

Entity Resolution with Deep Interactions and Fine-Grained Difference Extraction based on BERT

Huiting Yuan, Liang Zhu ^(✉), Yu Wang ^(✉) and Zhouyan Liu

Hebei University, Baoding, Hebei 071002, China
zhu@hbu.edu.cn; wy@hbu.edu.cn

Abstract. Entity Resolution (ER) is crucial for data integration, identify if record pairs from one or multiple datasets refer to the same real-world entity. Traditional ER struggles with complex and diversity record structures, using coarse features that overlook subtle semantics, hindering performance. Furthermore, processing each record pair individually also increases computational costs. To overcome these issues, we propose DIBER, a novel ER model based on Siamese networks structure and a pre-trained language model (PLM) that generates contextually rich representations of records. It uses co-attention for inter-record link analysis and combines fusion with weighted attention to highlight subtle differences. A feature extractor further refines matching details, enhancing discrimination. It is adaptable for blocking and outperforms state-of-the-art (SOTA) methods on small datasets without injecting specific domain knowledge, as shown by extensive experiments.

Keywords: entity resolution, deep interaction, fine-grained, blocking.

1 Introduction

Entity Resolution (ER), also known as entity matching, record linkage, duplicate record detection, or reference reconciliation, aims to determine whether records refer to the same real-world entity. It plays a crucial role in data-centric processing, such as data cleaning, data integration, and data mining. After decades of development, ER has been widely used in various fields such as e-commerce [1], healthcare [2], and population censuses [3]. Due to its profound impact on these areas, it has garnered significant attention. In the current research, ER is treated as a binary classification problem [4]: if two records refer to the same entity, the pair of records is predicted to be a match; conversely, if the records refer to different entities, they are predicted to be unmatched. For instance, in Fig. 1, which consists of two tables that separately describe products from Amazon and Google, the record pairs <431, 1687> and <113, 3109> are considered to refer to the same entity because they represent the same real-world product. Conversely, the record pairs <431, 3109> and <113, 1687> are indicative of distinct entities as they correspond to different real-world products.

The ER process typically consists of two steps: blocking and matching. Our work mainly focuses on the matching task. In the core task of matching, accurately determining whether two records refer to the same real-world entity relies heavily on an exhaustive comparison of their key informational elements, such as brand names, product types, and specifications. If these elements align perfectly, it can be confidently concluded that the two records are indeed a match.

The concept of ER is formally proposed in [5] and much research has been conducted on matching, resulting in diverse solutions being proposed that encompass rule-based [6,7,8], crowd-sourcing [9,10] and machine learning [11,12,13] methods. In recent years, the development of deep learning (DL) technology has facilitated significant breakthroughs in ER. Compared with traditional ER methods, DL with its strong feature learning capability, can deeply explore the semantic similarity between records while significantly reducing the need for human involvement. In particular, the performance of ER is greatly enhanced by employing pre-trained language models (PLM) to obtain vector representations for records. Numerous studies have confirmed it [14,15].

Google			
ID	Title	Manufacturer	Price
113	microsoft windows server 2003 client additional license for devices5 pack	microsoft	209.0
431	microsoft licenses win svr 2003 ext conn lic (r3900292)	microsoft licenses	3371.85

Amazon			
ID	Title	Manufacturer	Price
1687	microsoft r39-00292 open win svr 2003 ext conn	NULL	1863.78
3109	microsoft windows server 2003 client additional license for devices5 pack(823930)	NULL	158.39

Fig. 1. Entity resolution on two datasets coming from Google and Amazon

The Ditto model [14], as a solution based on BERT [16], concatenates a pair of records into a single sequence and separates two records by the [SEP] token. It harnesses the special token [CLS] generated by BERT to represent the contextual embedding of the entire sequence pair, and feeds the vector into a simple fully connected layer for binary classification. However, this approach, while considering the semantic associations, brings certain limitations. Firstly, under the self-attention mechanism, although BERT can capture contextual information, it tends to focus on all words, and the semantics of a record may be influenced by irrelevant words from another record, failing to accurately capture the crucial and subtle differences between record pairs [17]. It implies that even if two records exhibit high consistency in key features, their importance may be relatively diminished when computing in conjunction with other non-critical descriptive words. As a result, the model may not be able to accurately focus on the core information that most determines whether two records are match, thus affecting the model’s precision. Furthermore, the global information generated by BERT for record pairs does not directly represent the semantics of each individual record within the pair.

Additionally, concatenating record pairs into a single sequence inevitably leads to increased computational complexity. This limitation hinders the practical application of blocking techniques to effectively shrink the search space, consequently leading to an increase in computational time complexity up to the order of $O(n^2)$ [18].

In this paper, we propose a new model named DIBER (i.e., Deep Interaction based on BERT Entity Resolution). Inspired by the architectures of existing models such as SBERT [19] and ColBERT [20], our model adopts a Siamese network structure to facilitate flexible application of blocking techniques. Due to the inherent limitations of such a structure in capturing deep interactions between pairs of records, we design a series of modules that enhance the model’s interactive capabilities, thereby enabling it to discern correlations within record pairs, extract granular disparities, and identify pivotal matching features, ultimately enhancing overall matching accuracy. Additionally, by inputting each record individually into BERT, we can truncate the maximum sequence length, which effectively reduces the parameter requirements of the model. To cater to heterogeneous datasets, which consist of varying structures, we concatenate individual records into strings, thus breaking down the strict boundaries between attributes and fostering extensive applicability across various scenarios. In summary, the contributions are as follows:

- We propose a novel model named DIBER, which can be simultaneously applied to both blocking and matching.
- Our proposed model is highly sensitive to nuanced discrepancies within record pairs, dynamically modulating its attention to assign increased weight to critical disparities. Moreover, the model effectively extracts features that play a critical role in determining the matching outcome, thereby achieving a significant enhancement in performance.
- We have conducted extensive experiments to evaluate on 10 datasets without injecting domain knowledge, and the results show that DIBER outperforms some state-of-the-art (SOTA) methods significantly.

The subsequent sections of the paper are outlined as follows. Related work is introduced in section 2. The problem definition and the overall framework of the model are delineated in section 3. The detailed implementations of matching and blocking are elaborated in section 4 and section 5, respectively. The detailed analysis of extensive experimental results from 10 datasets is presented in section 6. The conclusions are summarized in section 7.

2 Related Work

ER stands as a cornerstone in the fields of data mining and artificial intelligence. To effectively handle large-scale datasets without succumbing to the high computational complexity inherent in direct pairwise matching, ER is typically divided into two crucial steps: blocking and matching. Initially, the original datasets are carefully parti-

tioned using blocking technology, breaking them down into smaller-sized blocks, significantly reducing unnecessary comparison counts and conserving computational resources. Then, the matching process is conducted within these blocks.

