GNEM: A Generic One-to-Set Neural Entity Matching Framework

Runjin Chen
chenrunjin@sjtu.edu.cn
Shanghai Jiao Tong University

Yanyan Shen*
shenyy@sjtu.edu.cn
Shanghai Jiao Tong University

Dongxiang Zhang
zhangdongxiang@zju.edu.cn
Zhejiang University

ABSTRACT

Entity Matching is a classic research problem in any data analytics pipeline, aiming to identify records referring to the same real-world entity. It plays an important role in data cleansing and integration. Advanced entity matching techniques focus on extracting syntactic or semantic features from record pairs via complex neural architectures or pre-trained language models. However, the performances always suffer from noisy or missing attribute values in the records. We observe that comparing one record with several relevant records in a collective manner allows each pairwise matching decision to be made by borrowing valuable insights from other pairs, which is beneficial to the overall matching performance. In this paper, we propose a generic one-to-set neural framework named GNEM for entity matching. GNEM predicts matching labels between one record and a set of relevant records simultaneously. It constructs a record pair graph with weighted edges and adopts the graph neural network to propagate information among pairs. We further show that GNEM can be interpreted as an extension and generalization of the existing pairwise matching techniques. Extensive experiments on real-world data sets demonstrate that GNEM consistently outperforms the existing pairwise entity matching techniques and achieves up to 8.4% improvement on F1-Score compared with the state-of-the-art neural methods.

CCS CONCEPTS

• **Information systems → Data cleaning**; **Search results deduplication**; **Deduplication**; • **Computing methodologies → Artificial intelligence**.

KEYWORDS

Entity matching; Graph neural network; Deep learning

ACM Reference Format:
Runjin Chen, Yanyan Shen, and Dongxiang Zhang. 2020. GNEM: A Generic One-to-Set Neural Entity Matching Framework. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021 ,Ljubljana, Slovenia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3442381.3450119

1 INTRODUCTION

In the era of Big Data, high-quality data is required in many data-driven applications. Entity matching (EM), aiming to identify records referring to the same real-world entity over one or multiple data sources, is an essential task in data cleaning and integration. Early researches [4, 8] on entity matching mainly used string similarity measures such as Jaccard or TF-IDF to judge whether two records refer to the same entity. However, these methods assume that records referring to the same entity always share similar tokens and this assumption may not hold in practice. A line of researches [16, 17, 27] tried to understand the relationship between two records at semantic level. They developed a set of domain-specific rules and applied them to every pair of records. The pairs that satisfy all the rules are regarded as matched. However, the rule-based methods usually suffer from poor generalization ability.

In the past few years, machine learning algorithms has been widely applied in real-world applications and yields the state-of-the-art performance. Hence, recent efforts [7, 12, 15, 20, 30, 32] have been devoted to developing learning-based approaches to solve the EM problem. Magellan [15] created handcrafted features for records and applied classic machine learning techniques such as SVM and Decision Tree to classify the matching label of two records. DeepER [12], DeepMatcher [20] and MCA [30] proposed neural network models incorporating LSTM, Attention, Highway Network to learn latent semantic features for a pair of records automatically. BERT [7] and Auto-EM [32] adopted the pre-trained models to transfer collective knowledge from external data to get a better understanding of the semantics of records.

Motivation. EM neural methods have been shown to act as strong baselines [7, 12, 20, 30, 32]. They introduced elaborate neural network designs to extract semantic information from each record and model feature interactions within a record pair towards higher

Table 1: Motivating Examples for One-to-Set EM

(a) (r_1^a, r_1^b) **matched** $\wedge$ (r_1^b, r_2^b) **matched** $\Rightarrow$ (r_1^a, r_2^b) **matched**

No.	Name	Gender	City	Occupation	$(r_1^a, \cdot)$
r_1^a	John Smith	female	Seattle, Washington	–	–
r_1^b	J. Smith	female	Seattle, Washington	teacher	*matched*
r_2^b	J. Smith	–	Seattle, WA	teacher	*matched*

(b) (r_2^a, r_3^b) **unmatched** $\wedge$ (r_3^b, r_4^b) **matched** $\Rightarrow$ (r_2^a, r_4^b) **unmatched**

No.	Title	Artist	Genre	Tag	$(r_2^a, \cdot)$
r_2^a	My Love	Westlife	pop	Band, Heart Touching	–
r_3^b	My Love	Sia	Indie rock	OST, Heart Touching	*unmatched*
r_4^b	My Love	–	–	OST, Heart Touching	*unmatched*

*corresponding author

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450119

prediction accuracy. However, to the best of our knowledge, all the existing neural methods follow a pairwise EM framework that accepts a pair of records and predicts a binary matching label. The performance may suffer from noisy or insufficient information in the records. For instance, it is difficult to compare the pairs of (r_1^a, r_2^b) and (r_2^a, r_4^b) in Table 1 due to the missing values. The key inspiration of this paper comes from a recent work [28] which posited that displaying multiple images simultaneously plays an important role in making labelling judgement on image pairs by human. For instance, consider three images a, b, c showing the faces of the same person at young, middle and older ages, respectively. If all the three images are provided and compared, we may easily determine that they all refer to the same person. When only a and c are available, it is difficult for us to tell whether they point to the same person or not. This fact also holds true in the context of EM, where relevant records may provide insights on the matching result of a pair of records.

Motivating Example 1. Table 1a shows three records r_1^a, r_1^b, r_2^b referring to the same entity. Suppose we aim to determine whether (r_1^a, r_2^b) is a matched pair. The similarity between r_1^a and r_2^b is relatively low due to the missing attribute values in the records. Hence, given only r_1^a and r_2^b, it is difficult for a pairwise EM method to recognize that they are matched. Interestingly, when r_1^b is provided, we can easily tell that (r_1^a, r_1^b) is matched, and r_1^b, r_2^b are similar. This allows us to infer that (r_1^a, r_2^b) is also a matched pair.

Motivating Example 2. Consider three records r_2^a, r_3^b, r_4^b in Table 1b, where r_3^b, r_4^b refers to the same entity and r_2^a is different from both of them. Our task is to decide whether (r_2^a, r_4^b) is a matched pair. Since r_2^a and r_4^b share the same or similar values on Title and Tag attributes, it is difficult to conclude they are unmatched. By referring to r_3^b, we realize that (r_2^a, r_3^b) is unmatched, and r_4^b is more similar to r_3^b than r_2^a. This provides a clue that (r_2^a, r_4^b) is unmatched.

