propose high-quality training data.

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 automatically 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 [45]; (2) loss-centric methods, *e.g.,* CGANS [12]; (3) pseudo-labeling techniques, *e.g.,* [46]; (4) data selection methods, *e.g.,* [47]; and (5) pre-training methods, *e.g.,* [48]. For more details, refer to the survey [49].

In entity resolution, unsupervised domain adaptation has been developed for entity matching, including DADER [5], MFSN [6], TL-ER [50], TransER [51] and MELD [8]. 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.

VIII. 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 to improve performance. 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.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[2] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Trans. Big Data*, 2021.

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

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

[12] 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.

[13] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on neural networks*, vol. 20, no. 2, pp. 189–201, 2009.

[14] W. Wang, H. Li, Z. Ding, and Z. Wang, "Rethink maximum mean discrepancy for domain adaptation," *arXiv preprint arXiv:2007.00689*, 2020.

[15] 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.

[16] 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.

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

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

[19] 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.

[20] M. Hulsebos, Ç. Demiralp, and P. Groth, "Gittables: A large-scale corpus of relational tables," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 30:1–30:17, 2023.

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

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

[23] 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.

[24] 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.

[25] *https://huggingface.co/BAAI/bge-small-en-v1.5.*

[26] *https://huggingface.co/sentence-transformers/all-mpnet-base-v2.*

[27] *https://huggingface.co/BAAI/bge-large-en-v1.5.*

[28] *https://huggingface.co/BAAI/bge-m3.*

[29] https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct.

[30] https://huggingface.co/microsoft/Phi-3.5-mini-instruct.

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

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

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

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

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

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

[37] 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.

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

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

[40] 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.

[41] 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.

[42] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.

[43] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," 2021. [Online]. Available: https://arxiv.org/abs/2004.11362

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

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

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

[47] 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.

[48] 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.

[49] A. Ramponi and B. Plank, "Neural unsupervised domain adaptation in NLP - A survey," in *COLING*, D. Scott, N. Bel, and C. Zong, Eds., 2020.

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

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