The concept of ER is formally introduced in [5], and the problem of ER is first defined as a classification problem in [4]. The blocking techniques are applied to the ER task in [21]. With the advancement of technology, research in this field has become more systematic and scientific, and various methods have been proposed.

For blocking, traditional methods include nearest neighbor sequence indexing [22], clustering [23], and hash-based methods [24], which typically assume that two tables share the same schema. To our knowledge, the earliest work is DeepER [25], which first converts tuples into distributed representations and then employs locality sensitive hashing to ensure that similar tuples obtain the same hash code. AutoBlock [26] is a novel hands-off blocking framework for ER, based on similarity-preserving representation learning and nearest neighbor search. DeepBlocker [27] defines a large space of DL solutions for blocking and develops eight representative solutions in this space.

For matching, the three typical categories of methods are rule-based, crowd-sourcing, and machine learning. The continuous evolution of DL technology has led to its widespread application in ER. DeepER transforms each record into a vector and measure the similarity between pairs of records. However, relying solely on a single representation may not adequately capture the intricate nuances and distinctive features within records. DeepMatcher [28] provides a design space for solving ER problem and has demonstrated excellent performance on various benchmark datasets. This method performs pairwise comparisons at the attribute level and derives matching results based on similarity calculations. The Seq2SeqMatcher [29] adheres to the align-compare-aggregate pattern, capturing token-level representations and fostering semantic connections between tokens. It enables to flexibly compare token between different attributes, thereby resolving the issue of heterogeneous datasets.

With the rise of PLM such as BERT, the field of ER has entered a new stage. These models are pre-trained on large-scale unlabeled text data, thereby acquiring rich linguistic structure and semantic representation capabilities. [30] analyzes the performance of four SOTA attention-based Transformer architectures (BERT, XLNet [31], RoBERTa [32] and DistilBERT [33]) in the context of ER problems. Ditto fine-tuning PLMs, integrates domain knowledge and applies data augmentation, but its reliance on the [CLS] token output might overlook detailed semantic information. JointBERT [15] innovatively combines binary and multi-class tasks, improving ER via dual-objective training; however, its performance is contingent upon ample labeled data, which poses a challenge in low-resource scenarios.

To reduce the dependence on large amounts of labeled data, several new models have been proposed. DADER [36] greatly reduces the reliance on expensive manually labeled data by incorporating domain adaptation techniques and smart data utilization strategies. CLER [37] proposes an end-to-end iterative Co-learning framework, jointly training the blocker and the matcher by leveraging their cooperative relationship. It iteratively generates and updates pseudo labels as a bridge to facilitate knowledge sharing between them, thereby enhancing the supervisory information and effectively addressing entity parsing tasks under low-resource scenarios.

Despite significant progress in ER due to DL and PLM, strategies remain crucial for effectively leveraging these technologies in scenarios of data scarcity, along with the ability to flexibly integrate blocking and matching components. Our proposed framework tackles this challenge by thoroughly exploring subtle semantic associations and discrepancies between records, effectively addressing these issues.

3 DIBER Model for Blocking and Matching Records in Two Datasets

This section describes the design of our DIBER model for blocking and matching records in two datasets. We firstly introduce the definition of the ER problem, and then present the overall framework of the model.

3.1 Problem Definition

In this work, a record of a dataset represents a description of a real-world entity such as a person, product and company, by storing the values of attributes of the entity in a table/relation. Let table \mathbf{A} and table \mathbf{B} be two sets of records with $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_{|\mathbf{A}|}\}$ and $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{|\mathbf{B}|}\}$, where $|\mathbf{A}|$ and $|\mathbf{B}|$ are the number of the records in tables \mathbf{A} and \mathbf{B} , respectively. For a record \mathbf{a} in \mathbf{A} , we will discuss the set of all words of \mathbf{a} , which is also denoted by $\mathbf{a} = \{a_1, a_2, \dots, a_u\}$, where u is the number of words in \mathbf{a} , in Fig. 1, for example, $\mathbf{a}_{431} = \{\text{microsoft, licenses, win, svr, 2003, ext, conn lic, r3900292, microsoft, licenses, 3371.85}\}$. Similarly, for a record \mathbf{b} in \mathbf{B} , in Fig. 1, for example, $\mathbf{b}_{1678} = \{\text{microsoft, r39-00292, open, win, svr, 2003, ext, conn, 1863.78}\}$.

Matching. For $\mathbf{a} \in \mathbf{A}$ and $\mathbf{b} \in \mathbf{B}$, the purpose of matching is to determine whether the record pair (\mathbf{a}, \mathbf{b}) refer to a same real-world entity. If \mathbf{a} and \mathbf{b} point to the same entity, we call it a match, then $y = 1$; otherwise, if \mathbf{a} and \mathbf{b} point to two different entities, we call it a non-match, then $y = 0$. The probability $P(y | \mathbf{a}, \mathbf{b})$ will be the output of our model. Generally, the number of non-matching record pairs is much greater than the number of matching record pairs.

Blocking. The purpose of blocking is to find potential matching pairs between tables \mathbf{A} and \mathbf{B} , so that the matching process only needs to judge these pairs of candidate sets rather than all entity pairs, thereby reducing the computational cost of matching. We assume that there are unlabeled records in tables \mathbf{A} and \mathbf{B} . The input of blocking is all pairs from $\mathbf{A} \times \mathbf{B}$, the output is a set of similar candidate pairs.

3.2 Framework of DIBER Model

As illustrated in Fig. 2, our model consists of three components: Encoder, Blocker, and Matcher. For the Encoder, its purpose is to map each word in the record to a high-dimensional vector, obtaining rich semantic information within the record. Common processing methods include GloVe [34], Word2Vec [35], fastText [38], BERT [16].

Since the same token may correspond to different semantics in various contexts, and BERT can effectively capture abundant context information to address the issue of polysemy, we use BERT as our encoder. We input all the records from tables \mathcal{A} and \mathcal{B} into BERT, respectively, obtaining their corresponding vector representation sets \mathcal{R}_A and \mathcal{R}_B . For BERT-base, its output includes an embedding layer and 12 hidden layers. We use the embedding layer as the vector representation for Blocker, and select one hidden layer as the vector representation for Matcher.