The above two examples illustrate the benefits of referring to relevant records during the EM process, especially for difficult pairs. To be more specific, in the Motivating Example 1, r_1^b plays a transition role to help us identify the truth matching pair (r_1^a, r_2^b), which reduces the chance of a false negative; in the Motivating Example 2, r_3^b is used to distinguish r_2^a from r_4^b, thus preventing a false alarm. We argue that the existing EM neural methods mainly focus on measuring syntactic or semantic similarity via pairwise records comparison. They fail to exploit valuable information from relevant records, and the performance is compromised.

Solution. In this paper, we develop a generic One-to-Set EM neural framework named **GNEM**. It prepares matching instances in the format of (r, V_r), where r is a record and V_r involves all the relevant records to be matched with r. Different from the conventional pairwise matching process that infers the matching labels between r and each record in V_r independently, GNEM is able to predict the matching labels for r against all the records in V_r simultaneously. To achieve this, we construct a *matching pair graph* $\mathcal{G}_r$ for (r, V_r), where each node represents a record pair $(r, r' \in V_r)$. We associate every two nodes in $\mathcal{G}_r$ with a weighted edge, and adopt a gated version of graph neural network to facilitate the interaction among pairs. By doing so, each pair is able to borrow valuable information from other pairs such as the similarity/dissimilarity signals when making its own matching decision. Further, we show that GNEM can be interpreted as either an extension or a generalization of the existing pairwise EM techniques. Extensive experiments on three EM tasks demonstrate the effectiveness of GNEM compared with the state-of-the-art pairwise EM methods.

Contributions. In summary, the major contributions of this paper are the following:

- **Interaction** We propose a generic One-to-Set neural framework named GNEM for entity matching. GNEM allows each record pair to interact with relevant records and exploit valuable information from other pairs during the decision of its matching label, which is beneficial to the overall matching performance.
- **Generality** We show that GNEM can be interpreted as an extension and generalization of the existing pairwise EM neural methods. It enhances the pairwise matching process by devising a record pair graph to facilitate the interaction among pairs for information propagation.
- **Effectiveness** We conduct extensive experiments on three public entity matching datasets. The results demonstrate the promise of one-to-set EM framework and GNEM achieves significant improvement in the matching accuracy, compared with eight EM methods. Our codes are publicly available at https://github.com/ChenRunjin/GNEM

The remainder of this paper is organized as follows. Section 2 presents the entity matching problem and the existing pairwise methods. Section 3 introduces our proposed one-to-set EM framework. Section 4 provides the experimental results. We review the related works in Section 5 and conclude this paper in Section 6.

2 PRELIMINARIES

2.1 Notations and Problem

We consider a set $\mathbf{E}$ of real-world *entities*, where each entity $e \in \mathbf{E}$ is associated with a set $\mathbf{A}$ of n *attributes*, i.e., $e.A_1, \cdots, e.A_n$. Let $\mathbf{R^a}$ and $\mathbf{R^b}$ denote two collections of *records* following the same schema $\mathbf{A}$, where each record $r \in \mathbf{R^a} \cup \mathbf{R^b}$ refers to an entity in $\mathbf{E}$. The goal of *entity matching* is to determine, given two records from $\mathbf{R^a}$ and $\mathbf{R^b}$ respectively, whether they refer to the same entity in $\mathbf{E}$.

Definition 1 (Entity Matching). *Let $\mathbf{E}$ be a domain of entities described by an attribute set $\mathbf{A}$ and $\mathbf{R^a}, \mathbf{R^b}$ be two sets of records in the format of $(x_1, \cdots, x_n)$. For any record $r \in \mathbf{R^a} \cup \mathbf{R^b}$, $r.x_i$ is the record value on attribute A_i, which can be a textual description, or empty. Given two records $r \in \mathbf{R^a}, r' \in \mathbf{R^b}$, the* Entity Matching *problem is to predict the probability that r, r' refers to the same real-world entity in $\mathbf{E}$, i.e., $Pr(y = \text{matched} \mid r, r')$.*

2.2 Pairwise EM Neural Methods

Typically, the entity matching problem is treated as a binary classification task, i.e., $y \in \{\text{matched}, \text{unmatched}\}$. Recent efforts have been devoted to developing *pairwise EM neural network models* [7, 12, 15, 20, 30, 32] towards higher classification accuracy. These models take as input a pair of records from $\mathbf{R^a}$ and $\mathbf{R^b}$ respectively and produce the probability that the two records refer to the same entity. The key idea is to learn a common feature space where records referring to different entities can distinguish with each

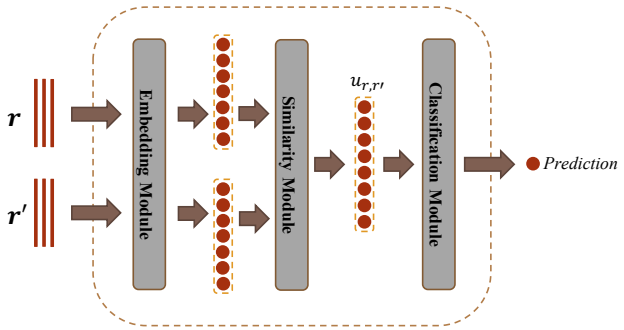

Figure 1: Pairwise EM neural methods.

other based a learned similarity function. As depicted in Figure 1, a pairwise EM neural network model generally consists of an *embedding module* to encode the textual description in each attribute, followed by a *pairwise similarity module* to compute a semantic similarity vector for the two input records, and finally a *classification module* to produce the matching probability based on the similarity vector. We use the collection of deep learning models SIF, RNN, Attention, and Hybrid proposed in [20] as the instances of pairwise EM neural methods. Consider an input pair of records $r = (x_1, \cdots, x_n) \in \mathbf{R^a}, r' = (x'_1, \cdots, x'_n) \in \mathbf{R^b}$.

The embedding module applies the *de-facto* word embeddings [5, 11] to transform each token in an attribute into a latent vector. Let $r.x_i$ be a sequence of tokens $(w_1, \cdots, w_k)$ on attribute A_i, which can be an empty string. The corresponding embedding vectors are denoted by $(\mathbf{w}_1, \cdots, \mathbf{w}_k)$.

The pairwise similarity module absorbs two sequences of embedding vectors of r, r', and computes a *similarity representation vector* $\mathbf{u}_{r,r'}$. For each attribute A_i, the first step is to aggregate the involved embedding vectors properly to derive the *attribute representation* of $r.x_i$, which is defined as follows:

$$r.\mathbf{x}_i = \phi(\mathbf{w}_1, \cdots, \mathbf{w}_k) \tag{1}$$

where ϕ is the aggregating function such as averaging or a neural network. The second step is to encode the sequences of attribute representations of r, r' and generate the similarity vector. Let $\mathbf{r} = (r.\mathbf{x}_i, \cdots, r.\mathbf{x}_n)$ denote the attribute representation sequence of r.

