GraphER: Token-Centric Entity Resolution with Graph Convolutional Neural Networks*

Bing Li,[1] **Wei Wang,**[1,2] **Yifang Sun,**[1] **Linhan Zhang,**[1] **Muhammad Asif Ali,**[1] **Yi Wang**[2]
[1]School of Computer Science and Engineering, University of New South Wales, Australia
{bing.li, weiw, yifang.sun, muhammadasif.ali}@unsw.edu.au, linhan.zhang@student.unsw.edu.au
[2]Dongguan University of Technology, China
wangyi@dgut.edu.cn

Abstract

Entity resolution (ER) aims to identify entity records that refer to the same real-world entity, which is a critical problem in data cleaning and integration. Most of the existing models are attribute-centric, that is, matching entity pairs by comparing similarities of pre-aligned attributes, which require the schemas of records to be identical and are too coarse-grained to capture subtle key information within a single attribute. In this paper, we propose a novel graph-based ER model GraphER. Our model is token-centric: the final matching results are generated by directly aggregating token-level comparison features, in which both the semantic and structural information has been softly embedded into token embeddings by training an Entity Record Graph Convolutional Network (ER-GCN). To the best of our knowledge, our work is the first effort to do token-centric entity resolution with the help of GCN in entity resolution task. Extensive experiments on two real-world datasets demonstrate that our model stably outperforms state-of-the-art models.

Introduction

Entity resolution (ER) (*a.k.a.*, entity matching, record linkage, and duplicate record detection) aims at identifying entity records that refer to the same real-world entity from different data sources. Table 1 shows an example of ER task. In Table 1, there are three records *r1*, *r2* and *r3* that are derived from *Amazon* and *Google* dataset, respectively. An ER system needs to find *r1* and *r2* refer to a same real-world entity, while *r3* does not.

Entity resolution is a fundamental problem in data cleaning and data integration (Dong and Srivastava 2013). There are numerous applications of entity resolution such as knowledge graph construction (Chen et al. 2015), e-commerce (Gokhale et al. 2014), and data warehouses (Bhattacharya and Getoor 2007). The importance of entity resolution has led to a substantial amount of research over the past few decades. Currently machine learning (ML) based solution has become the *de-facto* standard for ER task.

Given two entity records, typical ML-based ER approaches first compare pre-aligned attributes on top of hand-crafted (Konda et al. 2016) or deep neural (Ebraheem et al. 2018; Mudgal et al. 2018) features, then learn a classifier (*e.g.*, logistic regression, multilayer perceptron, *etc.*) to aggregate comparison results of all attributes to make final ER decisions. Recently, deep learning (DL) models have been used to learn attribute representations, including RNN and LSTM in conjunction with attention mechanism (Ebraheem et al. 2018; Mudgal et al. 2018). These deep learning models can capture semantic and syntactic information to better represent semantic similarity, especially for textual attributes.

Table 1: An example of ER task.

Google

Record	TITLE	MANUFACTURER	PRICE
r1	microsoft powerpoint 2004 mac apple	-	228.95

Amazon

Record	DESCRIPTION	MANUFACTURER	PRICE
r2	powerpoint 2004 mac by microsoft	microsoft	229.99
r3	powerpoint 2004 upgrade mac	microsoft	109.99

Despite DL-based ER solutions have been proven successful in improving the general performance of ER task (Mudgal et al. 2018), their attribute-centric paradigm share some common drawbacks, which hinder them from further performance improvement, as listed below:

i) *Attribute representation causes semantic sparsity and information dilution problem.* The attribute representation can only capture token-attribute relations while ignoring record-attribute and more general token-record relations which carry semantics that among different attributes, *i.e.*, tokens in a same record but within different attributes could also share a strong semantic correlation (*e.g.*, "microsoft" and "powerpoint" in *r3* of Table 1). Another problem is information dilution. Since the granularity of an attribute is coarse-grained, especially for descriptive textual attributes, it is hard for few key tokens (*e.g.*, "upgrade" in entity *r3* of Table 1) to show enough significance in the whole sentence.