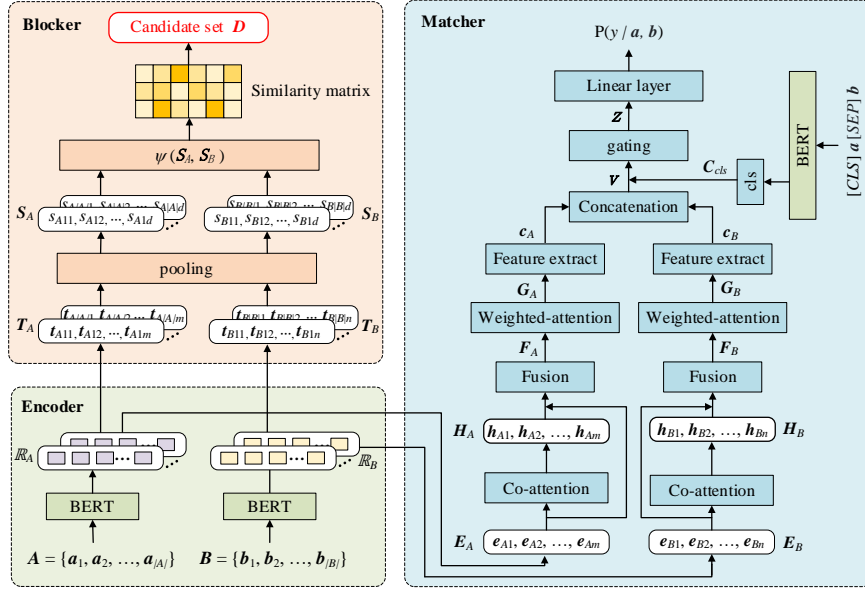


Fig. 2. Architecture of DIBER for blocking and matching records in two tables \mathcal{A} and \mathcal{B}

In the matcher component of our model DIBER (see Fig. 2), the matcher consists of a co-attention mechanism, fusion layer, weighted attention mechanism, feature extraction layer, gated fusion, and prediction layer. Firstly, the outputs \mathcal{R}_A and \mathcal{R}_B of Encoder component are used as the inputs of the Matcher component, the matrix or vector representations of matching are denoted as $E_A = (e_{A1}, e_{A2}, \dots, e_{Am}) \in \mathcal{R}^{d \times m}$ and $E_B = (e_{B1}, e_{B2}, \dots, e_{Bn}) \in \mathcal{R}^{d \times n}$, where $\mathcal{R}^{d \times n}$ is the set of $d \times n$ matrices, $e_{Ai} \in \mathcal{R}^{d \times 1}$ is the vector representation of the token tk_i ($1 \leq i \leq m$) in \mathbf{a} , $e_{Bj} \in \mathcal{R}^{d \times 1}$ is the vector representation of the token tk_j ($1 \leq j \leq n$) in \mathbf{b} , m and n are the number of tokens in records \mathbf{a} and \mathbf{b} , respectively, and d is dimension of the BERT output vector. By concatenating \mathbf{a} and \mathbf{b} into a single sequence with the special token [SEP], we input the sequence into BERT to acquire an overall semantic representation C_{cls} for the record pair. Secondly, by inputting E_A and E_B into the co-attention layer, we capture the deep interactive information. Thirdly, we integrate the semantic information and interaction information of the two records to derive their associations and subtle differences. Then, by employing a weighted attention mechanism, we assign higher weights to critical differences, and

utilize a feature extractor to derive fine-grained, pivotal matching features which are local. Finally, via a gating mechanism, we fuse the extracted local vectors with the global vector C_{cls} , and pass the fused output through a simple fully connected layer to yield the final result $P(y | \mathbf{a}, \mathbf{b})$.

In the blocker component, we deal with all records in tables \mathbf{A} and \mathbf{B} , with their token-level vector representation sets denoted as $\mathbf{T}_A = \{\mathbf{t}_{A1}, \mathbf{t}_{A2}, \dots, \mathbf{t}_{A|\mathbf{A}|}\}$, $\mathbf{T}_B = \{\mathbf{t}_{B1}, \mathbf{t}_{B2}, \dots, \mathbf{t}_{B|\mathbf{B}|}\}$, respectively, where $\mathbf{t}_{Ai} = (\mathbf{t}_{Ai1}, \mathbf{t}_{Ai2}, \dots, \mathbf{t}_{Aim}) \in \mathcal{R}^{d \times m}$ is a matrix for \mathbf{a}_i in \mathbf{A} , $\mathbf{t}_{Bj} = (\mathbf{t}_{Bj1}, \mathbf{t}_{Bj2}, \dots, \mathbf{t}_{Bjm}) \in \mathcal{R}^{d \times n}$ is a matrix for \mathbf{b}_j in \mathbf{B} . Firstly, we employ pooling operations to aggregate the token-level representations into record-level representations. Then, we use a function to calculate similarity scores between any \mathbf{a}_i in \mathbf{A} and every record within \mathbf{B} , thus obtaining a set of potential matching records for \mathbf{a}_i . This process is iterated for each record in \mathbf{A} , ultimately yielding the candidate match set \mathbf{D} .

The other letters in the figure represent the outputs of each module.

4 Matching Records in Two Tables

For the matcher, we primarily enhance the model’s performance by focusing on the following aspects: (1) Harnessing rich semantic features in records; (2) Facilitating deep interaction between records to obtain their associations; (3) Delving into fine-grained discrepancies and pivotal matching features; (4) Fusing the local and global features to obtain multi-grained and comprehensive information. Next, we will introduce the implementation process in detail. Since \mathbf{a} and \mathbf{b} are processed in a symmetrical manner, we will only show the process for \mathbf{a} .

4.1 Co-Attention Layer

After obtaining the vector representation of record pairs, it is necessary to perform soft alignment between the record pairs to obtain their interaction information. For each token tk_i in \mathbf{a} , its vector is $\mathbf{e}_{Ai} \in \{\mathbf{e}_{A1}, \mathbf{e}_{A2}, \dots, \mathbf{e}_{Am}\}$, firstly, we calculate its similarity with all tokens in \mathbf{b} to obtain a similarity vector $\boldsymbol{\alpha}_i$. Then, $\boldsymbol{\alpha}_i$ is normalized using *softmax* function, thereby obtaining the interaction information between \mathbf{a} and \mathbf{b} at the token level. The similarity value indirectly represents the importance of tk_i with respect to each token in \mathbf{b} :

$$\boldsymbol{\alpha}_i = (\mathbf{W}_k \cdot \mathbf{E}_B)^\top \cdot (\mathbf{W}_q \cdot \mathbf{e}_{Ai}) \quad (1)$$