Some implementations such as SIF [20] and RNN [20] learn a *semantic feature vector* $\mathbf{f}_r$ solely based on $\mathbf{r}$ and different records are encoded separately. Some works such as Hybrid [20] and MCA [30] compute $\mathbf{f}_r, \mathbf{f}_{r'}$ by taking into account the correlations between $\mathbf{r}, \mathbf{r}'$. In either way, the similarity vector $\mathbf{u}_{r,r'}$ is then derived by fusing the feature vectors $\mathbf{f}_r, \mathbf{f}_{r'}$. Recently, since transformer architectures have proven to be effective and bring a massive performance boost on many NLP tasks, some works [7] adopted the pre-trained transformer architectures to learn the similarity vector of two records for EM. Instead of performing comparison between two records, the pre-trained transformer architectures simply concatenate the attribute representation sequences of r, r' and conduct one-pass scan to obtain the similarity vector. In a nutshell, the similarity vector $\mathbf{u}_{r,r'}$ can be computed in the following high-level way:

$$\mathbf{u}_{r,r'} = \varphi(\mathbf{r}, \mathbf{r}') \tag{2}$$

The classification module takes the similarity vector $\mathbf{u}_{r,r'}$ as input and outputs the matching probability between r, r' via fully connected layers or the Highway network [24].

As described before, the existing pairwise EM neural methods have fully exploited the latent features of each record and modeled the pairwise semantic relevance, reporting the state-of-the-art matching accuracy. The key limitation is that they fail to utilize the information of relevant records that are useful to eliminate false alarms or identify truth matched pairs, especially when the records involve noisy or missing attribute values. This inspires us to develop a generic framework that could enhance the pairwise matching process by taking multiple and relevant records into account simultaneously. We emphasize that our proposed framework is developed on top of the pairwise EM neural methods and hence is orthogonal to the existing approaches [7, 12, 15, 20, 30, 32], which will be discussed in Section 3.6.

3 ONE-TO-SET NEURAL EM FRAMEWORK

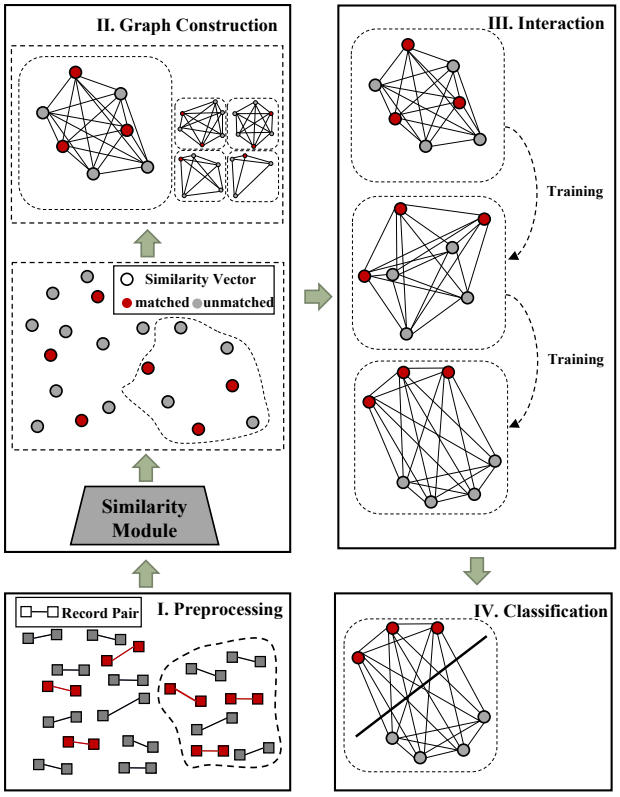

Figure 2: Framework of GNEM.

In this section, we resort to the one-to-set entity matching process and develop a generic EM neural framework named GNEM. As depicted in Figure 2, GNEM consists of four major components: (1) *Preprocessing* extracts relevant records to augment each record pair and prepares one-to-set matching instances; (2) *Graph construction* organizes the matching pairs in an instance into a graph structure

with weighted edges; (3) *Interaction* propagates matching information among nodes in the graph to boost the matching performance; (4) *Classification* produces the final matching probabilities. In what follows, we provide the details of each component in GNEM and then discuss the relation of GNEM to the existing pairwise EM neural methods.

3.1 Preprocessing

Given two collections of records $\mathbf{R^a}$ and $\mathbf{R^b}$, the goal of preprocessing is for each record in $\mathbf{R^a}$, to identify relevant records in $\mathbf{R^b}$ and vice versa. A record in $\mathbf{R^a}$ (resp. $\mathbf{R^b}$) with a set of relevant records in $\mathbf{R^b}$ (resp. $\mathbf{R^a}$) compose a one-to-set matching instance to be supplied to the remaining components.

Initially, we consider all the record pairs in $\mathbf{R^a} \times \mathbf{R^b}$, where $\times$ is the cartesian product over two sets. In order to identify relevant records, we resort to the *blocking mechanism* [21, 22, 31] which has been widely used to filter out irrelevant record pairs and improve the matching efficiency. The key idea is to divide records in $\mathbf{R^a} \cup \mathbf{R^b}$ into disjoint blocks and only the records in the same block are likely to be matched. After blocking, we are able to eliminate matching irrelevant records and obtain a candidate set $\mathbf{C} \subseteq \mathbf{R^a} \times \mathbf{R^b}$. Note that the blocking mechanism is assumed to have no false negatives but the candidate set may include relevant yet unmatched pairs, which are also useful according to examples in Table 1. There exist a number of blocking methods [21, 22, 31] in the literature. Our framework is flexible to apply any one of them.

To derive the one-to-set matching instances, for each record r in $\mathbf{R^a} \cup \mathbf{R^b}$, we retrieve all the pairs in $\mathbf{C}$ involving r, i.e., $\{(r_i, r_j) \in \mathbf{C} \mid r_i = r \vee r_j = r\}$. We now recognize the set V_r of records that are relevant to r as follows:

$$V_r = \{r' \mid (r, r') \in \mathbf{C} \vee (r', r) \in \mathbf{C}\} \tag{3}$$

Hence, we obtain (r, V_r) as a *one-to-set matching instance*. Different from the prior works that compute the matching probabilities between r and any record in V_r independently, we propose one-to-set matching process that infers matching results for r against all the records in V_r in a collective manner.

Note that each record pair (r, r') in $\mathbf{C}$ is included in two matching instances. The matching results can be different by referring to different relevant record sets, i.e., V_r and $V_{r'}$. The two matching results will be fused to produce the final matching score.

