



Blocker and Matcher Can Mutually Benefit: A Co-Learning Framework for Low-Resource Entity Resolution

Shiwen Wu
The Hong Kong University of Science
and Technology
swubs@connect.ust.hk

Qiyu Wu
The University of Tokyo
qiyuw@g.ecc.u-tokyo.ac.jp

Honghua Dong
University of Toronto & Vector
Institute
honghuad@cs.toronto.edu

Wen Hua
The Hong Kong Polytechnic
University
wency.hua@polyu.edu.hk

Xiaofang Zhou
The Hong Kong University of Science
and Technology
zxf@ust.hk

ABSTRACT

Entity resolution (ER) approaches typically consist of a blocker and a matcher. They share the same goal and cooperate in different roles: the blocker first quickly removes obvious non-matches, and the matcher subsequently determines whether the remaining pairs refer to the same real-world entity. Despite the state-of-the-art performance achieved by deep learning methods in ER, these techniques often rely on a large amount of labeled data for training, which can be challenging or costly to obtain. Thus, there is a need to develop effective ER systems under low-resource settings. In this work, we propose an end-to-end iterative Co-learning framework for ER, aimed at jointly training the blocker and the matcher by leveraging their cooperative relationship. In particular, we let the blocker and the matcher share their learned knowledge with each other via iteratively updated pseudo labels, which broaden the supervision signals. To mitigate the impact of noise in pseudo labels, we develop optimization techniques from three aspects: label generation, label selection and model training. Through extensive experiments on benchmark datasets, we demonstrate that our proposed framework outperforms baselines by an average of 9.13-51.55%. Furthermore, our analysis confirms that our framework achieves mutual benefits between the blocker and the matcher.

PVLDB Reference Format:

Shiwen Wu, Qiyu Wu, Honghua Dong, Wen Hua, and Xiaofang Zhou. Blocker and Matcher Can Mutually Benefit: A Co-Learning Framework for Low-Resource Entity Resolution. PVLDB, 17(3): 292 - 304, 2023. doi:10.14778/3632093.3632096

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at <https://github.com/wusw14/CLER>.

1 INTRODUCTION

Entity Resolution (ER) determines whether two data entries refer to the same real-world entity. This long-standing problem has received

much attention [2, 18, 22, 24, 25, 33, 42]. Because of the increasing dataset size in practice, an exhaustive pairwise evaluation of all pairs of entries is almost unaffordable [42]. As a result, most existing ER systems follow a two-step process consisting of a *blocking* step and a *matching* step [18, 31]. During the blocking phase, a *blocker* quickly prunes the entry pairs that are unlikely to match, thereby greatly reducing the number of candidate pairs [42]. Subsequently, a *matcher* serves as a more accurate binary classifier that determines whether entries are matched [39].

A successful ER system generally requires both the blocker and the matcher to be effective. The state-of-the-art (SOTA) results are achieved by deep learning-based (DL-based) methods. DL-based blockers map each entry into embedding space, and employ the vector pairing strategy to retrieve potential matches [31]. DL-based matchers take a pair of entries as input and predict its probability of being a match [12, 18]. However, DL-based methods always require a great amount of labeled data for training [15, 33, 38]. The high human cost in labeling has become a primary bottleneck in adopting advanced DL-based approaches in real-world applications [38].

In light of this, recent efforts pay attention to low-resource ER where the annotation budget is limited. One mainstream direction is to broaden the limited supervision signals by incorporating additional information [12, 15, 33]. For example, CollaborEM [12] utilizes the semantic knowledge from pre-trained models for pseudo labels. DADER [33] explores transferring the knowledge from well-labeled source ER datasets to target datasets and figures out that a “close” source domain is beneficial. However, the information utilized by these methods is external, while the external information with very similar semantics to the target dataset is not always available. In fact, besides such external information, there is another type of easily available intrinsic information within an ER system that can be utilized, *i.e.*, the blocker and the matcher naturally provide extra information to each other. As shown in Figure 1(a), the matcher is a fine-grained classifier for entry pairs, and its precise classification results could calibrate the blocker’s similarity judgment to be more accurate. On the other hand, the blocker searches for potential matches by similarity-based pairing in the entire dataset, which could provide information about the similarity ranking from a global perspective to the matcher. Inspired by the Co-training mechanism [1, 3], a straightforward solution is to allow the blocker and the matcher to learn from each other beyond the limited annotated data, which is yet underexplored.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 17, No. 3 ISSN 2150-8097. doi:10.14778/3632093.3632096

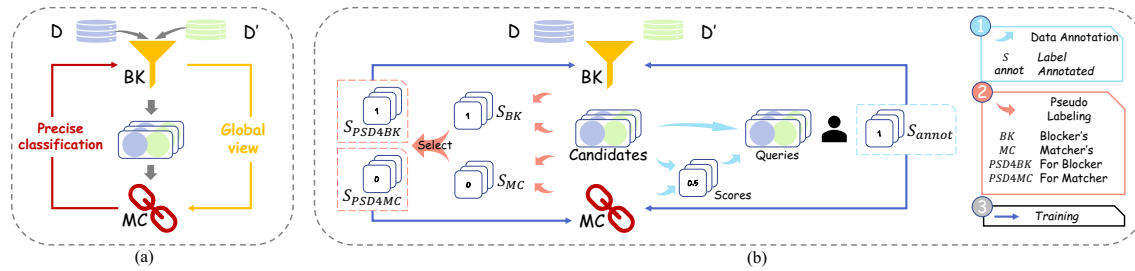


Figure 1: (a) Illustration of the Co-learning between the blocker and the matcher in terms of information breadth and prediction accuracy. The blocker learns from the matcher’s precise classification ability while the matcher learns from the blocker’s global view of the similarity ranking. The gray arrows represent the data flow. (b) The overview of our CLER framework in one training iteration, containing three steps (1) Data Annotation: The blocker (BK) first produces a candidate set C from all pairs of entities (e, e') where $e \in D$ and $e' \in D'$. The matcher (MC) then generates scores for each candidate, which are used to select informative examples to be annotated. (2) Pseudo-labeling: The blocker and the matcher generate pseudo labels for C separately. The generated ones are further processed into two sets feeding the blocker and the matcher, respectively. (3) Training: Both the annotated data S_{annot} and the pseudo-labeled data are utilized for training the blocker and the matcher.

To this end, we propose an end-to-end iterative Co-Learning framework for low-resource Entity Resolution (called CLER). Our framework enables iterative updates of the blocker and the matcher during the training stage, leveraging both annotated data and the knowledge distilled from each other to enhance the system’s performance, as illustrated in Figure 1(b). To be specific, we let their knowledge be distilled in the form of pseudo-labeled data, which is a commonly adopted approach in weakly supervised learning [3, 45]. This can significantly reduce the model’s reliance on limited annotations. However, pseudo labels are inevitably noisy, which could affect the performance of our CLER model. In this work, we attempt to mitigate the impact of label noise from three aspects: label generation, label selection, and model training.

- (1) How to generate high-quality pseudo labels? We design pseudo-labeling strategies with confidence awareness for the blocker and the matcher, respectively. We further leverage the transitivity constraint of data entries to improve the quality of the pseudo labels.
- (2) How to select pseudo-labeled data from the generated data for training? As the quality of the pseudo-labels generated by the blocker and the matcher can differ, we adopt different label selection strategies for each model. Specifically, the blocker accepts part of the high-confidence pseudo-labeled data generated by the matcher and discards the pseudo-labels generated by itself, while the matcher integrates pseudo-labeled data from both models. This approach allows the matcher to learn from the blocker and avoid forgetting what it has already learned.
- (3) How to train the models by incorporating the annotated and pseudo-labeled data? Annotated data is always considered to be absolutely reliable, while pseudo-labeled data brings noises along with information. Thereby, we employ a re-weighting strategy to balance the contribution of the annotated data and pseudo-labeled data to the training of the blocker and the matcher.

The key contributions of this work are summarized as follows:

- We investigate the potential advantage of utilizing a cooperative relationship between the blocker and the matcher in the ER system. Empirical results confirm that both the blocker and the matcher can benefit from each other. This observation suggests

that the cooperation between the blocker and the matcher should be taken into account in ER systems.

- We present CLER, a novel end-to-end ER framework for low-resource settings, which allows the blocker and the matcher to iteratively learn from each other, in addition to learning from the annotated data. In order to realize a Co-learning mechanism, we let their learned knowledge to be distilled in the form of pseudo-labeled data. Correspondingly, we propose several designs for better pseudo-label generation and utilization, thus mitigating the impact of label noise.
- We conduct extensive experiments on the Magellan datasets [16] and Alaska benchmark [7]. Experimental results demonstrate that CLER outperforms the previous SOTA by a 9.13-51.55% improvement on average in all datasets, which empirically shows the effectiveness and superiority of our framework.

Outline. Section 2 gives a formal definition of low-resource ER and overviews the relevant background knowledge. Section 3 describes the overall CLER framework along with our designs to mitigate the impact of label noise. Section 4 discusses the inference strategy. Our experimental results are presented in Section 5. We review related work in Section 6 and conclude in Section 7.

2 PRELIMINARIES

In this section, we first give a formal definition of the low-resource ER task, and then describe the pre-trained language models (LMs) used by our blocker and matcher, how a DL-based blocker works in the ER system, and how to fine-tune the LM as the matcher.