$$\boldsymbol{\alpha}'_i = \text{softmax}(\boldsymbol{\alpha}_i) \quad (2)$$

$$\mathbf{h}_{Ai} = (\mathbf{W}_v \cdot \mathbf{E}_B) \cdot \boldsymbol{\alpha}'_i \quad (3)$$

where the matrices $\mathbf{W}_q, \mathbf{W}_k$ and $\mathbf{W}_v \in \mathcal{R}^{d \times d}$ are randomly initialized and jointly learned during the training process; $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}'_i \in \mathcal{R}^{n \times 1}$ are column vectors of n dimensions; $\mathbf{h}_{Ai} \in \mathcal{R}^{d \times 1}$ is the interaction vector of tk_i with respect to \mathbf{b} ; “ \top ” represents the transpose of a matrix. Moreover, we can obtain the interaction matrix $\mathbf{H}_A = (\mathbf{h}_{A1}, \mathbf{h}_{A2}, \dots, \mathbf{h}_{Am}) \in \mathcal{R}^{d \times m}$ regarding \mathbf{a} with \mathbf{b} at the token level.

4.2 Fusion Layer

The fusion layer combines the semantic representation e_{Ai} from the encoding layer with interactive representation h_{Ai} from the co-attention layer, generating a token-level comparison vector. It can emphasize the distinctions among record pairs and extract more discriminative features, thereby allowing the model to sensitively discern subtle nuances between record pairs and better comprehend complex semantics. The specific implementation is as follows:

$$\beta_i = e_{Ai} - h_{Ai} \quad (4)$$

$$\beta'_i = \beta_i \odot \beta_i \quad (5)$$

$$f_{Ai} = e_{Ai} \oplus \beta_i \oplus \beta'_i \quad (6)$$

where β_i and $\beta'_i \in \mathcal{R}^{d \times 1}$ are column vectors; “ $-$ ” represents the subtraction operation between vectors, aiming to capture relational and differential features; “ \odot ” represents the element-wise multiplication operation on vectors, aiming to further accentuate differences, extract more subtle discrepancies, and thereby obtain fine-grained comparative information; “ \oplus ” represents the (vertical) concatenation operation of column vectors. Therefore, we can obtain a vector $f_{Ai} \in \mathcal{R}^{3d \times 1}$ and a fusion matrix $F_A = (f_{A1}, f_{A2}, \dots, f_{Am}) \in \mathcal{R}^{3d \times m}$.

4.3 Weighted Attention Layer

The inspiration for the weighted attention mechanism stems from the observation that in different fields, words with significant discriminative power should be given special attention [39]. In the ER task, the varying degrees of differences between records contribute differently towards determining whether they refer to the same entity. To better utilize the salient disparities in the matching process, we assign high weights to these differences, which can further improve the performance of matching. Therefore, we propose a weighted attention mechanism. Firstly, we calculate the weights using dot product combined with the *softmax* function for normalization, obtaining the weight vector γ'_i . By applying γ'_i to the fusion matrix F_A , we obtain a global vector g_{Ai} with differential weights:

$$\gamma_i = (W_\beta \cdot F_A)^\top \cdot (W_\alpha \cdot f_{Ai}) \quad (7)$$

$$\gamma'_i = \text{softmax}(\gamma_i) \quad (8)$$

$$g_{Ai} = (W_\gamma \cdot F_A) \cdot \gamma'_i \quad (9)$$

where $W_\alpha, W_\beta, W_\gamma \in \mathcal{R}^{3d \times 3d}$ are trainable matrices, while $\gamma_i, \gamma'_i \in \mathcal{R}^{m \times 1}$ and $g_{Ai} \in \mathcal{R}^{3d \times 1}$ are column vectors. Therefore, we can get a weighted matrix $G_A = (g_{A1}, g_{A2}, \dots, g_{Am}) \in \mathcal{R}^{3d \times m}$.

4.4 Feature Extraction Layer

After obtaining the weight information of the records, we hope to achieve a more fine-grained feature representation. We employ sentence-level CNN to extract multi-gram features, capturing the local information of tokens, and the feature extraction layer (which is called $Fel(\cdot)$ in this paper) comprises a set of convolutions, a batch normalization, a non-linear function and max pooling operations [40].

$$\mathbf{c}_A = Fel(\mathbf{G}_A) \quad (10)$$

where $\mathbf{c}_A \in \mathcal{R}^{cg \times 1}$ is a column vector, c is the number of kernels, and g is the kernel size, $\mathbf{G}_A = (\mathbf{g}_{A1}, \mathbf{g}_{A2}, \dots, \mathbf{g}_{Am}) \in \mathcal{R}^{3d \times m}$ is obtained above,. Thus, for \mathbf{E}_A , we can obtain a local vector \mathbf{c}_A that represents fine-grained interaction and key information. For \mathbf{E}_B , similarly, we can obtain a local vector \mathbf{c}_B .

4.5 Gate and Prediction Layer

The gating mechanism (GM) is used to control information flow. We use GM to assign different weights to \mathbf{v} and \mathbf{C}_{cls} , where \mathbf{v} is obtained by concatenating \mathbf{c}_A and \mathbf{c}_B , and \mathbf{C}_{cls} is a vector representing the contextual information of the entire record pair, generated by BERT. Through the combination of local and global information, we can obtain a more comprehensive and enriched feature representation \mathbf{z} :

$$\mathbf{v} = \mathbf{c}_A \oplus \mathbf{c}_B \quad (11)$$

$$\lambda = \sigma(\mathbf{W}_1 \cdot \mathbf{v} + \mathbf{W}_2 \cdot \mathbf{C}_{cls} + b_g) \quad (12)$$

$$\mathbf{z} = (1 - \lambda) \cdot \mathbf{C}_{cls} + \lambda \cdot \mathbf{v} \quad (13)$$

where $\mathbf{W}_1 \in \mathcal{R}^{1 \times 2cg}$ and $\mathbf{W}_2 \in \mathcal{R}^{1 \times d}$ are trainable matrices, $\mathbf{v} \in \mathcal{R}^{2cg \times 1}$ and $\mathbf{C}_{cls} \in \mathcal{R}^{d \times 1}$ are column vectors, and b_g is the corresponding bias term, and σ is a non-linear activation function (say, the function $Sigmoid(\cdot)$ is used in our experiments), λ ($0 \leq \lambda \leq 1$) is a variable used to control the flow of information. When $\lambda = 0$, \mathbf{v} does not take effect. As λ increases, more emphasis is placed on using information from \mathbf{v} with fine-grained interaction during the generation of \mathbf{z} , while the weight of the representation \mathbf{C}_{cls} derived directly from BERT is reduced.