ii) *Hard attribute alignment causes inflexible comparison.* The prerequisite for attribute comparison is that the schemas of all records are identical, which is usually achieved by an extra hard schema-mapping step (*e.g.*, mapping TITLE↔DESCRIPTION, MANUFACTURE↔MANUFACTURE, and PRICE↔PRICE for the two tables in Table 1), error-

*B Li and W. Wang are the co-corresponding authors.

prone and inflexible in handling schema heterogeneity and noisy data. For example, in entity *r1* of Table 1, the value of MANUFACTURER is missing (although implied in TITLE attribute), thus, comparing on this attribute will lead to misleading results.

iii) *Incapable of handling heterogeneous attribute types.* The attributes of records are often heterogenous, *i.e.*, they have different data types. For example, in Table 1, TITLE and MANUFACTURER are textual while PRICE is numerical. Obviously, different data types should be treated differently during representation learning, *e.g.*, a good model is supposed to capture co-occurrence relation for textual words and relative differences for numerical values. Unfortunately, existing DL-based ER models often adopt vanilla RNN or LSTM to encode attributes, which are unable to adaptively distinguish different data types. Some research efforts have sought a compromised strategy, adopting different comparison function for different data type (Kong et al. 2016; Jurek and Deepak 2018), which is cumbersome and incompatible to up-to-date deep learning fashion.

Observing above drawbacks of existing models, we propose a novel ER model *GraphER*, in the hope to simultaneously address the above three problems via a concise, unified, and end-to-end deep learning framework. Our model offers a fundamentally different perspective for ER task: represent and compare entity records in a *token-centric* manner, and the final ER decisions are made by directly aggregating token-level comparison features. To be specific, we model record-attribute, attribute-token, and token-token relations of all records into a single Entity Record Graph (ER-Graph) where the token-token edge is type-sensitive that enable us to effectively handle heterogeneous attribute types (address problem iii). On top of ER-Graph, we employ Graph Convolutional Networks (GCNs) (Kipf and Welling 2017), a special form of Laplacian smoothing, to capture high-order schematic and structural information and embed them into token representations. In this way, any nodes (including records, attributes, and tokens) can get updated as long as they share common paths in ER-Graph with current matching pairs, thus the semantic sparsity could be greatly reduced (address problem i). For records comparison, we propose a novel cross-encoding and token-gating comparison method, which is helpful to prevent to information dilution problem. Moreover, since token-centric paradigm inherently avoids hard attribute alignment, our model is of high flexibility in handling schema heterogeneity and noisy data (address problem ii).

To the best of our knowledge, this is the first effort to address structured ER problem in such a token-centric manner. The evaluation on *Amazon-Google* and *BeerAdvo-RateBeer* datasets demonstrates our model significantly outperforms state-of-the-art ER models.

Method

Model Overview

The architecture of our model is summarized in Figure 1. From bottom to top, our model is a four-layer layout. Given a matching pair $\langle P, Q \rangle$, the first layer generates embedding that implicitly contains structural information. Then, the comparison layer is employed to compare the matching pair token by token. Finally, the aggregation layer aggregates comparison features into a fixed-length matching vector and feed it into the prediction layer for an ER decision.