2.1 Problem Setting

The objective of entity resolution (ER) is to identify pairs of data entries (e, e') that refer to the same real-world entity, where e and e' belong to two collections of data entries D and D' , respectively. These pairs are commonly referred to as matches [22]. Following [15, 18, 31], we assume that the data entries in both D and D' have identical schema. Each data entry is represented as a set of

attribute-value pairs, i.e., $e = (\text{attr}_i, \text{val}_i)_{1 \leq i \leq k}$, where val_i represents the corresponding value of attribute attr_i . Since the number of all pairs $|D| \times |D'|$ could be enormous, most ER systems adopt the *blocker-matcher* workflow. The blocker generates a small candidate subset of all pairs with high recall, and the matcher classifies them as matches or non-matches.

This work considers a more practical low-resource setting for ER problems, where only limited data annotation is allowed. Specifically, we can only choose B pairs of data entries to be labeled (as matches or non-matches), where B is the budget. This significantly challenges the training of the blocker and matcher [15, 33].

Task Definition: Given two collections of data entries D and D' , the task of low-resource entity resolution is to identify those true matches (e, e') under the constraint of B annotations.

2.2 Using Pre-trained Language Models (LMs)

Inspired by the superiority of pre-trained LMs in semantic understanding, we apply them in both the blocker and the matcher. In the following, we introduce pre-trained LMs and describe the entry encoding and the outputs of LMs.

Language models. Recent advanced LMs are typically a stack of Transformer [34] layers pre-trained over a large corpus using self-supervised tasks [8, 19, 29]. After pre-training, they demonstrate excellent contextual extraction and semantic understanding capabilities, benefiting various downstream tasks. Some recent attempts have already utilized LMs for the ER task, e.g., DITTO [18] uses LMs for the matcher.

Entry encoding. Following [18, 23], we encode the data entry into a sequence of tokens that LMs can process. Specifically, for a data entry $e = \{(\text{attr}_i, \text{val}_i)\}_{1 \leq i \leq k}$, we serialize it as follows:

$$\text{serialize}(e) ::= [\text{COL}]\text{attr}_1[\text{VAL}]\text{val}_1 \dots [\text{COL}]\text{attr}_k[\text{VAL}]\text{val}_k,$$

where [COL] and [VAL] are special tokens indicating the start of attribute names and values, respectively. Before being processed by LMs, a special token [CLS] is added to the beginning of the serialized sequence, i.e., $\text{serialize}^*(e) ::= [\text{CLS}]\text{serialize}(e)$.

For a pair of entries (e, e') , we serialize them separately and add another special token [SEP] to separate them:

$$\text{serialize}^*(e, e') ::= [\text{CLS}]\text{serialize}(e)[\text{SEP}]\text{serialize}(e').$$

Then the LMs take the serialized entries as a natural language sentence, and the tokenizer of the pre-trained LMs is applied to obtain a sequence of input tokens.

Output of LMs. The raw output of pre-trained LMs is a sequence of contextualized embeddings of the same length as the input tokens. Following [8, 18, 30], we take the embedding corresponding to the [CLS] token as the representation of the whole sentence and define the LM embedding function as below:

$$\begin{aligned} f_{LM}(e) & ::= LM(\text{serialize}^*(e))_{[\text{CLS}]}, \\ f_{LM}(e, e') & ::= LM(\text{serialize}^*(e, e'))_{[\text{CLS}]}. \end{aligned}$$

2.3 Blocker: Entry Embedding Learning and Similarity-based Pairing

The key idea of the DL-based blocker is to convert each entry into an embedding vector, then quickly retrieve the entry pairs with a high similarity score between their vectors [31]. To be specific, the

Algorithm 1: KNN-Blocking(M_{BK}, K, D, D')

Input: a blocker M_{BK} ; K , the number of the most similar entries; two collections of data entries, D and D'

Output: A set of candidate pairs C

```

1 Initialize:  $C \leftarrow \phi$ 
2  $\{\mathbf{h}_i\} \leftarrow \{f_{BK}(e_i) \mid e_i \in D\}$ 
3  $\{\mathbf{h}'_i\} \leftarrow \{f_{BK}(e'_i) \mid e'_i \in D'\}$ 
4 for each entry  $e$  in  $D$  do
5    $N_e \leftarrow \text{K-Most-Similar}_{e' \in D'}(\mathbf{h}_e, \{\mathbf{h}'_i\}, D', K)$ 
6    $C \leftarrow C \cup \{(e, e') \mid e' \in N_e\}$ 
7 return  $C$ 

```

blocker takes two sets of entries D and D' as inputs and maps them into corresponding sets of entry embeddings:

$$\{\mathbf{h}_i\} = \{f_{BK}(e_i) \mid e_i \in D\}, \quad \{\mathbf{h}'_i\} = \{f_{BK}(e'_i) \mid e'_i \in D'\},$$

where $f_{BK}(\cdot)$ is the embedding mapping function of the blocker. The optimization goal of a DL-based blocker is to learn a good mapping function $f_{BK}(\cdot)$, which is one of the focuses of this work (refer to Section 3.4.1).

Based on the computed embeddings, a similarity-based pairing strategy [31, 42] is leveraged to retrieve the candidate pairs of entries. Following [31], we adopt a cosine function to measure the similarity between two entries (e, e') , i.e.,

$$\text{sim}(e, e') = \cos(f_{BK}(e), f_{BK}(e')) = \cos(\mathbf{h}, \mathbf{h}') = \frac{\mathbf{h}^T \mathbf{h}'}{\|\mathbf{h}\| \|\mathbf{h}'\|},$$

where a larger value means two entries are more similar. One commonly adopted approach for candidate pair generation is the K -nearest neighboring (KNN) method, i.e., to keep K most similar entries in D' for each entry $e \in D$, as outlined in Algorithm 1. Consequently, the size of the candidates C is $|D| \times K$ in total.

2.4 Matcher: Pairwise Binary Classifier by Fine-tuning LMs

In ER systems, the matcher plays as a pairwise binary classifier. Specifically, the matcher takes a pair of entries (e, e') as the input, and outputs the probability of the pair of entries being a match:

$$\hat{y} = g_{MC}^m(e, e'),$$

where g_{MC} is a binary classifier for a pair of entries, and its predicted probability of being a match is taken as the score of the entry pair, denoted as $g_{MC}^m(\cdot)$.

In this work, we fine-tune a pre-trained model RoBERTa, to realize the function $g_{MC}(\cdot)$. Please refer to the original work for implementation details [19]. Following previous works [8, 18], we append a one-layer MLP head (linear layer) along with softmax to the embedding of the pairwise entries. During the fine-tuning stage, all the parameters are updated jointly based on the labeled data. We define the function for matching as:

$$g_{MC}(e, e') ::= \text{softmax}(\text{MLP}(f_{LM}(e, e'))),$$

where e and e' are a pair of entries to be tested, respectively. By doing so, we are able to implement the matcher by fine-tuning the pairwise entry embedder and the MLP head over labeled data.

Algorithm 2: The CLER Framework

Input: Two collections of data entries, D and D' ; training budget B ; budget for each iteration b ; validation set S_{valid} ; K , the number of the most similar entries; p , percentile.

Output: The blocker M_{BK} , and the matcher M_{MC} .

- 1 **Initialization:** Initialize the blocker M_{BK} and matcher M_{MC} ; annotated data $S_{annot} \leftarrow \phi$
- 2 **# Prepration**
- 3 $C \leftarrow \text{KNN-Blocking}(M_{BK}, K, D, D')$
- 4 $S_{BK} \leftarrow \text{PseudoLabelByBlocker}(M_{BK}, C, S_{valid}, p)$
- 5 $M_{MC} \leftarrow \text{UpdateMatcher}(M_{MC}, S_{BK})$
- 6 **while** $B > 0$ **do**
- 7 **# Annotate the samples**
- 8 $S_{annot} \leftarrow \text{Annotation}(M_{MC}, C, b, S_{annot})$
- 9 $B \leftarrow B - b$
- 10 **# Update the blocker**
- 11 $S_{MC} \leftarrow \text{PseudoLabelByMatcher}(M_{MC}, C, S_{valid})$
- 12 $S_{PSD4BK} \leftarrow \text{SelectPseudoLabelForBK}(S_{MC})$
- 13 $M_{BK} \leftarrow \text{UpdateBlocker}(M_{BK}, S_{annot}, S_{PSD4BK})$
- 14 $C \leftarrow \text{KNN-Blocking}(M_{BK}, K, D, D')$
- 15 **# Update the matcher**
- 16 $S_{BK} \leftarrow \text{PseudoLabelByBlocker}(M_{BK}, C, S_{valid}, p)$
- 17 $S_{MC} \leftarrow \text{PseudoLabelByMatcher}(M_{MC}, C, S_{valid})$
- 18 $S_{PSD4MC} \leftarrow \text{SelectPseudoLabelForMC}(S_{BK}, S_{MC})$
- 19 $M_{MC} \leftarrow \text{UpdateMatcher}(M_{MC}, S_{annot}, S_{PSD4MC})$

3 THE CLER FRAMEWORK

In this section, we first give an overview of our CLER framework. To realize Co-learning between the two models, we let their knowledge be distilled into pseudo-labeled data and propose corresponding designs to address the impact of label noise. Specifically, we introduce our pseudo-label generation strategies in Section 3.2, pseudo-label selection strategies in Section 3.3, and the utilization of the pseudo-labeled data and annotated data for training in Section 3.4.

3.1 Overview of CLER

Under the constraint of a limited annotation budget, we propose an end-to-end iterative Co-Learning framework for Entity Resolution tasks (called CLER, illustrated in Figure 1(b)). Our framework exploits the cooperative relationship between the blocker and the matcher to better utilize the annotation budget. In addition to learning from the annotated data, we devise a Co-learning mechanism that allows the blocker and the matcher to learn from each other via iteratively updated pseudo-labeled data. We summarize the overall pipeline in Algorithm 2 and introduce the main steps in the following.