Then, the gated-generated vector \mathbf{z} is fed into a fully connected layer followed by a softmax layer to obtain the final prediction results:

$$p = P(y | \mathbf{a}, \mathbf{b}) = softmax(\mathbf{W} \cdot \mathbf{z} + b) \quad (14)$$

4.6 Model Learning

Given a training set $\mathbf{T} = \{(\mathbf{a}_i, \mathbf{b}_i, y_i)\}_{i=1}^{|\mathbf{T}|}$ containing a series of training examples, $y_i \in \{0, 1\}$ is the ground truth label, we train our model by minimizing the cross-entropy loss function:

$$loss = -\frac{1}{|T|} \sum_{i=1}^{|T|} [y_i \log(p) + (1 - y_i) \log(1 - p)] \quad (15)$$

where $|T|$ is the number of training set, and p is the predicted output of our model as shown in Equation (14).

5 Blocking Records in Two Tables

We propose a feasible blocking method based on the blocking framework template provided in [27]: firstly, we obtain the token-level representation of each record in tables \mathbf{A} and \mathbf{B} , and convert them into a single vector that represents the entire record. Then, we use a similarity measure to efficiently find tuple pairs (\mathbf{a}, \mathbf{b}) with highly similar score, where $\mathbf{a} \in \mathbf{A}$, $\mathbf{b} \in \mathbf{B}$.

5.1 Record Embedding

We use pooling operations to aggregate \mathbf{t}_{Ai} and generate the vector \mathbf{s}_{Ai} for the entire record. The pooling methods we employ include max pooling, min pooling, and average pooling.

$$\mathbf{s}_{Ai} = \text{pooling}(\mathbf{t}_{Ai}) \quad (16)$$

where $\mathbf{s}_{Ai} = (s_{Ai1}, s_{Ai2}, \dots, s_{Aid})^T \in \mathcal{R}^{d \times 1}$ is a column vector. Thus, we can obtain the record representations \mathbf{S}_A for all records in table \mathbf{A} and $\mathbf{S}_A = \{s_{A1}, s_{A2}, \dots, s_{A|A|}\}$. Similarly, we can obtain $\mathbf{s}_{Bi} \in \mathcal{R}^{d \times 1}$, and $\mathbf{S}_B = \{s_{B1}, s_{B2}, \dots, s_{B|B|}\}$.

5.2 Candidate Pair Generation

We leverage a function $\psi(\cdot)$ for efficiently identifying the corresponding similarity vector in table \mathbf{B} for each record in table \mathbf{A} . In our experiment, we utilize the cosine similarity function as our $\psi(\cdot)$ function. There are typically two methods for selecting similar record pairs: select records with similarity scores exceeding a certain threshold (e.g., $\text{threshold} = 0.8$), or select the $\text{top-}k$ records with the highest similarity scores. Compared with the threshold method, employing the $\text{top-}k$ approach effectively mitigates the loss of potentially similar records due to inappropriate threshold settings. Moreover, for datasets with diverse scales and distributions, the $\text{top-}k$ method consistently identifies record pairs of a certain quality level by selecting an appropriate k value, thereby exhibiting robustness against variations in the data. Thus, we employ the $\text{top-}k$ method to select record pairs. Specifically, for each record \mathbf{a}_i in table \mathbf{A} , we initially compute the cosine scores between its vector $\mathbf{s}_{Ai} \in \mathbf{S}_A$ and every $\mathbf{s}_{Bj} \in \mathbf{S}_B$. Subsequently, based on these cosine scores, we sort all records in table \mathbf{B} in descending order of similarity and select the $\text{top-}k$ records to form a subset $\mathbf{K}_i \subseteq \mathbf{B}$. For \mathbf{a}_i , we can obtain k

possible match pairs (a_i, b_j) , where $b_j \in K_i$. The matching records collectively constitute the candidate set D_i for a_i . It can further yield the candidate set D for all records in A .

$$\text{sim}(i, j) = \psi(\mathbf{s}_{A_i}, \mathbf{s}_{B_j}), \quad \forall i, j \quad (1 \leq i \leq |A|, 1 \leq j \leq |B|) \quad (17)$$

$$K_i = \text{top}_k\{\text{sim}(i, j_1), \text{sim}(i, j_2), \dots, \text{sim}(i, j_{|B|})\} \quad (18)$$

$$D_i = \{(a_i, b_j) \mid b_j \in K_i\} \quad (19)$$

$$D = \{D_1, D_2, \dots, D_{|A|}\} \quad (20)$$

where the function $\text{top}_k(\cdot)$ is used to find the top- k most similar pairs between record a_i and all records in table B .

6 Experiments

In this section, we evaluate our model DIBER and report the experimental results. In section 6.1, we provide the experimental settings, in section 6.2, we preprocess the data, in section 6.3, we evaluate the overall performance of the model, and in section 6.4, we conduct an ablation study.

6.1 Experimental Settings

Benchmark Datasets. To evaluate our model, we use two kinds of datasets to conduct experiment: standard structured datasets and dirty datasets, from various domains. There are six structured datasets, and four dirty datasets as illustrated in Table 1. For structured datasets, we use Amazon-Google (abbr. A-G), Beer, DBLP-ACM₁ (abbr. D-A₁), Walmart-Amazon₁ (abbr. W-A₁), iTunes-Amazon₁ (abbr. I-A₁), DBLP-Scholar₁ (abbr. D-S₁). These datasets are publicly available and have been extensively used for the ER task. For the dirty datasets, we use DBLP-ACM₂ (abbr. D-A₂), Walmart-Amazon₂ (abbr. W-A₂), iTunes-Amazon₂ (abbr. I-A₂), DBLP-Scholar₂ (abbr. D-S₂) datasets, which are generated from their respective original ones (e.g., I-A₂ is generated from I-A₁). These datasets are generated by randomly moving the value of each attribute to the attribute *title* in the same record with 50% probability. By the dirty datasets, we can measure the robustness of the model. Each dataset is split into training, validation, and test sets using the ratio of 3:1:1, provided by [28].

Baselines. We compare our model with the following baselines: (1) Magellan: A classical non-DL ER baseline that has demonstrated strong performance across various datasets. In this approach, a set of classifiers is trained using a rich feature set. (2) Deep-Matcher: A SOTA matching model which provides a categorization of DL solutions and defines a design space for these solutions. It requires that the record pairs to be matched have the same pattern. (3) Ditto: A pre-trained Transformer-based language model. The model casts ER as a sequence-pair classification problem, and applies three optimization techniques, including leveraging domain knowledge, summarizing long

entries, and augmenting training data. (4) DeepBlocker: A SOTA blocking model based on DL which defines a large space of DL solutions for blocking, which contains solutions of varying complexity.