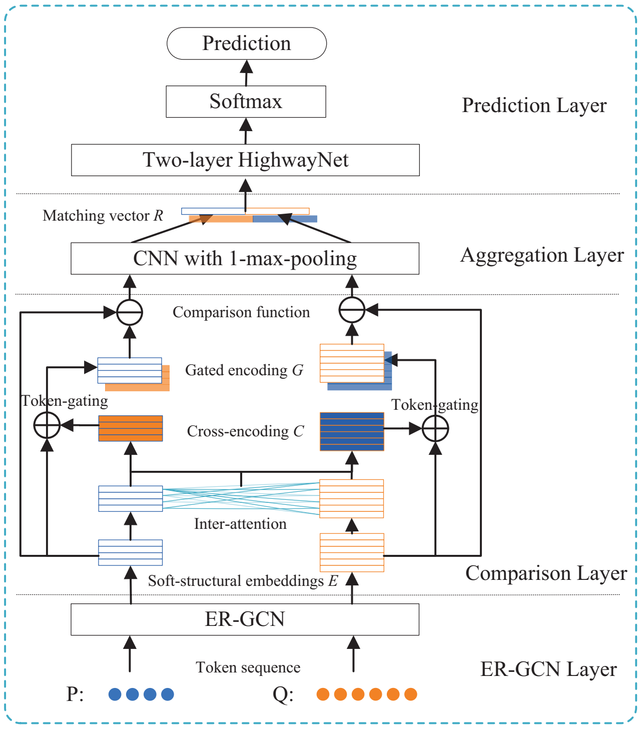


Figure 1: Architecture of the proposed model.

ER-GCN Layer. This layer aims to represent each token into a d-dimensional vector in which both the semantic and structural information (including attribute types and record-attribute-token hierarchy) are “softly” kept for future comparison and aggregation.

Comparison Layer. This layer compares the two matching records and outputs two sequences of comparison vectors. It mainly contains three steps: firstly, we perform a cross-encoding step to capture the token-level semantic inclusion relation between the two entities; secondly, we use a token-gating mechanism to adaptively weight the importance of tokens to lower the significance of unimportant words (*e.g.*, auxiliary words); finally, a comparison function is employed to yield the final comparison vectors.

Aggregation Layer. In this layer, we employ a one-layer Convolutional Neural Network (CNN) to find the important matching features and aggregate them into a fixed-length matching vector to feed as the input of the prediction layer.

Prediction Layer. This layer is to evaluate the probability distribution $p(y|P, Q)$ based on the matching features generated by the aggregation layer. We employ a two-layer HighwayNet (Srivastava, Greff, and Schmidhuber 2015) followed by a $softmax$ function. For ER task, the final prediction contains two values, and each indicates the probability of match/non-match.

We will elaborate on the first three layers in the following subsections.

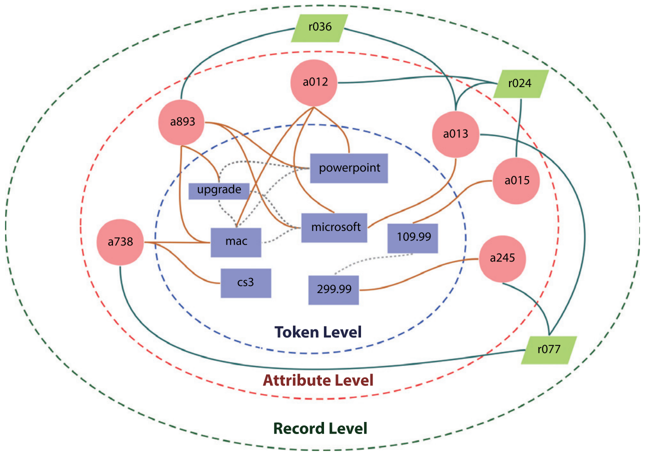


Figure 2: An excerpt from *Amazon-Google* dataset's ER-Graph. Green parallelogram nodes denote entity records, and red circular nodes denote attributes, while others are token nodes.

Entity Record Graph Convolutional Networks (ER-GCN)

Entity Record Graph (ER-Graph). We build an entity record graph to model schematic and structural information that can be easily adapted to graph convolution. An ER-Graph G can be formally denoted as $G = (V, E)$, where V is the node set and E is the edge set. The number of nodes (*i.e.*, $|V|$) is the number of all entity records plus unique attribute and unique tokens. As shown in Figure 2, there are connections between record and attributes as well as attributes and tokens as long as there exist inclusion relations between them. At the token-level, tokens co-occurring in the same sliding window (textual) or with similar numerical values are also connected.