Preparation stage. In this stage, we adopt pre-trained models to warm up the blocker and the matcher. In terms of the blocker, we employ the `stsb-roberta-base`¹ version of SBERT [30], as the initial blocker, whose efficacy has been demonstrated in previous ER literature [12, 18]. The blocker generates candidate pairs, denoted

¹<https://huggingface.co/sentence-transformers/stsb-roberta-base>

Algorithm 3: PseudoLabelByBlocker(M_{BK}, C, S_{valid}, p)

Input: a blocker M_{BK} ; the candidates C ; validation set S_{valid} ; p^{th} percentile

Output: A set of labeled data S_{BK}

- 1 **Initialize:** $S_{BK} \leftarrow \phi$, $\text{diff} \leftarrow \phi$
- 2 **for each entry** e **in** D **do**
- 3 $\text{diff} \leftarrow \text{diff} \cup \{\min_{e'} \{\text{sim}(e, e') \mid (e, e', 1) \in S_{valid}\} - \max_{e'} \{\text{sim}(e, e') \mid (e, e', 0) \in S_{valid}\}\}$
- 4 $t_{\text{diff}} \leftarrow \text{Percentile}(\{d \mid d > 0, d \in \text{diff}\}, p)$
- 5 **for each candidate pair** (e, e') **in** C **do**
- 6 **if** $e' = \text{top}(e) \wedge e = \text{top}(e')$ **then**
- 7 $S_{BK} \leftarrow S_{BK} \cup (e, e', 1)$
- 8 **else**
- 9 **if** $\text{sim}(e, \text{top}(e)) - \text{sim}(e, e') > t_{\text{diff}}$ **and**
- 10 $e = \text{top}(\text{top}(e))$ **then**
- 11 $S_{BK} \leftarrow S_{BK} \cup (e, e', 0)$
- 11 **return** S_{BK}

as C , via similarity-based pairing, as introduced in Section 2.3. For the matcher, we fine-tune a pre-trained LM, `roberta-base`², using the method introduced in Section 2.4. We use the pseudo labels generated by the initial blocker to warm up the matcher. We provide details about the pseudo-label generation in Section 3.2.

Iteration stage. For the iterative training stage, we have a total budget of B for annotations, and allocate a budget of b to each iteration. In each iteration, the matcher selects b pairs of entries from candidates C for annotation. To select informative samples, we adopt an uncertainty-based active learning strategy [15, 40] that selects examples from the potential positive and negative samples with high entropy scores. We update the accumulated annotated set S_{annot} after each round of annotation. Afterwards, we first train the blocker and then train the matcher. For each model, we first generate up-to-date pseudo-labeled data (S_{MC} and S_{BK}), and then select data used for training from the generated ones. The model is trained with the annotated data S_{annot} and the selected pseudo-labeled data S_{PSD4BK} or S_{PSD4MC} . The candidate set C is renewed if the blocker is updated. We discuss the pseudo-labeled data selection in Section 3.3 and the training of the models utilizing both pseudo-labeled data and annotated data in Section 3.4. The iteration repeats until the total budget of B is exhausted.

3.2 Pseudo-label Generation

In the following, we first introduce the pseudo-labeling strategies used by the blocker (Section 3.2.1) and the matcher models (Section 3.2.2). We then describe how we leverage the transitivity property of ER to improve the quality of pseudo labels (Section 3.2.3).

3.2.1 Pseudo labeling by the blocker. Following previous works [5, 12], the blocker labels the candidate pairs according to the similarity scores between the entries. We use the same similarity function as in Section 2.3 and define most similar entry as:

$$\text{top}(e) = \arg \max_{e' \in D'} \text{sim}(e, e'), \quad \text{top}(e') = \arg \max_{e \in D} \text{sim}(e, e'). \quad (1)$$

²<https://huggingface.co/roberta-base>

The labeling process is depicted in Algorithm 3, where each candidate pair (e, e') goes through the following three steps:

- (1) If $e' = \text{top}(e)$ and $e = \text{top}(e')$, i.e., they are mutually most similar to each other, the blocker labels them as matches [5].
- (2) Else if e can find its mutually most similar entry and the difference between the similarity of $\text{sim}(e, e')$ and that of the most similar pair $\text{sim}(e, \text{top}(e))$ is greater than a threshold t_{diff} , the blocker labels the pair as a non-match.
- (3) In other cases, the blocker would ignore this pair as it is hard to decide whether they are matched.

We denote the generated pseudo-labeled data as S_{BK} . Considering that the distribution of similarity scores might change as training progresses, we let t_{diff} auto-adapt instead of being a static scalar. In this work, t_{diff} is the p^{th} percentile of the gap between the similarity scores of matches and non-matches in the validation set. Note that improving the quality of the blocker's entry embeddings through training would benefit the quality of the generated pseudo labels.

3.2.2 Pseudo labeling by the matcher. For each candidate pair (e, e') , the matcher first predicts its probability of being matches $g_{MC}^m(e, e')$. Then the most straightforward way is to regard the pairs of entries whose probabilities are above 0.5 as matches while the remaining ones as non-matches.

$$\begin{aligned} S_+ &= \{(e, e', 1) \mid (e, e') \in C, 0.5 < g_{MC}^m(e, e') \leq 1.0\}, \\ S_- &= \{(e, e', 0) \mid (e, e') \in C, 0.0 \leq g_{MC}^m(e, e') \leq 0.5\}. \end{aligned} \quad (2)$$

However, such a pseudo-labeling strategy will bring too many noisy labels, especially in the early stages of training. Instead of roughly using 0.5 as the cut-off point, we differentiate the reliability of the pseudo labels based on their predicted probabilities by introducing two thresholds t_+ and t_- that depend on the validation set. Specifically, we set t_+ as the median predicted score of the matches in the validation set and t_- similarly:

$$\begin{aligned} t_+ &= \max(0.5, \text{median}(\{g_{MC}^m(e, e') \mid (e, e', 1) \in D_{\text{valid}}\})), \\ t_- &= \min(0.5, \text{median}(\{g_{MC}^m(e, e') \mid (e, e', 0) \in D_{\text{valid}}\})). \end{aligned} \quad (3)$$

In this way, the thresholds auto-adapt as the training progresses.

Then we select the high-confidence (h) pairs according to their predicted probabilities as follows:

$$\begin{aligned} S_+^h &= \{(e, e', 1) \mid (e, e') \in C, t_+ \leq g_{MC}^m(e, e') \leq 1.0\}, \\ S_-^h &= \{(e, e', 0) \mid (e, e') \in C, 0.0 \leq g_{MC}^m(e, e') \leq t_-\}, \end{aligned} \quad (4)$$

and merge the high-confidence ones as S_{MC}^h and all pseudo-labeled ones as S_{MC} . Note that S_{MC}^h is a subset of S_{MC} and can be easily derived from S_{MC} with proper implementation.

$$S_{MC}^h = S_+^h \cup S_-^h, \quad S_{MC} = S_+ \cup S_-. \quad (5)$$

3.2.3 Transitivity property. The transitivity property for ER in a two-database setting states that if entry $e_i \in D$ matches $e'_i \in D'$, $e_j \in D$ matches $e'_j \in D'$, and $e_i \in D$ matches $e'_i \in D'$, we must conclude that e_j also matches e'_j [38]. A set that represents matches in ER should satisfy the transitivity property. The pseudo-labeling strategy of S_{BK} only labels mutually most similar pairs to be matches, ensuring that each entry appears at most once in the matches, making it naturally satisfying the transitivity property.

Algorithm 4: Transitivity Checking

Input: D and D' ; The S containing matches to be checked.
Output: The S^* containing matches follow the transitivity.

- 1 **Initialize:** $S^* \leftarrow \phi$
- 2 **for each entry e in D do**
- 3 $\lfloor \text{matches}_e \leftarrow \{e' \mid (e, e', 1) \in S\}$
- 4 **for each entry e' in D' do**
- 5 $\lfloor \text{matches}_{e'} \leftarrow \{e \mid (e, e', 1) \in S\}$
- 6 **for each entry e in D do**
- 7 $u_e \leftarrow \bigcup_{e' \in \text{matches}_e} \text{matches}_{e'}$
- 8 $v_e \leftarrow \bigcap_{e \in u} \text{matches}_e$
- 9 $S^* \leftarrow S^* \cup \{(e, e', 1) \mid e' \in v_e\}$
- 10 **return** S^*

For other sets of matches, we devise a transitivity-checking algorithm depicted in Algorithm 4 to ensure that the matches follow the transitivity property.