For Magellan, DeepMatcher and DeepBlocker, we directly use the results reported in their papers. For fair comparison, the Ditto result in our paper is obtained by re-run the open-source using the BERT pre-trained model with the epoch and batch size of 40 and 32 respectively. The data augmentation technique employed is deleting a span of tokens. And all other hyper-parameters are set as described in original paper.

Table 1. Statistics of two kinds of datasets used in our experiments

Type	Datasets	Domain	Size	#Pos.	#Att.
Structured	A-G	Software	11,460	1,167	3
	Beer	Beer	450	68	4
	D-A ₁	Citation	12,363	2,220	4
	W-A ₁	Electronics	10,242	962	5
	I-A ₁	Music	539	132	8
Dirty	D-S ₁	Citation	28,707	5,347	4
	W-A ₂	Electronics	10,242	962	5
	D-A ₂	Citation	12,363	2,220	4
	D-S ₂	Citation	28,707	5,347	4
	I-A ₂	Music	539	132	8

Metric for Matching. We use F1 score to measure the matching accuracy, which is defined as $F1 = 2PR/(P + R)$, where P is *precision* and R is *recall*. *Precision* is defined as $P = |TP|/(|TP|+|FP|)$ and *recall* is defined as $R = |TP|/(|TP|+|FN|)$. The F1 score on the test datasets as our metric. Ideally, we want a high F1 score.

Metric for Blocking. We utilize *recall* and the Candidate Set Size Ratio (*CSSR*) to evaluate our blocking, where *recall* is defined as $recall = |\mathbf{R} \cap \mathbf{D}|/|\mathbf{D}|$, *CSSR* is defined as $CSSR = |\mathbf{D}|/|\mathbf{A} \times \mathbf{B}|$, with \mathbf{D} being the candidate set generated by the blocker, and \mathbf{R} representing the actual matches between tables \mathbf{A} and \mathbf{B} . Ideally, we aim for high *recall*, low *CSSR*, and low *runtime*.

Parameters. The convolutional layers have kernel sizes set to [1, 2, 3], with $g = 128$ kernels for each size. The Adam optimizer is employed for parameter tuning, initializing with a learning rate of $3e-5$. The batch size is set to 32 for the D-S₁ and D-S₂ datasets, while for all other datasets, the batch size is reduced to 16. We fix the max sequence length to be 128. The training process runs for a fixed number of epochs, which is set to 15, 30 or 50 based on the size of the datasets. We present the average F1 score for results of three independent experiments. In the context of blocking, k represents the number of records from table \mathbf{B} that each record in table \mathbf{A} is paired with based on the cosine similarity. The size of k has a significant impact on both the *recall* and *CSSR*.

We conduct experiments using the same value of k as in DeepBlocker [27]. We implement DIBER in PyTorch and the Transformers library and conduct all experiments on a single RTX 3090 GPU.

6.2 Data Preprocessing

We preprocess the input data to enhance the performance of DIBER. In many NLP tasks, PLMs are sensitive to the order of input record pairs, especially when dealing with entity relationships. Therefore, we adopt the method of swapping records while retaining their original labels to increase both the quantity and diversity of training data. For instance, (a, b) is converted to (b, a) and their labels are the same. It enables the model to capture information more comprehensively and effectively enhance its generalization capability. Moreover, some datasets contain punctuation marks that do not carry semantic information and are irrelevant to the matching decision, such as “(”, “)”, “/”, and “-”. Therefore, we filter out these punctuation marks, retaining only the most critical information. It enables the model to focus on the elements directly related to the ER task, thereby enhancing its performance.

6.3 Effectiveness Evaluation

Matching. In the matching, we compare the DIBER with Magellan, DeepMatcher, and Ditto. Table 2 summarizes the F1 scores of all models on all datasets. We use bold text to emphasize the highest scores in each row. As shown in Table 2, the average F1 scores for Magellan, DeepMatcher, Ditto and DIBER model across all datasets are 74.03, 82.24, 89.99 and 91.83, respectively. We can infer that DIBER exhibits outstanding overall performance.

To be specific, DIBER outperforms Magellan and DeepMatcher in terms of F1 score on all datasets. Especially on some special datasets with limited training data or for product description, such as Beer, W-A₁, W-A₂, I-A₁, and I-A₂, our model demonstrates significant improvement in performance. In these datasets, the shared common vocabulary among record pairs is relatively scarce, demanding that the model effectively excavates and extracts abundant semantic information between record pairs. It indicates that our model possesses a strong capability for semantic understanding. Although Ditto has a strong semantic understanding ability, our model achieve F1 score increases of 8.9, 1.8, and 3.5 on the Beer, I-A₁, and I-A₂ datasets, respectively. It is mainly due to the high sensitivity of our model to subtle differences, which allows it to accurately unearth and profoundly grasp the intrinsic connections between records. Therefore, when the amount of training data is limited, DIBER demonstrates significant performance improvements. This attribute enables DIBER to be more widely applicable in scenarios with insufficient data.

Compared with Ditto, the F1 score of DIBER has been improved by 3.2, 1.2, and 0.5 respectively on datasets A-G, W-A₁ and W-A₂. For these datasets, record pairs referring to different entities often share some common terms, yet exhibit subtle disparities in a few pivotal words, such as product models, which plays a critical role in the matching process. This indicates that DIBER is capable of accurately identifying and extracting

subtle differences between record pairs, particularly those nuanced variations that are critically important for ER. It assigns significant weight values to such differences, enabling the model to discern subtle yet crucial matching information within record pairs that determines the matching results.

Table 2. F1 score of DIBER on all datasets

Type	Datasets	Magellan	DeepMatcher	Ditto	DIBER
Structured	A-G	49.1	69.3	74.5	77.7
	Beer	78.8	78.8	87.5	96.4
	D-A ₁	98.4	98.4	99.1	98.8
	W-A ₁	71.9	67.6	81.5	82.7
	I-A ₁	91.2	88.5	96.4	98.2
Dirty	D-S ₁	92.3	94.7	94.7	94.8
	W-A ₂	37.4	53.8	79.9	80.4
	D-A ₂	91.9	98.1	98.4	98.4
	D-S ₂	82.5	93.8	95.0	94.5
	I-A ₂	46.8	79.4	92.9	96.4
average		74.03	82.24	89.99	91.83

On other datasets, our model presents room for improvement. On the D-S₁ dataset, we manage to lift the F1 score by a mere increment of 0.1, while maintaining comparable performance on the D-A₂ dataset. However, on the D-A₁ and D-S₂ datasets, the F1 scores are marginally decreased by 0.3 and 0.5, respectively. It can be attributed to the abundance of repeated vocabulary among record pairs within these datasets, which results in weaker distinguishing features, consequently posing a lower threshold of challenge for all models, including ours, in discerning semantic similarities.