Formally, the weight of edge between node i and node j is defined as

$$A_{i,j} = \begin{cases} \text{IDF}_{i,j} & i \text{ is record, } j \text{ is attribute} \\ \text{TF-IDF}_{i,j} & i \text{ is attribute, } j \text{ is token} \\ \text{TSW}(i,j) & i,j \text{ are tokens} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $\text{TSW}(i,j)$ is the type-sensitive weight function to compute the edge weight between i, j considering the data type $t(i)$ and $t(j)$, which is defined as

$$\text{TSW}(i,j) = \begin{cases} \text{PPMI}(i,j) & t(i), t(j) \text{ are textual} \\ \max(0, 1 - \frac{2*|i-j|}{i+j}) & t(i), t(j) \text{ are numerical} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The PPMI value is the positive point-wise mutual information of token i and j, which is computed as

$$\text{PPMI}(i,j) = \max\left(0, \log\frac{\#(i,j)\cdot\#W}{\#(i)\cdot\#(j)}\right), \tag{3}$$

where $\#(i,j)$ is the co-occurrences of i, j in sliding windows over textual attributes, $\#W$ is the total number of sliding windows, and $\#(i)$ and $\#(j)$ are the occurrences of i and j, respectively.

A higher PPMI value indicates a higher semantic correlation of tokens. Thus, for textual attributes, we employ PPMI to weight the semantic correlation between words. The PPMI weighting also enables our ER-GCN model to mimic the behavior of conventional skip-gram model whose objective is equivalent to implicitly factorizing a PPMI matrix (Levy and Goldberg 2014).

For numerical type, we employ relative similarity to weight numerical pairs, and simply cut-off the weights less than 0 to exclude very different values from computation.

It is worth noting that, the adjacency matrix A is non-negative and symmetric, indicating that it is strictly positive semi-definite, which is a necessity to perform Fourier transform-based convolution on G.

Graph Convolutional Networks (GCNs). Given the adjacent matrix A of graph G, the spectral convolutions (Bruna et al. 2013) on G's node feature matrix $X \in \mathbb{R}^{n\times m}$ is defined as

$$g_\theta \otimes X = U g_\theta U^\top X, \tag{4}$$

where g_θ denotes the convolution kernel, U is the eigenvectors of graph-Laplacian (commonly defined as $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U\Lambda U^\top$, where $D_{ii} = \sum_j A_{i,j}$), and $U^\top X$ is the Fourier transform of X. Considering its computational cost is expensive ($\mathcal{O}(n^2)$ as the multiplican of eigenvectors), we employ the layer-wise linear GCN (LGCN) (Kipf and Welling 2017; Yao, Mao, and Luo 2019) instead, which has a linear computational complexity with stellar results.

For each convolution layer, LGCN can capture information within 1-hop neighbors, and information within i-hop neighbors can be fetched by sequentially stacking i convolution layers. ρ is an activation function, such as ReLU or Leaky ReLU. The new k-dimensional node feature matrix $L^{(i)} \in \mathbb{R}^{n\times k}$ of i-th layer is computed as

$$L^{(i)} = \rho(\widetilde{A} L^{(i-1)} \Theta^{(i)}), \tag{5}$$

where $\widetilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix, and $\Theta^{(i)} \in \mathbb{R}^{m\times k}$ is a weight matrix. Initially, $L^{(0)}$ is an identity matrix, *i.e.*, $L^{(0)} = I$.

In our ER-GCN model, we employ a two-layer LGCN with ReLU activation function, and the final node embeddings E can be computed as

$$E = \text{ReLU}(\widetilde{A}\,\text{ReLU}(\widetilde{A} I \Theta^{(1)})\Theta^{(2)}). \tag{6}$$

This two-layer ER-GCN model enables that (1) since the nodes belonging to a same record are at maximum 2-hop away, information within a same record can be comprehensively integrated; (2) although there are no direct record-record links in the graph, information can still be exchanged among different records via token-attribute-record paths. The above intuition is in accordance with our experimental results showing that two-layer GCN performs better than other layer sizes.