3.2 Graph Construction

After preprocessing, the input to this component is the set $\mathbf{D}$ of one-to-set matching instances, i.e., $\mathbf{D} = \{(r, V_r)\}_{r \in \mathbf{R^a} \cup \mathbf{R^b}}$. Our ultimate goal is to compute the matching probabilities between r and all the records in V_r simultaneously. The rationale behind is that the matching information encoded in a pair $(r, r' \in V_r)$ might be useful to infer the matching results between r and other records in V_r. In order to propagate pair information properly, we propose to construct a *matching pair* graph $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E}_r)$ for $(r, V_r) \in \mathbf{D}$. Each node in $\mathcal{V}_r$ represents a pair $(r, r' \in V_r)$ and hence the total number of nodes in $\mathcal{V}_r$ is $|V_r|$. By doing so, we are able to transform the one-to-set matching problem into a set of node classification problems, i.e., predicting the binary matching labels for all the nodes in $\mathcal{V}_r$. There are still two problems to be addressed: (1) *how to obtain the initial node representations in the graph?* (2) *how to establish edges among nodes?*

(1) Initializing node representations for $\mathcal{V}_r$. Consider a node $v = (r, r')$ where $r' \in V_r$. Ideally, the initial representation of v should encode the features of r as well as the feature interactions between r and r'. This inspires us to leverage the off-the-shelf embedding module and similarity module developed by pairwise EM neural methods (in Section 2.2), and compute the similarity vector for the pair as the initial node representation. Following Equation 2, we use $\mathbf{u}_{r,r'}$ as the initial representation for node v. We denote by $X_r \in \mathbb{R}^{|V_r| \times d}$ all the initial node representations for $\mathcal{V}_r$, where d is the dimension of the similarity vectors $\mathbf{u}$.

(2) Establishing edges among nodes in $\mathcal{G}_r$. In our one-to-set EM setting, we allow each record pair to refer to other pairs (involving relevant records) directly during the matching process. We thus establish an edge for every two nodes in $\mathcal{V}_r$ (including self-loops) and the graph $\mathcal{G}_r$ is a complete graph. Note that there are essentially two kinds of nodes in the graph, representing matched pairs and unmatched pairs, respectively. While the complete graph $\mathcal{G}_r$ allows direct information exchange between any two pairs, treating all the edges equally may propagate noisy information among nodes with different matching results. Therefore, we propose to augment each edge with a weight to distinguish different relationships among pairs. We shall next describe how to assign weights to the edges.

Recall the Motivating Example 1. Both r_1^b and r_2^b are relevant to r_1^a, and (r_1^a, r_1^b) is clearly matched. Regarding the high similarity between r_1^b and r_2^b, we may infer that (r_1^a, r_2^b) is matched. In other words, the similarity between (r_1^a, r_1^b) and (r_1^a, r_2^b) has the potential to pull the representation of the difficult-to-match pair (r_1^a, r_2^b) and that of the easy-to-match pair (r_1^a, r_2^b) closer to each other in the latent feature space so that the same matching label can be inferred for both pairs. Likewise, in Motivating Example 2, r_3^b and r_4^b are alike and (r_2^a, r_3^b) is apparently unmatched. The similarity between (r_2^a, r_3^b) and (r_2^a, r_4^b) motivates us to propagate the unmatching signal from (r_2^a, r_3^b) to (r_2^a, r_4^b) during its label classification. Following the above intuitions, we encourage the edges between pairs sharing the same matching label to have relatively larger weights than those between pairs with different labels. This facilitates the information propagation from easy-to-classify nodes to similar yet difficult-to-classify pairs with the same label in the next interaction stage.

Consider two nodes $v_i = (r, r_i)$ and $v_j = (r, r_j)$ in $\mathcal{V}_r$. Since we have no knowledge about their matching results, we propose to compute the edge weight based on the semantic feature vectors $\mathbf{f}_{r_i}$ and $\mathbf{f}_{r_j}$ for r_i and r_j respectively, which are learned by the pairwise similarity module as described in Section 2.2. We measure the difference between r_i, r_j by $Abs(\mathbf{f}_{r_i} - \mathbf{f}_{r_j})$, and supply it to a simple neural network. Formally, let $\mathcal{A}_r^{ij}$ denote the edge weight between v_i, v_j in graph $\mathcal{G}_r$. We have:

$$\mathcal{A}_r^{ij} = \mathcal{F}(Abs(\mathbf{f}_{r_i} - \mathbf{f}_{r_j})) \tag{4}$$

where $\mathcal{F}$ is a stack of L fully-connected layers (L is set to 1 or 2).

Note that the transformer-based pairwise similarity module [7] produces the similarity vector for r_i, r_j directly and the individual feature vector of each record is unavailable. To address this problem,

we concatenate the similarity vectors $\mathbf{u}_{r,r_i}, \mathbf{u}_{r,r_j}$, and supply the concatenated vector to $\mathcal{F}$. That is,

$$\mathcal{A}_r^{ij} = \mathcal{F}(\mathbf{u}_{r,r_i} \oplus \mathbf{u}_{r,r_j}) \tag{5}$$

Note that i could be equal to j to assign weights to self-loops.

Intuitively, if r_i, r_j share high similarity and these exists $r_k \in V_r$ that is different from r_i, we expect the neural network $\mathcal{F}$ could assign higher edge weights to (v_i, v_j) than (v_i, v_k) (i.e., $v_k = (r, r_k)$).

3.3 Interaction via Graph Neural Network

Given a graph $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{E})$ with weighted adjacency matrix $\mathcal{A}_r$, we aim to predict the binary matching labels for all the nodes in the graph. The one-to-set matching setting allows each pair to provide valuable insight to other pairs, thus benefiting the overall matching performance. To this end, we introduce the interaction component in GNEM to facilitate the propagation of information among nodes in the graph, which is achieved by a graph convolution network.