THEOREM 1. *The returned set of matches (S^*) satisfies the transitivity property if the intermediate value v satisfies $v_x = v_y$ or $v_x \cap v_y = \phi$ for all $x, y \in D$.*

PROOF. We build a bipartite graph where edges are the matches in S^* . Two nodes $x, y \in D$ are one-step connected if $v_x \cap v_y \neq \phi$, which means $v_x = v_y$. Therefore, two nodes $x, y \in D$ are connected if and only if $v_x = v_y$. Then for each connected component, all $x \in D$ in this connected component have the same value of v , forming a fully connected bipartite sub-graph that trivially satisfies the transitivity property. \square

THEOREM 2. *The intermediate value v in the algorithm satisfies $v_x = v_y$ or $v_x \cap v_y = \phi$ for all $x, y \in D$.*

PROOF. Let $x, y \in D$ be two entries. **(1)** First consider the case $\text{matches}_x \cap \text{matches}_y = \phi$, which indicates $v_x \cap v_y = \phi$ as $v_x \subseteq \text{matches}_x$ and $v_y \subseteq \text{matches}_y$. **(2)** Otherwise, the intersection is not empty, which means $y \in u_x$ and $x \in u_y$. **(2a)** When $u_y = u_x$, we can conclude $v_y = v_x$ as v is computed based on u . **(2b)** We then consider the case $u_x \setminus u_y \neq \phi$, which means $\exists e \in u_x$ such that $\text{matches}_e \cap \text{matches}_y = \phi$. As $y \in u_x$ and $e \in u_x$, we conclude that $v_x = \phi$. **(2c)** The case $u_y \setminus u_x \neq \phi$ is similar, and we can conclude that $v_y = \phi$. \square

We apply this algorithm on the matches of S_{MC}^h and get S_{MC}^{h*} that satisfies the transitivity property, that is

$$S_{MC}^{h*} = S_+^{h*} \cup S_-^{h*} = \text{TransitivityChecking}(S_+^h) \cup S_-^h.$$

3.3 Pseudo-label Selection

Simply feeding pseudo-labeled data generated by the blocker (S_{BK}) and the matcher (S_{MC}) to each other might mislead the models due to the noises. Instead, We introduce additional steps to select their training data from generated pseudo-labeled data based on the role and functionality of the blocker and the matcher.

Recall that the blocker retrieves potential matches at a coarse level while the matcher distinguishes the true matches more precisely in the ER system. Hence, the matcher tends to be more reliable than the blocker. In light of this, we let the blocker trust the high-confidence pseudo-labeled data S_{MC}^{h*} generated by the matcher. Since we adopt contrastive learning methods to train the blocker, where only one positive pair is required for each entry (see Section 3.4), we select the highest confidence positive pair in S_{MC}^{h*} when there are multiple positive pairs for one entry. We take entry $e \in D$ as an example and $e' \in D'$ is similar. Let

$$e'^* = \arg \max_{e'} \{\text{sim}(e, e') \mid (e, e', 1) \in S_{MC}^{h*}\}. \quad (6)$$

We only preserve $(e, e', 1)$ when there are multiple matches for e . We denote the selected pseudo-labeled data as S_{PSD4BK} .

As for the opposite direction, the matcher utilizes its whole pseudo-labeled data S_{MC} to check the blocker's pseudo-labeled data S_{BK} . That is, if the label of (e, e') in S_{BK} is consistent with the label of (e, e') in S_{MC} , the matcher accepts this pseudo-labeled data. Consequently, the matcher will increase the confidence (predicted score) of this pair after training. We define

$$S_{PSD4MC}^{BK} = \{(e, e', y) \mid y \in \{0, 1\}, (e, e', y) \in S_{BK}, (e, e', y) \in S_{MC}\}. \quad (7)$$

Note that the transitivity property is satisfied for S_{PSD4MC}^{BK} as each entry still appears at most once in the matches.

Beyond that, the matcher also adopts part of the self-generated high-confidence pseudo-labeled data to avoid forgetting what it has already learned. Specifically, the matcher keeps the high-confidence pairs that do not exist in the blocker's pseudo-labeled data,

$$S_{PSD4MC}^{\text{self}} = \{(e, e', y) \mid y \in \{0, 1\}, (e, e', y) \in S_{MC}^{h*}, (e, e', y) \notin S_{BK}, (e, e', 1 - y) \notin S_{BK}\}. \quad (8)$$

Formally, the set of pseudo-labeled data for training the matcher is $S_{PSD4MC}^{\text{self}} = S_{PSD4MC}^{BK} \cup S_{PSD4MC}^{\text{self}}$. Overall, the matcher learns from itself and the partial knowledge provided by the blocker.

3.4 Training

Given the annotated and pseudo-labeled data, we then introduce how our blocker and matcher are updated by learning from these two sources of labeled data³ in the following.

3.4.1 Training of Blocker. As discussed in Section 2.3, the optimization goal of a DL-based blocker is to learn a good mapping function $f_{BK}(\cdot)$ to convert entries into semantic-preserving vectors. Recall that a data entry is represented as a set of attribute-value pair $e = (\text{attr}_i, \text{val}_i)_{1 \leq i \leq k}$. After serializing the entry into a series of tokens, learning entry embedding is equivalent to learning sentence embedding. In light of this, we leverage contrastive learning to train the blocker since recent studies have demonstrated the advantage of contrastive learning of sentence embeddings [11].

One critical issue is how to construct positive pairs to train the blocker via contrastive learning. The common approach for positive pair construction is to apply a data augmentation module to transform the original text into two correlated views. The key point for

data augmentation is to ensure that the augmented views share similar semantics. Considering input data characteristics in ER tasks, we adopt the following three entry transformation operations.

- (1) *Token shuffle.* We randomly choose one attribute attr_i and shuffle the tokens of the corresponding value val_i . The rationale is that the semantics in attribute values does not always depend strictly on the order of the tokens.
- (2) *Column shuffle.* We shuffle the attribute-value pairs: $e_{\text{aug}} = (\text{attr}_{r_i}, \text{val}_{r_i})$, where r is a random permutation. It is evident that the permuted sequence has the same semantics as the original one.
- (3) *Token deletion.* In general, the deletion of non-keywords would not change the semantics. We randomly delete the tokens in $\{\text{val}_i\}$ while ensuring that the deletion rate is less than 20% to reduce the likelihood of keyword deletion.

In addition to the augmented views, we utilize the matches in the labeled data to construct positive pairs. Specifically, if an entry e can find its matched entry e_{match} in the labeled data, we take (e, e_{match}) as a positive pair. Otherwise, we use augmentation methods to transform the data entry e into a correlated view e_{aug} and regard (e, e_{aug}) as a positive pair. The matches in the labeled data (e, e_{match}) can be considered as harder positive samples compared to the augmented ones (e, e_{aug}) , which facilitates entry representation learning. For easy illustration, we use $\text{pos}(e)$ to denote the corresponding paired entry of e (constructed from labeled data or augmentation).

It is worth noting that $\text{pos}(e)$ comes from three sources, each with different reliability. Thereby, we use a weighting factor w to differentiate their contribution to the gradient updates,

$$w = \begin{cases} w_{\text{annot}}, & (e, \text{pos}(e), 1) \in S_{\text{annot}} \\ w_{\text{aug}}, & \text{pos}(e) \text{ from augmentation} \\ w_{\text{psd}}, & (e, \text{pos}(e), 1) \in S_{PSD4BK} \end{cases} \quad (9)$$

we set $w_{\text{annot}} = 2.0$, $w_{\text{aug}} = w_{\text{psd}} = 1.0$ in the experiments.

Afterward, the blocker converts the two entries of the positive pair $(e, \text{pos}(e))$ into entry embeddings, *i.e.*,

$$\mathbf{h} = f_{BK}(e), \quad \tilde{\mathbf{h}} = f_{BK}(\text{pos}(e)). \quad (10)$$

In this work, we use the same network architecture as SBERT [30] to realize the function $f_{BK}(\cdot)$ and initialize it with pre-trained weights.

Then, a small network projection head is leveraged to map the representations into the space where contrastive loss is applied [6],

$$\mathbf{z} = \text{Proj}(\mathbf{h}), \quad \tilde{\mathbf{z}} = \text{Proj}(\tilde{\mathbf{h}}). \quad (11)$$

Here, we adopt one-layer MLP to realize the function $\text{Proj}(\cdot)$, following [11]. After the training is completed, the projection head is thrown away.

During the training stage, we sample batches $E \subseteq D \cup D'$ and optimize the parameters of the blocker with respect to the following contrastive loss:

$$L_{BK} = -\sum_{e_i \in E} w_i \log \frac{\exp(\cos(\mathbf{z}_i, \tilde{\mathbf{z}}_i)/\tau)}{\sum_{e_j \in E} \exp(\cos(\mathbf{z}_i, \mathbf{z}_j)/\tau)}, \quad (12)$$

where \mathbf{z}_i is the projected representation of e_i , w_i is the weighting factor differentiating the contribution of different sources of positive pairs, the entries in the same batch are regarded as negative examples, and τ is the temperature. We set τ as 0.05 in practice. In

³For simplicity, we refer to both the annotated data and the pseudo-labeled data as labeled data without special illustrations.

this way, contrastive learning enables us to pull semantically close samples together and push those non-similar ones apart.

3.4.2 Training of Matcher. As discussed in Section 2.4, the matcher is a pairwise binary classifier by fine-tuning LMs. To update the parameters of the matcher, we combine both the annotated data S_{annot} and pseudo-labeled data $SPSD4MC$. In general, the annotated data is of higher quality than the pseudo-labeled data since the latter contains noisy labels. Therefore, it would be beneficial to distinguish the importance of these two sources of labeled data, which is a critical research problem in the field of weakly-supervised learning [45]. Here, we adopt a simple re-weighting strategy that controls the contribution of the pseudo-labeled data to the gradients compared to the annotated data. Overall, the parameters of the matcher are learned via the following cross-entropy loss:

$$L_{MC} = -\sum_{(e,e',y) \in S_{annot}} (y \log(\hat{y}) + (1-y) \log(1-\hat{y})) - w_{MC} \sum_{(e,e',y) \in SPSED4MC} (y \log(\hat{y}) + (1-y) \log(1-\hat{y})), \quad (13)$$

where $\hat{y} = g_{MC}^m(e, e')$ is the probability of (e, e') being a match estimated by the matcher, $w_{MC} \in [0, 1]$ is a hyper-parameter controlling the influence of the pseudo-labeled data. When $w_{MC} = 1$, two sources of labeled data are treated equally; when $w_{MC} = 0$, the pseudo-labeled data is ignored. In this work, we set w_{MC} to be:

$$w_{MC} = \min\left(\frac{|S_{annot}|}{|SPSED4MC|}, 1\right). \quad (14)$$

In this way, we balance the overall contribution of all the annotated data and the overall contribution of all the pseudo-labeled data. Note that we treat all the pseudo-labeled data equally. Differentiating the contribution of each pseudo-labeled data based on its uncertainty might further boost the performance, which is beyond the scope of this work.