Blocking. Table 3 presents the experimental results of our model under different k values and pooling operations across various datasets, comparing its performance with that of the DeepBlocker. The boldface entries in each row indicate the highest recall achieved for each dataset.

It shows that the blocking solution proposed by us achieves a high *recall* on small candidate sets. And even with larger candidate sets generated, our *runtime* is not particularly long. In terms of *recall*, our proposed blocker outperforms DeepBlocker on all datasets except for D-S₁ and D-S₂. This is mainly because there exist a large number of common terms in the record pairs within datasets D-S₁ and D-S₂, which aid the model in better capturing the semantic information of the records. For the datasets W-A₁ and W-A₂, which have fewer shared terms and distinct expressions of key attributes, our blocker has improved *recall* by 6.1 and 9.8, respectively. For ER tasks, using the CLS token output directly from BERT or averaging the token-level embedding to represent the entire record as a vector yields unsatisfactory results compared with using max pooling or min pooling.

Table 3. Experimental results of different methods on various datasets

datasets	k	$ \mathcal{D} $	CSSR (*10 ⁻³)	runtime	Recall				
					mean	max	min	CLS	Deep-Blocker
A-G	50	68.2k	15.4	14.1s	98.6	96.6	96.2	76.1	97.1
Beer	50	217.3k	16.6	19.1s	86.8	86.8	85.3	33.8	-
D-A ₁	5	13.1k	2.1	12.6s	99.3	99.5	99.7	87.8	99.6
W-A ₁	20	51.1k	0.9	73.9s	96.5	98.2	98.3	53.5	92.2
I-A ₁	50	1036.1k	2.7	477.6s	69.7	93.2	90.9	30.3	-
D-S ₁	150	392.4	2.3	209.1s	94.3	96.6	97.4	51.8	98.1
W-A ₂	20	51.1k	0.9	73.2s	96.5	98.2	98.1	51.8	88.0
D-A ₂	5	13.1k	2.1	12.5s	99.3	99.4	99.7	63.3	99.6
D-S ₂	150	392.4k	2.3	208.6s	93.9	96.5	97.2	45.9	98.1
I-A ₂	50	1036.1k	2.7	439.7s	65.9	89.4	87.1	34.8	-

6.4 Ablation Study

Our model consists of several components: data preprocessing (DP), co-attention mechanism (CAM), fusion, weighted attention mechanism (WAM), and feature extraction layers (FEL). To verify the effectiveness of each module in DIBER, we will perform an ablation study by gradually eliminating one component at a time.

We compare DIBER with these variants, presenting the comparative results in Table 4: (1) D-DP, which the data preprocessing is omitted. As observed, the average F1 decreased by 3.54 compared with DIBER. This decline highlights the importance of DP, which enables the model to learn more comprehensive and diverse features while focusing on the core content of records. (2) D-CAM, which removes the co-attention mechanism module and directly passes the semantic information of record pairs to the fusion module. It is observed that the F1 decreases by nearly 24 on average without CAM. It indicates that CAM is able to align disparate record representations, enabling the model to delve more profoundly into the interactions between records and associated information, which is a crucial factor in enhancing model performance. (3) D-Fusion, which uses $e_i \oplus (e_i - h_i) \oplus (e_i \odot h_i)$ instead of the fusion mechanism in DIBER. The result shows that the F1 score under D-Fusion has decreased by an average of 3.11, indicating that the fusion in DIBER excels at capturing and highlighting the subtle but critical distinguishing features between record pairs. (4) D-WAM, which assigns equal weights to all token-level differences. For datasets I-A₁ and I-A₂, which describe music, they have a higher number of positive samples, and for negative samples, their semantics vary significantly. Therefore, removing WAM does not affect their F1. For other datasets, the removal of WAM lead to an average decrement of 0.65 in F1, demonstrating that giving greater emphasis to key discrepancies indeed enhances the F1 effectively. (5) D-FEL, which substitutes FEL with an average pooling and a simple fully connected layer. We find a drop by at most 6.7 in F1 when removing FEL. It is due to

FEL can extract finer-grained key matching information, thereby enhancing the accuracy and efficiency of the model. According to the results, all of the components contribute effectively to the performance improvement.

Table 4. F1 score of the DIBER and its variants

Datasets	D-DP	D-CAM	D-Fusion	D-WAM	D-FEL	DIBER
A-G	73.1	47.2	73.8	77.0	71.3	77.7
Beer	92.9	47.6	87.5	96.3	89.7	96.4
D-A ₁	96.8	95.9	97.8	98.4	98.3	98.8
W-A ₁	72.9	30.4	77.7	81.1	78.8	82.7
I-A ₁	94.5	79.2	96.4	98.2	94.6	98.2
D-S ₁	93.5	90.3	94.2	94.6	93.4	94.8
W-A ₂	74.6	28.8	74.1	79.1	78.2	80.4
D-A ₂	97.7	95.5	97.2	98.1	97.3	98.4
D-S ₂	93.8	89.9	94.0	93.9	93.5	94.5
I-A ₂	92.9	80.0	94.5	96.4	91.5	96.4

7 Conclusion

In this paper, we propose a novel entity resolution Siamese network model DIBER. This model can be applied to both matching and blocking steps. For the matching step, we use a co-attention mechanism to interact between record pairs, fuse semantic information with interaction information, and compare the differences between two records. We use weighted attention to assign high weight values to key differences and use a feature extractor to extract fine-grained differences and key matching features. In addition, we conduct experiments on a blocking method to validate that our model can be flexibly applied to blocking tasks. We conducted extensive experiments on ten datasets, and the results show that our model outperforms some SOTA models.

For future work, we will integrate the blocking and the matching task to develop an end-to-end ER system, further enhancing the performance of ER tasks. In addition, extensive experiments have shown that semantic understanding capability is crucial for performance. We will propose a semantic data augmentation method to optimize it.