We first perform the column-wise softmax operation over $\mathcal{A}_r$ and obtain the normalized adjacency matrix $\tilde{\mathcal{A}}_r$. We adopt a gated version of GCN, where the gating mechanism is applied on each convolution layer to prevent the over-smoothing problem [9]. In the first layer, we use the initial node representations X_r as the input, i.e., $H^{(0)} = X_r$. At a high level, each gated layer learns to filter the current node representations and augment each node with the features from its neighbors. The new node representations are computed and then supplied to the next layer. Formally, the l-th graph convolution layer is defined as:

$$\begin{aligned} z^{(l)} &= \sigma_1\big(W_{z,s}^{(l)} H^{(l-1)} + W_{z,n}^{(l)} \tilde{\mathcal{A}}_r H^{(l-1)}\big) \\ \tilde{H}^{(l)} &= \sigma_2\big(W_{o,s}^{(l)} H^{(l-1)} + W_{o,n}^{(l)} \tilde{\mathcal{A}}_r H^{(l-1)}\big) \\ H^{(l)} &= z^{(l)} \odot \tilde{H}^{(l)} + (1 - z^{(l)}) \odot H^{(l-1)} \end{aligned} \tag{6}$$

where $H^{(l)}$ denotes the node representations produced by the l-th layer, $\sigma_1(\cdot)$ is the sigmoid function, $\sigma_2(\cdot)$ is the *tanh* function, and $\odot$ means the Hadamard point-wise multiplication operator. The W terms are weighted matrices to be learned.

In fact, the gated version of GCN imitates the gating mechanism in Highway Network [24] which regulates the information the updated messages to be propagated. In our case, $z^{(l)}$ acts as the gate to regulate the information from neighboring pairs at the l-th layer. By doing so, each node (representing a pair of records) can retain its own representation as well as borrow highly valuable information from its neighbors. After L' layers, the final node representations $H^{(L')}$ is the output of the gated GCN. Let $\tilde{X}_r = H^{(L')}$.

Since the graph $\mathcal{G}_r$ is a complete graph, a single gated convolution layer is sufficient for each node to refer to all the relevant pairs involved in the one-to-set instance. We observe that more gated convolution layers can further boost the matching accuracy for the graphs of larger sizes. The effects of the number of layers will be discussed in the experiments.

3.4 Classification

The input to this component is the learned representations $\tilde{X}_r$ for all the nodes in the graph $\mathcal{G}_r$, and we aim to infer a binary matching label for each node. That is, $Pr(y_{r,r'} \mid \tilde{X}_{r,r'})$, for $r \in V_r$. Note that $\tilde{X}_{r,r'}$ has already encoded the information of the record pair (r, r') and that from the context pairs. Some pairwise EM methods [20, 30] use Highway Network to produce the classification results. To control the model complexity, we supply $\tilde{X}_{r,r'}$ to a single fully connected layer followed by a standard softmax classifier. Let $s_{r,r'}$ denote the vector generated by the fully connected layer given $\tilde{X}_{r,r'}$. The matching probability is computed as follows:

$$Pr(y_{r,r'} \mid \tilde{X}_{r,r'}) = \text{softmax}(W s_{r,r'} + b) \tag{7}$$

As mentioned before, each record pair (r, r') will appear in two graphs $\mathcal{G}_r, \mathcal{G}_{r'}$ in which the context pairs can be different. We fuse the two matching probabilities as follows:

$$Pr(\hat{y} \mid r, r') = \text{Average}\big(Pr(y_{r,r'} \mid \tilde{X}_{r,r'}), Pr(y_{r',r} \mid \tilde{X}_{r',r})\big) \tag{8}$$

Further, we compare $Pr(\hat{y} \mid r, r')$ with a threshold θ. If the fused probability $Pr(\hat{y} = \text{matched} \mid r, r')$ is larger than θ, we report the two records as matched. Otherwise, they are regarded as unmatched. The value of θ can be pre-defined or tuned as a hyperparameter through cross-validation.

3.5 Training and Optimization

We adopt the *binary cross-entropy* as the loss function. Let $\mathbf{T}$ be the set of training record pairs with binary matching labels. We have:

$$L_{loss} = -\sum_{(r,r') \in \mathbf{T}} [y_{r,r'} \log \hat{y}_{r,r'} + (1 - y_{r,r'}) \log(1 - \hat{y}_{r,r'})] \tag{9}$$

where $y_{r,r'} \in \{0, 1\}$ denotes the ground-truth matching label and $\hat{y}_{r,r'}$ is the predicted matching probability in Equation (8). All the parameters are learned by the Adam optimizer [13].

3.6 Relation to Pairwise EM Neural Methods

While our proposed GNEM is a one-to-set neural framework for entity matching, it can be viewed as an extension and generalization of the existing pairwise EM methods. We provide two perspectives to relate GNEM with the pairwise techniques.

First, GNEM uses the similarity vectors $\mathbf{u}$ produced by the pairwise similarity module in various pairwise EM methods as the initial representations for the record pairs to be matched, i.e., $\{X_r\}$. The interaction component allows each pair to borrow valuable insights from relevant pairs. Consider a pair (r, r'). By removing the interaction via GNN, GNEM is simplified and only able to predict the matching label for (r, r') based on $\mathbf{u}_{r,r'}$. Hence, GNEM is an extension of the existing pairwise EM neural methods.

Second, the interaction between record pairs in a graph $\mathcal{G}_r$ is controlled by the edge weights defined in the adjacency matrix $\mathcal{A}_r$. Consider the extreme case that the weights of self-loops are pushed to $+\infty$. The normalized adjacency matrix $\tilde{\mathcal{A}}_r$ will become an identity matrix and we have $\tilde{\mathcal{A}}_r H^{(l-1)} = H^{(l-1)}$. The Equation (6) can be transformed into the following:

$$\begin{aligned} z^{(l)} &= \sigma_1\big(W_z^{(l)} H^{(l-1)}\big) \\ \tilde{H}^{(l)} &= \sigma_2\big(W_o^{(l)} H^{(l-1)}\big) \\ H^{(l)} &= z^{(l)} \odot \tilde{H}^{(l)} + (1 - z^{(l)}) \odot H^{(l-1)} \end{aligned} \tag{10}$$

where $W_z^{(l)} = W_{z,s}^{(l)} + W_{z,n}^{(l)}$ and $W_o^{(l)} = W_{o,s}^{(l)} + W_{o,n}^{(l)}$. The gated graph convolution layer mimics a layer in the Highway Network [24]. Hence, GNEM is a generalization of the pairwise EM techniques.

Table 2: Statistics of the datasets.

Dataset	Type	Domain	Pairs	Matched Pairs	Avg. graph size (Training)	Avg. graph size (Test)
Abt-Buy	Textual	Product	9575	1028	5.95	11.76
Amazon-Google	Structured	Software	11460	1167	4.72	10.33
Walmart-Amazon	Structured	Electronics	10242	962	2.40	5.32

4 EXPERIMENTS

In this section, we conduct experiments on three public EM datasets. We compare GNEM with 8 existing EM neural methods. The goals of the experiments is to answer the following three research questions:

- **RQ1** Can our proposed GNEM boost matching performance compared with the pairwise EM techniques?
- **RQ2** How does the number of the gated graph convolution layers influence the performance of GNEM?
- **RQ3** How does GNEM perform over different sizes of the constructed record pair graphs?