4 INFERENCE

In this section, we discuss how to conduct inference after the training is completed.

One typical solution is a two-step process, *i.e.*, using the blocker to retrieve the potential matches at first, then using the matcher to predict whether the pair of entries is a match. For instance, the blocker can either retrieve pairs where the similarity score exceeds a threshold (e.g., 0.7) or keep K nearest neighbors for each entry (as outlined in Algorithm 1) [31]. However, these blocking strategies require a pre-defined threshold or K , of which the optimal value is always different for different datasets (as observed in Figure 2).

To this end, we propose a dynamic nearest neighboring blocking strategy to retrieve candidates with the assistance of the matcher, as summarized in Algorithm 5. At each step, the blocker retrieves k most similar entries for each entry $e \in D$, and sorts them by similarity score from highest to lowest. Then, the matcher makes a prediction on k pairs. If at least one of the recent k pairs is predicted as a match or the lowest similarity score is larger than a threshold t_{BK} , letting the blocker retrieve the next k most similar entries; otherwise, the process finishes. Here, t_{BK} is determined by the statistic of validation set, *i.e.*,

$$t_{BK} = \text{Mean}(sim_{pos}) - \alpha \text{Std}(sim_{pos}), \quad (15)$$

$$sim_{pos} = \{\text{sim}(e, e') \mid (e, e', 1) \in S_{valid}\},$$

Algorithm 5: Inference($M_{BK}, M_{MC}, D, D', S_{valid}, k$)

Input: a blocker M_{BK} ; a matcher M_{MC} ; two collections of data entries, D and D' ; validation set S_{valid} ; k , number of retrieved entries at each step

Output: A set of predicted matches $S_{predict}$

```

1 Initialize:  $S_{predict} \leftarrow \phi$ 
2  $\{h_i\} \leftarrow \{f_{BK}(e_i) \mid e_i \in D\}$ 
3  $\{h'_i\} \leftarrow \{f_{BK}(e'_i) \mid e'_i \in D'\}$ 
4  $sim_{pos} \leftarrow \{\text{sim}(e, e') \mid (e, e', 1) \in S_{valid}\}$ 
5  $t_{BK} \leftarrow \text{Mean}(sim_{pos}) - \text{Std}(sim_{pos})$ 
6 for each entry  $e \in D$  do
7    $K \leftarrow 0; N_{last} \leftarrow \phi; S_e \leftarrow \phi$ 
8   while True do
9      $K \leftarrow K + k$ 
10     $N_e \leftarrow \text{K-Most-Similar}_{e' \in D'}(h_e, \{h'_i\}, D', K)$ 
11     $C_{cur} \leftarrow \{(e, e') \mid e' \in N_e \setminus N_{last}\}$ 
12     $S_{cur} \leftarrow \text{Predict}(M_{MC}, C_{cur})$ 
13     $S_e \leftarrow S_e \cup S_{cur}$ 
14    if (matches in  $S_e$  and no matches in  $S_{cur}$ ) or (no matches in  $S_e$  and  $\min_{e'} \{\text{sim}(e, e') \mid e' \in N_e\} < t_{BK}$ ) then
15       $\text{break}$ 
16    else
17       $N_{last} \leftarrow N_e$ 
18   $S_{predict} \leftarrow S_{predict} \cup S_e$ 
19 return  $S_{predict}$ 

```

where $\text{Mean}(\cdot)$ denotes the average operation, $\text{Std}(\cdot)$ denotes the standard deviation operation, and α is a hyperparameter. We select a value of α equal to 1.65, which corresponds to a z-score of 95% and has been shown to be effective for all datasets. Note that our method can be considered as an extension of a combination of threshold-based and KNN-based similarity pairing.

5 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate our CLER framework. Section 5.1 presents the experimental setup. Section 5.2 investigates whether the blocker and the matcher in our framework can mutually benefit. Section 5.3 compares the overall performance of our framework with baselines under different annotation budgets. Section 5.4 analyzes the impact of several designs.

5.1 Experimental Setup

Datasets. In this study, we conduct experiments on seven widely adopted public datasets from various domains for ER tasks. These datasets are obtained from the Magellan data repository [16] and the Alaska benchmark [7]. A summary of the dataset statistics can be found in Table 1. The M dataset has 24 tables, and we select two of them for experiments⁴. It is worth noting that all experimental datasets possess complete ground truth, enabling the annotation process to be simulated. For each dataset, 25% of the entries in D are randomly chosen as test entries, denoted as D_{test} , such that none

⁴This is considered as a similarity-join variant in the original work [7].

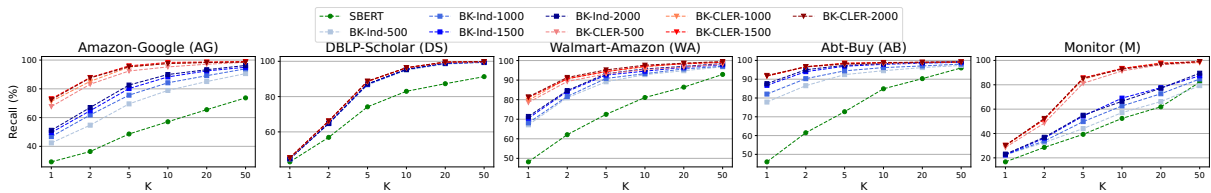


Figure 2: C-R curve comparison of blockers on ER datasets. The K on the X-axis is the number of most similar entries that the blocker returns for each input $e \in D$, which is proportional to CSSR. BK-CLER- B represents the performance of the blocker in our framework with an annotation budget being B , BK-Ind- B represents the performance of the blocker trained independently with an annotation budget being B using randomly selected annotated data, and SBERT [30] represents the performance of using the pre-trained SBERT as the blocker. We omit the results on the DA and FZ datasets as the blocker already achieves nearly perfect recall with $K = 1$.

Table 1: Statistics of benchmark datasets.

Dataset	# entries D, D'	# matches	(%) matches
Amazon-Google (AG)	1363, 3226	1300	0.0296
DBLP-ACM (DA)	2616, 2294	2224	0.0371
DBLP-Scholar (DS)	2616, 64263	5347	0.0032
Fodors-Zagats (FZ)	533, 331	112	0.0635
Walmart-Amazon (WA)	2554, 22074	1154	0.0020
Abt-Buy (AB)	1081, 1092	1098	0.0930
Monitor (M)	603, 4323	343	0.0132

of the pairs containing test entries are annotated during training. Additionally, 500 pairs are randomly selected and annotated as validation data for all compared methods, and do not count toward the total annotation budget.

Compared methods. We compare CLER with representative ER approaches, including unsupervised, supervised, and active learning methods. The following summarizes the compared methods.

- **CollaborEM** [12] is a recent unsupervised framework, which constructs rule-based pseudo labels for training and combines graph features from entity graphs and sentence features from LMs as entry embeddings. For comparability, we allow it to additionally use the same amount of annotated data for training.
- **DITTO** [18] is a representative supervised ER model, which casts ER as a sentence-pair classification problem and fine-tunes a pre-trained language model (as the matcher) by labeled data.
- **DTAL** [15] is an iterative active-learning-based entity matching method, which allocates half of the budget (*i.e.*, $b/2$) to the most likely false positives and false negatives (*i.e.*, predicted probability closest to 0.5), respectively. In addition, it adds $b/2$ pseudo labels to the highest confidence positives and negatives, respectively.

Settings. We adopt a low-resource setting for the ER task as defined in Section 2.1. We vary the total budget B within the range of [500, 1000, 1500, 2000], consistent with the sub-experiment on label efficiency in DITTO [18]. During the training process, for each entry $e \in D \setminus D_{test}$, the blocker retrieves $K = 10$ entries in D' whose corresponding embedding vectors have the highest $K = 10$ cosine similarity. For all the methods, we set the max sequence length to 256 and the batch size to 64, and perform a grid search on the learning rate in [1e-5, 2e-5, 3e-5]. For non-iterative methods, the number of epochs is 40. For iterative methods (*i.e.*, DTAL and

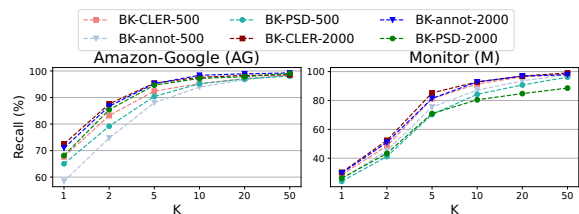


Figure 3: C-R curve comparison of blockers on two representative datasets. BK-CLER- B , BK-annot- B , and BK-PSD- B represent the performance of the blocker in our framework with an annotation budget being B , the blocker trained with annotated data, and the blocker trained with pseudo-labeled data, respectively.