References

1. Getoor, L., Machanavajjhala, A.: Entity resolution: Theory, Practice & Open Challenges. Proc. VLDB Endow. **5**(12), 2018-2019 (2012)
2. Mo, L., Cheng, R., Li, X., Cheung, D. W., Yang, X. S: Cleaning Uncertain Data for Top-k Queries. In: ICDE, pp. 134-145 (2013)
3. Winkler, W. E.: Overview of Record Linkage and Current Research Directions. Bureau of the Census. **25**(4), 603-623 (2006)
4. Wang, J., Li, G., Yu, J. X., Feng, J.: Entity Matching: How Similar is Similar. Proc. VLDB

- Endow. **4**(10), 622-633 (2011)
5. Newcombe, H. B., Kennedy, J. M., Axford, S. J., James, A. P.: Automatic Linkage of Vital Records: Computers can be used to extract "follow-up" statistics of families from files of routine records. *Science*. **130**(3381), 954-959 (1959)
 6. Bilenko, M., Mooney, R. J.: Adaptive duplicate detection using learnable string similarity measures. In: *KDD*, pp. 39-48 (2003)
 7. Singh, R., Meduri, V. V., Elmagarmid, A., Madden, S., Papotti, P., Quiané-Ruiz, J. A., SolarLezama A., Tang, N.: Synthesizing entity matching rules by examples. *Proc. VLDB Endow.* **11**(2), 189-202 (2017)
 8. Li, L., Li, J., Gao, H.: Rule-Based Method for Entity Resolution. *IEEE Trans. Knowl. Data Eng.* **27**(1), 250-263 (2014)
 9. Franklin, M. J., Kossmann, D., Kraska, T., Ramesh, S., Xin, R.: CrowdDB: answering queries with crowdsourcing. In: *SIGMOD*, pp. 61-72 (2011)
 10. Vespapunt, N., Bellare, K., Dalvi, N.: Crowdsourcing algorithms for entity resolution. *Proc. VLDB Endow.* **7**(12), 1071-1082 (2014)
 11. Singla, P., Domingos, P.: Entity resolution with markov logic. In: *ICDM*, pp. 572-582 (2006)
 12. Dharavath, R., Kumar, C.: Entity resolution based EM for integrating heterogeneous distributed probabilistic data. *J. Syst. Softw.* **107**, 93-109 (2015)
 13. Cohen, W. W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: *KDD*, pp. 475-480 (2002)
 14. Li, Y., Li, J., Suhara, Y., Doan, A., Tan, W. C.: Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* **14**(1), 50-60 (2020)
 15. Peeters, R., Bizer, C.: Dual-objective fine-tuning of BERT for entity matching. *Proc. VLDB Endow.* **14**, 1913-1921 (2021)
 16. Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *NAACL-HLT*, pp. 4171-4186 (2019)
 17. Ye, C., Jiang, S., Zhang, H., Wu, Y., Shi, J., Wang, H., Dai, G.: JointMatcher: Numerically-aware entity matching using pre-trained language models with attention concentration. *Knowl. Based Syst.* 251: 109033 (2022)
 18. Li, B., Miao, Y., Wang, Y., Sun, Y., Wang, W.: Improving the Efficiency and Effectiveness for BERT-based Entity Resolution. In: *AAAI*, vol. **35**, pp. 13226-13233 (2021)
 19. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *EMNLP/IJCNLP*, pp. 3980-3990 (2019)
 20. Khattab, O., Zaharia, M.: ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In: *SIGIR*, pp. 39-48 (2020)
 21. Hernández, M. A., Stolfo, S. J.: The merge/purge problem for large databases. In: *SIGMOD*. **24**(2), 127-138 (1995)
 22. Chu, X., Ilyas, I. F., Koutris, P.: Distributed data deduplication. *Proc. VLDB Endow.* **9**(11), 864-875 (2016)
 23. Bilenko, M., Kamath, B., Mooney, R. J.: Adaptive Blocking: Learning to Scale Up Record Linkage. In: *ICDM*, pp. 87-96. *IEEE* (2006)
 24. Caragea, D., Cook, D., Honavar, V. G.: Gaining insights into support vector machine pattern classifiers using projection-based tour methods. In: *KDD*, pp. 251-256 (2001)
 25. MESTS, J., Tang, M. O. N.: Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* **11**(11), 1454-1467 (2018)
 26. Zhang, W., Wei, H., Sisman, B., Dong, X. L., Faloutsos, C., Page, D.: Autoblock: A Hands-off Blocking Framework for Entity Matching. In: *WSDM*, pp. 744-752 (2020)
 27. Thirumuruganathan, S., Li, H., Tang, N., Uuzzani, M., Govind, Y., Paulsen, D., Fung, G.,

- Doan, A.: Deep Learning for Blocking in Entity Matching: A Design Space Exploration. *Proc. VLDB Endow.* **14**(11), 2459-2472 (2021)
28. Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., Raghavendra, V.: Deep Learning for Entity Matching: A Design Space Exploration. In: *SIGMOD*, pp. 19-34 (2018)
 29. Nie, H., Han, X., He, B., Sun, L., Chen, B., Zhang, W., Wu S., Kong, H.: Deep Sequence-to-Sequence Entity Matching for Heterogeneous Entity Resolution. In: *CIKM*, pp. 629-638 (2019)
 30. Brunner, U., Stockinger, K.: Entity Matching with Transformer Architectures: A Step Forward in Data Integration. In: *EDBT*, pp. 463-473 (2020)
 31. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., Le, Q. V.: XLNet: Generalized autoregressive pretraining for language understanding. In: *NeurIPS*, pp. 5754-5764 (2019)
 32. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy O., Lewis M., Zettlemoyer L., Stoyanov, V.: RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692* (2019)
 33. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019)
 34. Pennington, J., Socher, R., Manning, C. D.: GloVe: Global Vectors for Word Representation. In: *EMNLP*, pp. 1532-1543 (2014)
 35. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. In: *NIPS*, pp. 3111-3119 (2013)
 36. Tu, J., Han, X., Fan, J., Tang, N., Chai, C., Li, G., Du, X.: DADER: Hands-Off Entity Resolution with Domain Adaptation. *Proc. VLDB Endow.* **15**(12), 3666-3669 (2022)
 37. Wu, S., Wu, Q., Dong, H., Hua, W., Zhou, X.: Blocker and Matcher Can Mutually Benefit: A Co-Learning Framework for Low-Resource Entity Resolution. *Proc. VLDB Endow.* **17**(3), 292-304 (2023)
 38. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching Word Vectors with Subword Information. *Trans. Assoc. Comput. Linguist.* **5**, 135-146 (2017)
 39. Zhang, D., Nie, Y., Wu, S., Shen, Y., Tan, K. L.: Multi-Context Attention for Entity Matching. In: *Proceedings of The Web Conference 2020*, pp. 2634-2640 (2020)
 40. Kim, Y.: Convolutional neural networks for sentence classification. In: *EMNLP*, pp. 1746-1751 (2014)