Records Comparison via Hybrid Attention

The comparison layer consists of three steps: cross-encoding, token gating, and comparison. Cross-encoding aims to encode each token in P against Q (or encode Q against P) by an inter-attention mechanism. Token gating use intra-attention to adaptively weight the importance of tokens and allows tokens to be encoded by themselves according to their importance. Finally, we use a comparison function to generate a comparison vector by comparing encoded representation and the original representation.

Please note that, we make a bilateral matching (Wang, Hamza, and Florian 2017), *i.e.*, compare P with Q in two directions (both $P \rightarrow Q$ and $Q \leftarrow P$). To avoid the repetition, we only define our comparison method for one matching direction $P \rightarrow Q$ in the following. The readers can infer the reverse direction easily.

Cross-encoding via inter-attention. We feed two entity records $P = t_1^P, t_2^P, ..., t_{|P|}^P$ and $Q = t_1^Q, t_2^Q, ..., t_{|Q|}^Q$ into an ER-GCN to get their output embeddings $E_P = [e_1^P; e_2^P; ...; e_{|P|}^P]^\intercal \in \mathbb{R}^{|P| \times d}$, and $E_Q = [e_1^Q; e_2^Q; ...; e_{|Q|}^Q]^\intercal \in \mathbb{R}^{|Q| \times d}$. The inter-attention between the t_i^P and t_j^Q can be computed by

$$\alpha_{i,j} = \frac{\exp(g(e_i^P \odot e_j^Q))}{\sum_{k=1}^{|Q|} \exp(g(e_i^P \odot e_k^Q))}, \tag{7}$$

where $g(X) = W^\intercal X + b$ is a linear layer, where W and b are both parameters. $\odot$ denotes the Hadamard product. The inter-attention score is symmetric in P and Q.

Let $\boldsymbol{\alpha_i} = [\alpha_{i,1}; \alpha_{i,2}; ...; \alpha_{i,|Q|}] \in \mathbb{R}^{1 \times |Q|}$ be the attention vectors of t_i^P against all tokens in Q, then t_i^P's cross-encoding $c_i^P \in \mathbb{R}^{1 \times d}$ can be computed as the weighed sum of E_Q:

$$c_i^P = \boldsymbol{\alpha_i} E_Q. \tag{8}$$

The cross-encoding of P is the concatenation of c_i^P

$$C_P = [c_1^P; c_2^P; ...; c_{|P|}^P]^\intercal. \tag{9}$$

The intuition behind cross-encoding is that, if the semantic of token t_i^P contained in Q, it could be restored to the original state e_i^P by encoding Q, making cross-encoding c_i^P similar to e_i^P. Thus, one can conclude if token i is matched/mismatched by simply comparing c_i^P with e_i^P.

Token gating via intra-attention. Albeit cross-encoding could represent the matching relations between tokens, it is unable to weight the importance of a token itself. Commonly, the importance of a token in matching could be totally different, for example, consider the DESCRIPTION attribute of entities *r2* and *r3* in Table 1: "powerpoint 2004 *upgrade* mac" and "powerpoint 2004 mac *by* microsoft". Both "upgrade" and "by" mismatch in their counterparts. However, "upgrade" is obviously more important than "by" which is just an auxiliary word that can be neglected.

We use an intra-attention mechanism to adaptively weight the importance of tokens. The intra-attention weights $\beta^P \in \mathbb{R}^{1 \times n}$ are computed by

$$\beta^P = \sigma(w_2 \tanh(W_1 E_P^\intercal)), \tag{10}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, $W_1 \in \mathbb{R}^{d_a \times d}$ and $w_2 \in \mathbb{R}^{1 \times d_a}$ are parameters with hidden unit numbers d_a, where d_a is a hyper-parameter we can set arbitrarily.