CLER), the number of iterations ($\#iter$) is fixed to 10 so that the budget for each iteration is $b = B/10$, and the number of epochs is $5 \times (\#iter - 1) + 40$. For each baseline, all other hyperparameters are based on the original settings in their respective papers. All methods are implemented on a server with an A100 GPU using PyTorch [26] and the Transformers library [36]. We perform five runs with different seeds and report the average values.

5.2 Can Blocker and Matcher Mutually Benefit?

5.2.1 The Gain for Blocker. To evaluate the impact of the matcher on the training of the blocker, we compare the trained blocker in our CLER framework (BK-CLER) with a variant (BK-Ind) that is trained *independently* using randomly selected annotated data. In addition, we include the default blocker SBERT as a reference.

The CSSR-Recall (C-R) curves for the evaluated blockers at different annotation budgets over five datasets are presented in Figure 2. We use K as the X-axis for the plots, where the CSSR is proportional to K in terms of $CSSR = K/|D'|$. The results show that BK-CLER consistently outperforms BK-Ind and SBERT, with the gap being more significant when there is less annotation budget (AG, AB, M). This indicates the essential role of the matcher in training the blocker within our framework.

Beyond that, we decompose the impact of the matcher on the blocker’s training into two parts, *i.e.*, the annotated data selected by the matcher, and the pseudo-labeled data generated by the matcher.

Table 2: F1-score (%) of the compared matchers with annotation budget $B = 2000$ on the processed Magellan datasets. We bold the best result and underline the second-best result for each dataset. For reference, we also include the performance of DITTO-full (i.e., DITTO trained with the full training data of the processed Magellan datasets), and LLaMA-65B [32] and GPT3-175B [4] with 10-shot in-context learning [23].

Dataset	AG	DA	DS	FZ	WA	AB	Avg
CLER	77.65	98.90	93.65	97.54	88.48	95.84	92.01
CLER-non-cross	<u>70.61</u>	<u>98.84</u>	91.86	92.68	<u>86.93</u>	<u>93.56</u>	<u>89.08</u>
CollaborEM	61.84	98.08	72.72	<u>94.92</u>	73.40	90.38	81.89
DITTO	54.63	97.37	90.82	93.55	69.13	83.99	81.58
DTAL	67.39	98.46	<u>92.59</u>	87.66	84.52	90.28	86.82
DITTO-full	74.18	99.04	94.35	94.34	86.06	92.51	90.08
LLaMA-65B	56.50	93.29	74.21	96.30	71.20	72.87	77.39
GPT3-175B	58.97	94.16	83.80	94.34	81.36	66.53	79.86

Figure 3 shows the results of the blocker BK-annot- B that is trained with the annotated data, and BK-PSD- B that is trained with the pseudo-labeled data, respectively. The significant performance gain of BK-CLER-500 over BK-annot-500 and BK-PSD-500 suggests that both the annotated data and pseudo-labeled data contribute to the performance improvement of the blocker. One impressive result is the outperformance of BK-annot- B over BK-Ind- B (refer to Figure 2), which indicates the annotated data selected by the matcher is more informative than random selection, thus facilitating the training of the blocker.

5.2.2 The Gain for Matcher. To explore whether the matcher benefits from our Co-learning framework, we evaluate the matchers by the task of classifying the same set of entry pairs. Here, the test data is a subset of the processed Magellan dataset [16]⁵, i.e., $S_{test} = \{(e, e', y) | e \in D_{test} \wedge (e, e', y) \in S_{Mag}\}$, where S_{Mag} represents the processed Magellan dataset.

First, we compare the matcher trained in CLER with a non-cross-trained matcher trained in a variant of CLER, denoted as CLER-non-cross, where the blocker is removed from the iteration stage and the pseudo labels are only made by the matcher. As shown in Table 2, CLER⁶ consistently outperforms CLER-non-cross, suggesting that the matcher does benefit from the blocker. The advantage is more prominent on the AG dataset, where the improvement of the trained blocker is more significant, and thus the blocker leads to better candidate sets and pseudo-labeled data.

Moreover, we also compare our matcher with baseline matchers. Table 2 shows that our matcher exhibits superior performance compared to other methods across all the Magellan datasets. Notably, in datasets AG, FZ, WA, and AB, our matcher trained with an annotation budget of $B = 2000$ even outperforms DITTO-full, which is trained with the full data of the processed Magellan dataset. The reason could be that DITTO-full’s capacity is limited by the finite scope of processed data it can access (e.g., the size of the processed AG dataset is 9167). In contrast, our training framework can potentially assimilate a more expansive range of information, given that our

⁵The processed Magellan dataset contains the candidate pairs after the blocking step.

⁶In this subsection, we use CLER to represent the matcher in CLER, the same for the other methods.

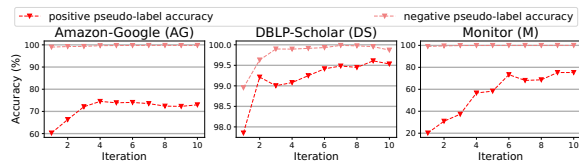


Figure 4: The accuracy of positive and negative pseudo labels in the pseudo-labeled data used by the matcher ($B = 2000$). The accuracy of positive labels is significantly improved during iterations, while the accuracy of negative labels is already high in the beginning.

blocker has access to the original data sets (e.g., on the AG dataset at $B = 2000$ the size of the labeled data is 12529, including 2000 annotated data and 10529 pseudo-labeled data). Among the baselines, DTAL achieves the best performance in most cases, indicating that selecting informative data for annotation is crucial when the budget is limited. It is also worth noting that CollaborEM’s performance is not satisfactory on the DS dataset, where all the other methods achieve over 90% F1-score. The reason for this could be that the noise in the generated pseudo-labeled data exceeds the amount of information it can bring. However, this problem does not occur in CLER, which might be mainly attributed to the improved quality of the pseudo labels as training proceeds (as shown in Figure 4) and the re-weighting mechanism.

Furthermore, we include the performance of LLaMA-65B and GPT3-175B with 10-shot learning as reference [23]⁷. Though LLaMA-65B and GPT3-175B achieve impressive performance with only ten randomly selected examples, they are still significantly inferior to the performance of CLER. The gap is small on relatively easy datasets DA and FZ, and large on harder datasets AG and AB. Moreover, they require more than 500x inference time, e.g., LLaMA-65B takes 52.9 minutes while our CLER finishes the inference within 0.1 minutes on the DA dataset.

5.3 Overall Performance with Low Resource

We assess the overall performance of the compared methods by employing the inference strategy proposed in Section 4. Since the compared baselines are mainly proposed for the matching step, we let baselines be combined with three kinds of blockers, i.e., DeepBlocker [31], Sudowoodo [35], and the blocker trained with the same setting in CLER (depicted in Section 3.4.1), and choose the best F1 scores among the three combinations for each baseline.

Figure 5 presents the results of all compared methods on different datasets across different budgets. Overall, our CLER framework consistently outperforms the baselines on all experimental datasets, indicating the effectiveness and robustness of our framework toward various types of datasets. On average, our method outperforms the baselines by 9.13-51.55% when $B = 500$, 10.06-27.46% when $B = 1000$, 12.33-23.71% when $B = 1500$, and 12.09-17.56% when $B = 2000$. Notably, all baselines perform poorly on the M dataset,

⁷All the other compared methods follow “pre-train, fine-tune” paradigm, and LLaMA-65B and GPT3-175B with 10-shot learning follow “in-context-learning” paradigm. Hence, we include their performance as reference.

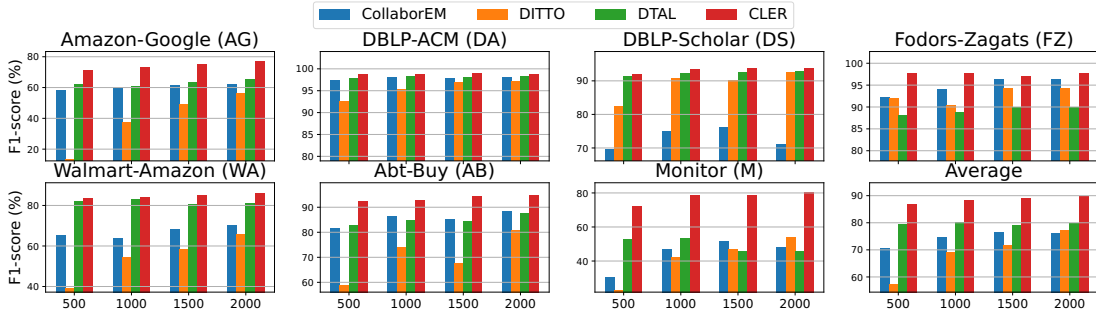


Figure 5: The overall performance (in terms of F1-score) of compared methods on seven EM datasets and the average over all datasets under different budgets $B = [500, 1000, 1500, 2000]$. All the settings are repeated five runs with different seeds. CLER shows consistent improvement over baselines in all datasets. On average, CLER achieves about 10% absolute improvement compared to the second-best method under four budget settings.

Table 3: F1-score (%) of pseudo-label generation strategies on the overall performance. “BK-Neg” represents the variant that the blocker takes all the unsure pairs as negative; “MC w/o conf” represents the variant without pseudo-label confidence differentiation; “MC w/o check” represents the variant by removing the transitivity checking of the matcher.

Budget Dataset	$B = 500$			$B = 2000$		
	AG	WA	M	AG	WA	M
CLER	71.25	83.62	71.84	76.99	86.20	80.24
BK-Neg	71.14	83.12	70.79	75.89	86.87	78.44
MC w/o conf	71.55	81.40	64.88	75.05	84.67	77.90
MC w/o check	71.62	82.92	70.03	75.54	84.94	79.05

whereas our CLER achieves 71.84% F1-score with only an annotation budget of 500. Additionally, our framework demonstrates promising results on the AG and WA datasets, which are known to be relatively challenging among the Magellan datasets, suggesting the potential superiority of CLER on challenging datasets.