4.1 Set Up

We implemented GNEM with PyTorch, a popular open-source deep learning framework, and executed on Nvidia GeForce RTX 2080 Ti GPU (12GB Memory) with Intel(R) Xeon(R) Silver 4110 CPU and 128GB Memory.

To make a fair comparison with each of the existing pairwise EM neural methods, we adopted the same embedding module and optimizer of the compared model to implement GNEM . Specifically, for SIF, RNN, Attention and Hybrid, we choose *fastText* [5] as embedding module, since character-level embeddings are more robust to infrequent words in datasets. While for BERT, XLNet, RoBERTa and DistilBERT, we keep using their corresponding embedding modules. All the comparison neural methods adopt *Adam* as the optimizer. We tuned the hyper-parameters for all the models according to their results on the validation set.

Since entity matching is essentially a binary classification problem, we use *F1-score* to evaluate the performance of all the methods. We ran each experiment five times and reported the average results.

4.2 Datasets

Entity Matching datasets can be divided into two types, *structured* and *textual* types. Attributes in a structured dataset are supposed to be short and atomic, such as name, sex, age, city, etc. Attributes in a textual dataset are associated with raw text entries. In general, EM on textual datasets is more difficult because we need to extract discriminative information from complex raw text.

The authors of DeepMatcher [20] have demonstrated the effectiveness of ML techniques in entity matching using datasets from [1] . Specifically, DeepMatcher achieves 100% F1-score on Fodors-Zagats and 98.4% F1-score on DBLP-ACM, etc. Hence, we focus our experiments on more challenging datasets : *Abt-Buy*, *Amazon-Google*, and *Walmart-Amazon* . The three datasets cover different types and application domains. All the datasets are published after blocking so that we can obtain an integrated labeled candidate set $\mathbf{C}$ directly. We divide each candidate set into three parts, training, validation and test, following the split ratio of 3:1:1. Training set is used to optimize model parameters and validation set is used to tune hyperparameters. The performances of different methods are evaluated using test set.

Table 2 provides the statistical details of all the datasets. The first two columns show the types and application domains of the datasets, respectively. *Pairs* reports the number of record pairs in the candidate sets after blocking. Due to the limited numbers of examples in the validation and test sets, the number of relevant records involved in each one-to-set instance is relatively small. To include more possibly beneficial records in the constructed graphs, we consider the whole candidate set to construct the one-to-set matching instances in the validation and test sets. The record pairs used in the training sets are kept as the same over all the methods. *Avg. graph size (Train)* and *Avg. graph size (Test)* in Table 2 provide the average matching pair graph sizes (i.e., the number of nodes) in the training and test sets, respectively.

4.3 Baselines

In our experiments, we consider eight state-of-the-art EM neural methods as baselines.

- **SIF** [20] The original DeepMatcher paper explored DL solution space and proposed four pairwise EM models. SIF is one of them. SIF simply applies an aggregate function over the embeddings of all the tokens in an input sequence to obtain the semantic features for each record. It then uses the absolute value of difference between two semantic features to generate the similarity vector of a pair.
- **RNN** [20] RNN is another pairwise model proposed by DeepMatcher. It takes the order of words in input sequences into account and uses a bidirectional RNN to summarize information contained in records. It also computes the absolute value of difference between two summarized features to generate the similarity vector for classification.
- **Attention** [20] Attention adopts decomposable attention to encode all the elements in a record pair into two semantic features. It uses the concatenation of two semantic features as the similarity vector.
- **Hybrid** [20] Hybrid is the most effective models in DeepMatcher. It merges RNN and Attention model together, incorporating both bidirectional RNN and decomposable attention to implement the information aggregating process. The concatenation of two features and the absolute value of difference between two features is used as the final similarity vector.
- **BERT** [7, 11] BERT is a famous pre-trained transformer-based language model which achieves marvelous performance in various NLP tasks. A Recent work [7] fine-tuned

[1] https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

Table 3: Performance comparison results.

		SIF	RNN	Attention	Hybrid	BERT	XLNet	DistilBERT	RoBERTa
Abt-Buy	origin	35.1	39.4	56.8	62.8	85.9	86.8	83.3	90.9
	GNEM (w/o interaction)	29.7	41.3	55.5	65.8	85.4	87.8	81.2	91.3
	GNEM	**44.2**	**45.5**	**61.5**	**71.9**	**87.7**	**88.7**	**83.6**	**93.0**
Amazon-Google	origin	**60.6**	59.9	61.1	69.3	71.3	71.6	69.4	70.4
	GNEM (w/o interaction)	48.5	58.5	62.3	67.6	72.5	75.4	72.0	72.5
	GNEM	59.6	**65.3**	**64.1**	**69.4**	**74.7**	**77.6**	**73.0**	**76.0**
Walmart-Amazon	origin	65.1	67.6	50.0	66.9	83.9	78.2	82.3	84.9
	GNEM (w/o interaction)	65.9	**69.9**	54.2	63.7	82.6	**82.4**	79.8	85.9
	GNEM	**66.7**	69.3	**58.7**	**68.7**	**86.7**	81.6	**85.0**	**86.5**

pre-trained BERT on entity matching task and prove it is effective to solve this problem.

- **XLNet** [7, 29] XLNet is another pre-trained transformer-based language model which addresses the limitations of BERT. XLNet is a permutation language model based on the architecture of a classical autoregressive language model. XLNet also incorporates the idea of Transformer-XL to learn dependencies beyond a fixed window length without disrupting temporal coherence. Hence, XLNet outperforms BERT and achieves the state-of-the-art results on many NLP tasks. Jacob Devlin et al. [7] analyzed the effectiveness of XLNet on entity matching.
- **RoBERTa** [7, 18] The authors of RoBERTa proposed several strategies to train BERT, and they released a new pre-trained BERT with properly tuned hyper-parameters and more training data.
- **DistilBERT** [7, 23] DistilBERT is a smaller yet effective language model. The core idea of DistilBERT is to train a simpler model which distills knowledge from BERT. Jacob Devlin et al. [7] have shown its superior performance on the entity matching task.