5.4 Further Analysis

In this section, we analyze several design choices in our framework by evaluating the variants in terms of overall performance.

5.4.1 *The effectiveness of pseudo-label generation strategies.* To assess the effectiveness of pseudo-label generation strategies, we compare CLER with its three variants: (1) “BK-Neg”: having the blocker treat all pairs except the most similar as negatives (equivalent to $t_{diff} = 0$) (2) “MC w/o conf”: without pseudo-label confidence differentiation (equivalent to $t_+ = t_- = 0.5$) and (3) “MC w/o check”: disabling the transitivity-checking to the matcher’s pseudo-labeled data S_{MC}^h . As shown in Table 3, the strategy of pseudo-label confidence differentiation greatly benefits the overall performance, while the advantages of the other two strategies are not significant except on the M dataset, possibly because subsequent selection and re-weighting designs dilute the impact of the pseudo label’s quality. Furthermore, we examine the average percentage of positive samples eliminated through transitivity checking. The observed percentages are 7.91% for the M dataset and 2.34% for the AG dataset,

Table 4: F1-score (%) of variants using different selection strategies. “BK w/o selection” means not using the selection strategy of BK ($S_{PSD4BK} = S_{MC}^{h*}$); “MC w/o BK” means MC does not learn from BK ($S_{PSD4MC} = S_{PSD4MC}^{self}$); “MC w/o self” means MC does not learn from itself ($S_{PSD4MC} = S_{PSD4MC}^{BK}$).

Budget Dataset	$B = 500$			$B = 2000$		
	AG	WA	M	AG	WA	M
CLER	71.25	83.62	71.84	76.99	86.20	80.24
BK w/o selection	71.72	82.57	65.08	75.30	85.59	75.51
MC w/o BK	66.19	77.09	67.02	72.15	84.84	76.77
MC w/o self	72.30	82.77	66.14	76.99	84.97	78.05

which may explain the significant degradation observed in the “MC w/o check” on the M dataset. Moreover, we figure out that these design elements appear to be complementary; for instance, in the case of “MC w/o conf”, the removal percentage surges to 9.31% (considerably greater than the 2.34%) on the AG dataset.

5.4.2 *The impact of pseudo-label selection strategies.* We evaluate the selection strategies by comparing CLER with its three variants: (1) “BK w/o selection”: without the selection strategy of BK ($S_{PSD4BK} = S_{MC}^{h*}$). (2) “MC w/o BK”: MC does not learn from BK ($S_{PSD4MC} = S_{PSD4MC}^{self}$); (3) “MC w/o self”: MC does not learn from itself ($S_{PSD4MC} = S_{PSD4MC}^{BK}$). As presented in Table 4, “BK w/o selection” shows significant degradation on the M dataset, where the quality of the matcher’s pseudo labels is relatively low compared to other datasets. This suggests the effectiveness of the blocker’s selection strategy when dealing with low-quality pseudo-labeled data. Additionally, the advantage of CLER over “MC w/o BK” indicates that the pseudo-labeled data generated by the blocker greatly benefits the matcher’s training, supporting our motivation of employing Co-learning between the blocker and the matcher. Furthermore, the benefit of CLER over “MC w/o self” demonstrates the advantage of allowing the matcher to learn what it already knows.

5.4.3 *The impact of re-weighting mechanism.* We assess the effectiveness of the re-weighting mechanism by comparing our framework with two variants: (1) “MC w/o Re-weighting”: the matcher

Table 5: F1-score (%) of variants using different re-weighting strategies (without re-weighting of BK and without re-weighting of MC).

Budget Dataset	$B = 500$			$B = 2000$		
	AG	WA	M	AG	WA	M
CLER	71.25	83.62	71.84	76.99	86.20	80.24
BK w/o Re-weighting	71.06	83.33	66.76	75.79	85.45	78.66
MC w/o Re-weighting	66.54	78.08	56.45	74.01	82.12	70.66

Table 6: Recall (%) and the average number of retrieved entries of the blocking results when total budget $B = 2000$.

Dataset	AG		WA		M	
	Recall	AvgK	Recall	AvgK	Recall	AvgK
CLER	98.12	14.32	98.45	34.90	96.43	11.28
Fixed K	97.93	15.00	99.07	35.00	94.52	12.00
	98.12	23.00	98.21	19.00	96.43	26.00
Fixed Threshold	98.43	15.00	98.36	35.00	95.48	12.00
	98.12	10.80	98.21	28.00	96.43	25.30

Table 7: The total (and breakdown) inference time of ER with and without the blocker on different datasets.

Dataset	w/o BK	with BK	BK Emb.	BK Sim.	MC Pred.
	Total Time(s)	Total Time(s)	Time(s)	Time(s)	Time(s)
AG	1451.70	5.07	2.77	0.06	2.25
WA	7107.83	10.35	8.43	0.31	1.61
M	302.61	2.38	1.97	0.06	0.35

treats pseudo-labeled and annotated data equally. (2) “BK w/o Re-weighting”: the blocker treats all three types of positive pairs equally. The results presented in Table 5 demonstrate that the re-weighting strategy significantly improves the matcher’s training, while it only brings a noteworthy improvement to the blocker’s training on the M dataset. These outcomes suggest that (1) the matcher, being a more precise classifier, is more sensitive to noisy data, and (2) the blocker can always learn from the pseudo labels generated by the matcher unless the quality is particularly low.

5.4.4 The impact of inference strategies. We evaluate our inference strategy by comparing it to KNN-Blocking with a fixed K , and a threshold-based blocking with a fixed threshold. In this part, we compare the recall rate under the same K value, and the K value under the same (similar) recall rate. As shown in Table 6, different datasets require different K or thresholds (this phenomenon can also be observed in Figure 2), which are difficult to determine in advance without prior knowledge. However, our method achieves an acceptable recall rate and greatly reduces the candidate size simultaneously, without requiring specific hyperparameters for the dataset. Note that the advantage of our method is significant on the M dataset, where the number of matched entries for each test entry varies considerably. And our method achieves comparable performance on the other two datasets. These results imply that our approach is applicable to datasets with varying characteristics.

5.4.5 Running time at inference stage. We analyze the running time of three operations at the inference stage: mapping entries

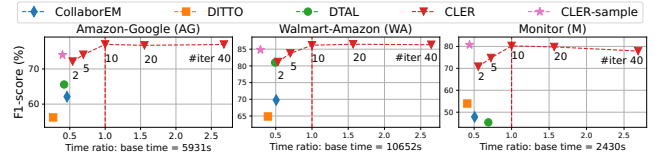


Figure 6: The training time and F1-score (%) of compared methods when $B = 2000$. The time is shown in the ratio w.r.t CLER #iter = 10. “CLER-sample” represents the variant of CLER that employs the sampling strategies in the training of the blocker and the matcher.

into embeddings by the blocker (denoted as “BK Emb.”), calculating the pairwise similarity score to find the nearest neighbors (denoted as “BK Sim.”), and predicting the probability by the matcher (denoted as “MC Pred.”). As shown in Table 7, the running time is primarily dominated by the blocker’s mapping operation and the matcher’s prediction operation, which emphasizes the necessity of the blocking step. Note that, in this work, we exhaustively calculate all the pairwise similarity scores via Pytorch. When the size of two collections is large, pairwise similarity calculation is not affordable, i.e., $O(|D| \times |D'|)$. There are some works on optimizing the nearest neighbor search [10, 20, 44], which is out of the scope of this work.

5.4.6 Running time at training stage. In this part, we first analyze the overall running time of compared methods. For our CLER, we vary the number of iterations (#iter) to study the trade-off between the training time and the test performance. As shown in Figure 6, investing more training time improves the accuracy while becomes very marginal when #iter > 10. Meanwhile, our CLER 2-iteration variant significantly outperforms the baselines at a similar or slightly larger time cost (except on the WA dataset, the variant is only slightly better than DTAL), which indicates the superiority of CLER even with same time cost. Furthermore, we investigate the breakdown of the running time for three operations of the blocker and matcher in our CLER at the iteration stage. As shown in Table 8, the training of the blocker and matcher takes a major part of the running time. Moreover, the training time varies significantly across datasets under the same annotation budget, which is mainly related to the size of the two data collections and the amount of pseudo-labeled data. These results give us some insights into optimizing time: (1) We can randomly sample $SPSD_{MC}$, and adjust the weighting factor w_{MC} for each training epoch, thus reducing the training time for the matcher. (2) We can randomly sample entries that cannot find their matched entries in the labeled data for each training epoch, thus reducing the training time for the blocker. We leverage these two sampling strategies over CLER with 10 iterations, denoted as “CLER-sample”, which greatly saves the training time but results in minor degradation in some cases (as shown in Figure 6).

5.4.7 Robustness analysis. To analyze the robustness of our CLER model to data with noisy attribute values, we compare its performance on the original structured datasets (denoted as Clean) and their dirty versions (denoted as Dirty)⁸, which are released by [22].

⁸<https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md>

Table 8: The breakdown of the running time of the six operations at the iteration stage when $B = 2000$.

operation	AG		WA		M	
	BK	MC	BK	MC	BK	MC
pseudo-label generation	21.33	233.25	31.76	185.30	14.82	71.48
pseudo-label selection	4.07	2.13	3.41	4.26	1.77	1.68
training	1499.38	3941.35	6002.04	3871.44	1337.98	858.92

Table 9: Robustness analysis of CLER (in terms of F1-score (%)) on the Clean and Dirty versions of structured datasets.