4.4 Comparison with Baselines (RQ1)

We report the *F1-score* results of all the comparison methods on *Abt-Buy*, *Amazon-Google* and *Walmart-Amazon* in Table 3. **origin** shows the performances of the original pairwise EM neural models. The results of *SIF*, *RNN*, *Attention* and *Hybrid* are collected from [20] directly, while the results of *BERT, XLNet, RoBERTa* and *DistilBERT* are obtained by running the open-source code of [7] over the three datasets. **GNEM** shows the results of our framework when applying the corresponding pairwise similarity module to compute initial node representations. We use a single-layer graph neural network for all the implementations of GNEM in Table 3. Further, to validate the effectiveness of the interaction module in GNEM, we report the results of **GNEM (w/o interaction)**, which abandons the interaction between nodes by removing all edges in graph except the self-loops. The GNEM (w/o interaction) follows the same design of the complete GNEM in terms of the parameters and structure, but force each node to focus on itself. The results of GNEM (w/o interaction) differ from the original pairwise models since they have different classification modules and batch samples (i.e., GNEM (w/o interaction) requires all relevant record pairs to be gathered within a batch). The performance gap between the origin and GNEM (w/o interaction) of SIF is because SIF only possesses few parameters in the similarity module, and its classification module affects the final performance to a larger extent.

Overall, from the results in Table 3, we can see that our framework consistently outperforms the existing pairwise EM methods in most cases over the three datasets. Specifically, GNEM achieves up to 8.4% improvement in F1-Score. GNEM can also surpass GNEM (w/o interaction), which indicates our framework benefits from the interaction between different nodes in the matching pair graph.

We observe that the textual dataset (Abt-Buy) benefits the most from our one-to-set EM setting, with an average **4.39 (8.9%)** improvement in F1-score compared with the original pairwise models and **4.76 (11.0%)** improvement compared with GNEM (w/o interaction). This is because Abt-Buy is in face of the severe attribute missing problem and it is hard for pairwise EM methods to learn informative and discriminative features from long sentences in difficult pairs where a large number of textual attributes are missing. In contrast, GNEM leverages the information of relevant records to boost the matching accuracy.

4.5 Effects of Gated Graph Convolution Layer (RQ2)

To investigate the effects of the number of gated graph convolution layers in GNEM, we set the layer number to be 1, 2 and 3 respectively and the results are provided in Figure 3. We can observe from the performance curves that a single layer is usually sufficient for record pairs to interact with their neighbors since the constructed graphs are complete. Increasing the number of GCN layers will bring in the over-smoothing problem, which may even cause performance degradation. On the Abt-Buy dataset, more layers achieve marginal performance improvement than single-layer. This is because the average graph size on Abt-Buy is relatively large and its records involve text blobs with possible complex semantics to be captured. Overall, we recommend to use a single-layer gated GCN

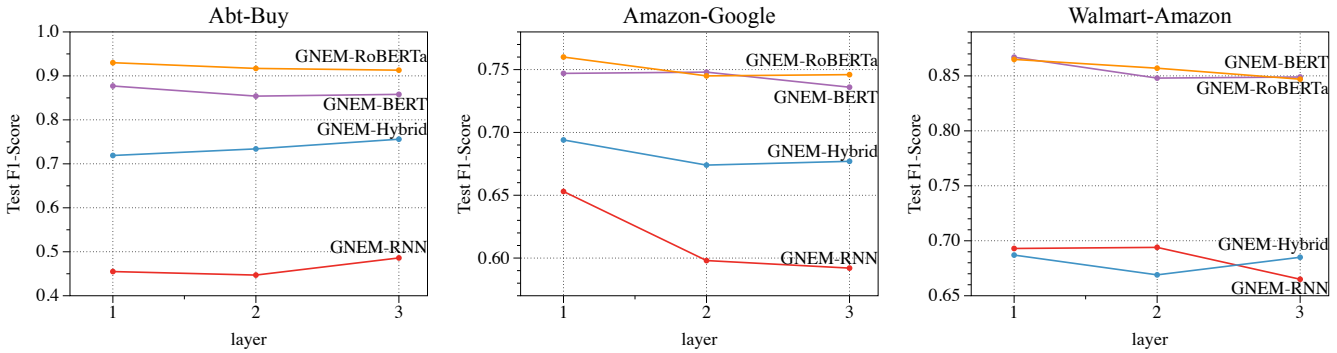

Figure 3: Effects of the number of gated graph convolution layers.

in GNEM for the trade-off between performance and parameter efficiency.

4.6 Effects of Graph Size (RQ3)

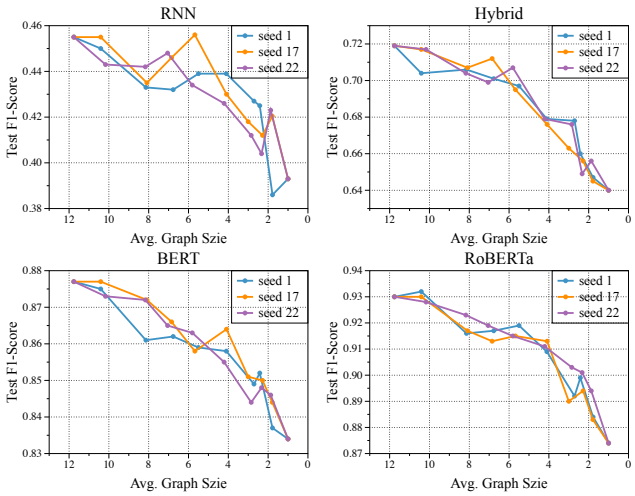

Figure 4: Effects of different graph sizes (Abt-Buy).

To further understand our framework, we randomly split the node set of $\mathcal{G}_r$ into several parts to decrease the average graph size during the test stage. We choose three different seeds to partition all the graphs in different ways and report the respective results in Figure 4. A smaller graph size means there are few relevant records to be concerned during each matching decision in the graph. When the graph size becomes one, GNEM is essentially transformed into GNEM (w/o interaction). Figure 4 shows the results on Abt-Buy as we observe similar trends on the other two datasets. We can see that the matching performance decreases as the average graph size becomes smaller, which follows our intuition. In general, the matching performance benefits from referring to more relevant records.

5 RELATED WORK

Entity Matching. Entity matching has attracted great attention from researchers in recent years. A number of different approaches have been proposed to solve the EM problem. Generally, the existing EM approaches can be divided into four categories. *Token-based methods* [4, 8] are popular in early EM researches. They compare word frequencies appearing in different records to determine the similarity between records. But these methods are not able to capture semantic-level information encoded in the records. *Rule-based methods* [1, 16, 17, 27] develop domain-specific rules and apply them on each record pair. Although rule-based methods are interpretable in most cases, they usually have poor generalization ability and require heavy involvement of domain experts. *Crowdsourcing-based methods* [25, 26, 28] transform record pairs in a candidate set into different queries to be posted on crowdsourcing platforms. The queries will be allocated to a large number of workers and the matching labels are derived based on the collected answers from the workers. Although crowdsourcing-based methods have proposed different strategies to improved the efficiency, the overall labeling process is still time-consuming and suffers from the scalability issue. *Learning-based methods* [7, 12, 15, 20, 30, 32] design different model structures incorporating elaborate machine learning techniques to encode semantic features for each record and treat the EM problem as a binary classification task. Existing methods such as DeepER [12], DeepMatcher[20], and MCA [30] adopted neural networks and have achieved outstanding performance on entity matching. Our proposed GNEM is a one-to-set EM framework, which can be interpreted as the extension and generalization of the existing pairwise learning-based methods.