Budget Dataset	$B = 500$			$B = 2000$		
	DA	DS	WA	DA	DS	WA
Clean	98.72	91.88	83.62	98.73	93.74	86.20
Dirty	98.72	91.90	81.18	98.90	93.86	84.55

As shown in Table 9, CLER achieves comparable performance on the clean and dirty versions of the DA and DS datasets; while on the WA dataset, CLER faces a slight degradation. In general, our model is robust to data with noisy attribute values.

6 RELATED WORK

6.1 Entity Blocking

Blocking plays a crucial role in enhancing efficiency and scalability by reducing the number of entry pairs considered for matching [24, 25]. Two mainstreams of traditional methods are key-based blocking (*a.k.a.*, hash-based blocking) and rule-based blocking [16], both of which require considerable domain expert efforts [42].

With the widespread of deep learning, there emerge several attempts in DL-based blocking, where the key point is to learn entry embeddings [14, 31]. For instance, DeepER represents each entry with an average of word embeddings and adopts locality-sensitive hashing to retrieve most similar candidates [9]. DeepBlock [31] further explores a large space of DL solutions for blocking, and figures out that Autoencoder appears to be highly promising. Based on labeled data, AutoBlock [42] differentiates the importance of words by leveraging the attention mechanism, but labeled data is not always available in practice. Recently, some studies utilize pre-trained LMs to generate the entry embedding [12, 18]. However, there might exist a semantic gap between the pre-training corpus and the data of ER task. In light of this, one recent work [35] leverages a classical contrastive learning framework to fine-tune LMs for a better blocker. In addition to contrastive learning, our work aims to bridge this gap via an iterative Co-learning framework with the assistance of the matcher and the annotated data.

6.2 Entity Matching

Recent studies on entity matching (EM) mainly leverage deep learning (DL) methods [2, 41]. To the best of our knowledge, DeepER is the first attempt that employs LSTM to encode structural data into a dense vector [9]. Later on, DeepMatcher [22] explores the effectiveness of many DL modules and demonstrates its superiority for textual and dirty EM problems. Since pre-trained models have shown remarkable advances in various NLP tasks, there emerge several

studies fine-tuning Transformer-based pre-trained LMs [17, 18, 27]. Despite the effectiveness, a satisfying performance always require a large number of annotated samples [12]. To reduce the labeling cost, some efforts focus on learning a matcher with limited or even zero labeled data [5, 15, 21, 33, 43]. For instance, CollaborEM [12] creates pseudo labels based on the embeddings generated by SBERT and then trains the matcher, which heavily relies on the rules and the generalization of SBERT to the target dataset. With the rapid improvement of LMs’ capabilities [32], the advanced LMs have shown impressive in-context learning performance in EM [23], but still could not achieve comparable performance to the fine-tuned model. Besides, the inference of these models is quite time-consuming due to their extremely large model sizes. Our work follows the fine-tuning paradigm and utilizes the advantages of blocker and matcher to generate more reliable pseudo labels to broaden the supervised signals.

6.3 Co-training and Co-teaching

Co-training has been a particularly effective technique when the training data is only partially labeled [1, 3, 28]. The basic idea of Co-training is that two or more models can learn from different views of the datasets, and the pseudo labels predicted by the models can enlarge the training set of the others’. Recent studies extend this concept to include different models as different views [28, 37]. Motivated by Co-training, Co-teaching maintains two models with the same network architecture and optimization goals simultaneously, and allows the two models to teach each other by recommending relatively clean data for training, which has shown robustness even with extremely noisy labels [13]. Different from Co-training that requires two views of the datasets, Co-teaching needs a single view of the dataset. In this work, we leverage the Co-learning mechanism to let the blocker and the matcher learn from each other under low-resource settings. Our Co-learning is stemmed from Co-training and Co-teaching, where the blocker and the matcher extract different information from input data, and they teach each other via pseudo-labeled data with potential noises.

7 CONCLUSION

In this work, we present CLER for low-resource ER, which exploits the cooperative relationship between the blocker and the matcher, allowing them to learn from each other to broaden the limited supervised signal. Empirical results on seven benchmark datasets demonstrate that they can mutually benefit in our framework. And the performance gain is more prominent in the cases of a very low annotation budget. Furthermore, the superiority of the blocker and the matcher makes our CLER framework better than baselines in terms of overall performance. A promising future direction is to explore how to improve training efficiency for an iterative Co-learning framework. Additionally, we are interested in investigating whether the blocker and the matcher can still benefit from each other’s learning without any annotated data.

ACKNOWLEDGMENTS

The research work described in this paper was partially conducted in the JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

REFERENCES

- [1] Maria-Florina Balcan, Avrim Blum, and Ke Yang. 2004. Co-training and expansion: Towards bridging theory and practice. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, Vol. 17. 89–96.
- [2] Nils Barlaug and Jon Atle Gulla. 2021. Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 3 (2021), 1–37.
- [3] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. 92–100.
- [4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [5] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1335–1349.
- [6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.
- [7] Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, Federico Piai, and Divesh Srivastava. 2021. Alaska: A flexible benchmark for data integration tasks. *arXiv preprint arXiv:2101.11259* (2021).
- [8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.
- [9] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1454–1467.
- [10] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment* 12, 5 (2019), 461–474.
- [11] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 6894–6910.
- [12] Congcong Ge, Pengfei Wang, Lu Chen, Xiaozhe Liu, Baihua Zheng, and Yunjun Gao. 2021. CollaborEM: a self-supervised entity matching framework using multi-features collaboration. *IEEE Transactions on Knowledge and Data Engineering* (2021).
- [13] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *Advances in neural information processing systems* 31 (2018).
- [14] Delaram Javdani, Hossein Rahmani, Milad Allahgholi, and Fatemeh Karimkhani. 2019. Deepblock: A novel blocking approach for entity resolution using deep learning. In *2019 5th International Conference on Web Research (ICWR)*. IEEE, 41–44.
- [15] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource Deep Entity Resolution with Transfer and Active Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5851–5861.
- [16] Pradap Konda, Sanjib Das, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, et al. 2016. Magellan: toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1581–1584.
- [17] Bing Li, Yukai Miao, Yaoshu Wang, Yifang Sun, and Wei Wang. 2021. Improving the efficiency and effectiveness for BERT-based entity resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 13226–13233.
- [18] Yuliang Li, Jimfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment* 14, 1 (2020), 50–60.
- [19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
- [20] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.
- [21] Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In *Proceedings of the 2021 International Conference on Management of Data*. 1303–1316.
- [22] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*. 19–34.
- [23] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *arXiv preprint arXiv:2205.09911* (2022).
- [24] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–42.
- [25] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. 2016. Comparative analysis of approximate blocking techniques for entity resolution. *Proceedings of the VLDB Endowment* 9, 9 (2016), 684–695.
- [26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
- [27] Ralph Peeters and Christian Bizer. 2021. Dual-objective fine-tuning of BERT for entity matching. *Proceedings of the VLDB Endowment* 14 (2021), 1913–1921.
- [28] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. 2018. Deep co-training for semi-supervised image recognition. In *Proceedings of the european conference on computer vision (eccv)*. 135–152.
- [29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.
- [30] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.
- [31] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2459–2472.
- [32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
- [33] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Chengliang Chai, Guoliang Li, Ruixue Fan, and Xiaoyong Du. 2022. Domain adaptation for deep entity resolution. In *Proceedings of the 2022 International Conference on Management of Data*. 443–457.
- [34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [35] Runhui Wang, Yuliang Li, and Jin Wang. 2023. Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1502–1515.
- [36] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).
- [37] Qiyu Wu, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, and Daxin Jiang. 2022. PCL: Peer-Contrastive Learning with Diverse Augmentations for Unsupervised Sentence Embeddings. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 12052–12066.
- [38] Renzhi Wu, Alexander Bendeck, Xu Chu, and Yeye He. 2023. Ground Truth Inference for Weakly Supervised Entity Matching. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–28.
- [39] Dezhong Yao, Yuhong Gu, Gao Cong, Hai Jin, and Xinqiao Lv. 2022. Entity Resolution with Hierarchical Graph Attention Networks. In *Proceedings of the 2022 International Conference on Management of Data*. 429–442.
- [40] Xueying Zhan, Qingzhong Wang, Kuan-hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B Chan. 2022. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450* (2022).
- [41] Dongxiang Zhang, Yuyang Nie, Sai Wu, Yanyan Shen, and Kian-Lee Tan. 2020. Multi-context attention for entity matching. In *Proceedings of The Web Conference 2020*. 2634–2640.
- [42] Wei Zhang, Hao Wei, Bunyamin Sisman, Xin Luna Dong, Christos Faloutsos, and David Page. 2020. AutoBlock: A Hands-off Blocking Framework for Entity Matching. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 744–752.
- [43] Chen Zhao and Yeye He. 2019. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference*. 2413–2424.
- [44] Xi Zhao, Yao Tian, Kai Huang, Bolong Zheng, and Xiaofang Zhou. 2023. Towards Efficient Index Construction and Approximate Nearest Neighbor Search in High-Dimensional Spaces. *Proceedings of the VLDB Endowment* 16, 8 (2023), 1979–1991.
- [45] Zhi-Hua Zhou. 2018. A brief introduction to weakly supervised learning. *National science review* 5, 1 (2018), 44–53.