Graph Neural Network. Inspired by the great success of CNN in computer vision domains, a large number of models [2, 3, 6, 10, 14, 19] were proposed to conduct analogous convolution operation on graph-structured data. Graph Convolutional Network (GCN) [14] defines graph convolution based on nodes' spatial relations, which propagates node information along edges in graph, and each node is updated with its neighbours' representations. In this paper, we adopt a gated-version of GCN to allow each pair to remember its own pairwise features and address the over-smoothing issue. But we emphasize that GNEM is flexible to incorporate other GNNs in the interaction component.

6 CONCLUSION

In this paper, we developed a generic one-to-set neural framework named GNEM for entity matching. GNEM compares one record

against a set of relevant records to derive binary matching labels simultaneously. For each one-to-set matching instance, GNEM constructs a graph for all the involved record pairs, where each node represents a matching pair and every two nodes are connected via an edge whose weight is determined by the similarity between pairs. GNEM then applies a gated version of graph neural network to propagate valuable information among pairs, and finally performs node classifications to predict matching labels for all the pairs. GNEM can be viewed as an extension and generalization of the existing pairwise EM techniques. We conducted experiments to demonstrate the effectiveness of GNEM in the matching performance, compared with the state-of-the-art EM neural methods.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their insightful reviews. This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831604) and NSFC (No.61602297). Dongxiang Zhang is partially support by the MoE grant (No. T1251RES1913) in Singapore.

REFERENCES

[1] Hiba Abu Ahmad and Hongzhi Wang. 2018. An effective weighted rule-based method for entity resolution. *Distributed and Parallel Databases* 36, 3 (2018), 593–612.
[2] James Atwood and Don Towsley. 2016. Diffusion-Convolutional Neural Networks. (2016), 1993–2001.
[3] Davide Bacciu, Federico Errica, and Alessio Micheli. 2018. Contextual Graph Markov Model: A Deep and Generative Approach to Graph Processing. *international conference on machine learning* 1 (2018), 294–303.
[4] Roberto J Bayardo, Yiming Ma, and Ramakrishnan Srikant. 2007. Scaling up all pairs similarity search. *Proceedings of the 16th international conference on World Wide Web* (2007), 131–140.
[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5, 1 (2017), 135–146.
[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral Networks and Locally Connected Networks on Graphs. *international conference on learning representations* (2014).
[7] Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures - a step forward in data integration. *International Conference on Extending Database Technology* (2020).
[8] Surajit Chaudhuri, Venkatesh Ganti, and Raghav Kaushik. 2006. A Primitive Operator for Similarity Joins in Data Cleaning. *22nd International Conference on Data Engineering (ICDE'06)* (2006), 5–5.
[9] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and Relieving the Over-smoothing Problem for Graph Neural Networks from the Topological View. *Proceedings of the VLDB Endowment* (2020).
[10] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. (2016), 3844–3852.
[11] Jacob Devlin, Mingwei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *north american chapter of the association for computational linguistics* (2019), 4171–4186.
[12] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *very large data bases* 11, 11 (2018), 1454–1467.
[13] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *international conference on learning representations* (2015).
[14] Thomas Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. (2017).
[15] Pradap Konda, Sanjib Das, G C Paul Suganthan, Anhai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F Naughton, et al. 2016. Magellan: toward building entity matching management systems. *Proceedings of the VLDB Endowment* 9, 12 (2016), 1197–1208.
[16] Lingli Li, Jianzhong Li, and Hong Gao. 2015. Rule-Based Method for Entity Resolution. *IEEE Transactions on Knowledge and Data Engineering* 27, 1 (2015), 250–263.
[17] Lingli Li, Jianzhong Li, Hongzhi Wang, and Hong Gao. 2011. Context-based entity description rule for entity resolution. *Proceedings of the 20th ACM international conference on Information and knowledge management* (2011), 1725–1730.
[18] 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: Computation and Language* (2019).
[19] Alessio Micheli. 2009. Neural Network for Graphs: A Contextual Constructive Approach. *IEEE Transactions on Neural Networks* 20, 3 (2009), 498–511.
[20] 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. (2018), 19–34.
[21] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, and Wolfgang Nejdl. 2013. A Blocking Framework for Entity Resolution in Highly Heterogeneous Information Spaces. *IEEE Transactions on Knowledge and Data Engineering* 25, 12 (2013), 2665–2682.
[22] George Papadakis, Jonathan Svirsky, Avigdor Gal, and Themis Palpanas. 2016. Comparative analysis of approximate blocking techniques for entity resolution. 9, 9 (2016), 684–695.
[23] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv: Computation and Language* (2019).
[24] Rupesh Kumar Srivastava, Greff Klaus, and Schmidhuber. Jürgen. 2015. Highway networks. *International Conference on Machine Learning* (2015).
[25] Vasilis Verroios, Hector Garciamolina, and Yannis Papakonstantinou. 2017. Waldo: An Adaptive Human Interface for Crowd Entity Resolution. *international conference on management of data* (2017), 1133–1148.
[26] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. CrowdER: crowdsourcing entity resolution. *very large data bases* 5, 11 (2012), 1483–1494.
[27] Steven Euijong Whang and Hector Garciamolina. 2010. Entity resolution with evolving rules. 3, 1 (2010), 1326–1337.
[28] Steven Euijong Whang, Peter Lofgren, and Hector Garciamolina. 2013. Question selection for crowd entity resolution. *Proceedings of the VLDB Endowment* 6, 6 (2013), 349–360.
[29] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Advances in neural information processing systems* (2019), 5753–5763.
[30] Dongxiang Zhang, Yuyang Nie, Sai Wu, Yanyan Shen, and Kian-Lee Tan. 2020. Multi-Context Attention for Entity Matching. *Proceedings of The Web Conference 2020* (2020). https://doi.org/10.1145/3366423.3380017
[31] Wei Zhang, Hao Wei, Bunyamin Sisman, Xin Luna Dong, Christos Faloutsos, and David C Page. 2020. AutoBlock: A Hands-off Blocking Framework for Entity Matching. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020), 744–752.
[32] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. (2019), 2413–2424.