Based on intra-attention weights, the token gating yields the gated encoding G_P as a mixture of the cross-encoding C_P and the original encoding E_P:

$$G_P = B_P C_P + (I - B_P) E_P, \tag{11}$$

where I denotes identity matrix, and $\odot$ denotes the Hadamard product, and B_P is the diagonal matrix of β^P:

$$B_P = \begin{bmatrix} \beta_1^P & & & \\ & \beta_2^P & & \\ & & \ddots & \\ & & & \beta_{|P|}^P \end{bmatrix}. \tag{12}$$

Here intra-attention weights act as gate values to control the degree to which original encoding is "mixed" into the cross-encoding. For example, for auxiliary words, their gate values would be likely low thus making it similar to its original encoding.

Comparison. We employ subtraction function (Wang and Jiang 2017) to compare E_P with G_P, and yield a comparison matrix $M^{(P \rightarrow Q)} \in \mathbb{R}^{|P| \times d}$:

$$M^{(P \rightarrow Q)} = (E_P - G_P) \odot (E_P - G_P). \tag{13}$$

Eq. 13 implies the Euclidean distance between E_P and G_P, where G_P comes from the cross-encoding from Q. Thus, if P is similar to Q, G_P should be similar to E_P, and subtraction function would yield a comparison matrix with smaller element values (implies a smaller Euclidean distance); if P is dissimilar to Q, the comparison function will yield a matrix with larger element values (implies a larger Euclidean distance).

Aggregation Layer

In aggregation layer, we employ a one-layer CNN to aggregate the comparison matrix $M^{(P \rightarrow Q)}$ into a fixed-length matching vector $r^{(P \rightarrow Q)}$:

$$r^{(P \rightarrow Q)} = \text{CNN}(M^{(P \rightarrow Q)}), \tag{14}$$

where CNN$(\cdot)$ is a composite function consisting of four cascaded operations: a convolutional operation, a 1-max-over-time pooling operation (Kim 2014), a dropout operation, and a ReLU unit. The convolutional layer has w filter widths and l filters, and each filter has the size of $h \times d$ (h denotes the filter width), which is helpful to capture multi-scale matching features (*i.e.*, h-grams). The 1-max-pooling

Table 2: Comparisons of our models with state-of-the-art models on entity resolution task. Except Magellan, we run all models 10 times and report mean $\pm$ standard deviation. The best results are marked in bold font.

Model	*Amazon-Google*			*BeerAdvo-RateBeer*		
	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
Magellan (Konda et al. 2016)	67.7	38.5	49.1	68.4	92.9	78.8
RNN (Mudgal et al. 2018)	59.33 ± 4.40	48.12 ± 6.06	52.77 ± 3.07	74.82 ± 4.48	70.00 ± 15.36	71.34 ± 7.53
Hybrid (Mudgal et al. 2018)	58.82 ± 5.43	64.02 ± 12.36	60.51 ± 4.73	73.44 ± 9.43	70.00 ± 8.11	71.08 ± 5.80
GraphER	$69. 11\pm 1.70$	67.13 ± 2.26	**68.08 ± 1.50**	79.34 ± 7.84	80.81 ± 5.41	**79.71 ± 2.16**

operation selects the largest value over the feature map of a particular filter to capture the most important feature. The output vector $r^{(P \to Q)}$ has a fixed size $1 \times wl$.

The final matching vector $R \in \mathbb{R}^{1 \times 2wl}$ is generated by concatenating the matching vectors of both $P \to Q$ and $Q \to P$ directions:

$$R = [r^{(P \to Q)}; r^{(Q \to P)}]. \tag{15}$$

Finally, we feed R into a two-layer fully-connected ReLU HighwayNet to get the final ER decision.

Objective Function

The training objective is to minimize standard cross-entropy loss:

$$\phi = \mathcal{L}(y, h(W, R)), \tag{16}$$

where $\mathcal{L}(l, p)$ denotes cross-entropy function between l and p. $h(W, R)$ is the predicted distribution of the final prediction layer, and y is the golden label.

Experimental Evaluation

In this section, we evaluated our model on entity resolution task, and demonstrated its superiority over state-of-the-art models.

Table 3: Statistics of the two datasets for experiments.

Datasets	Table Sizes	# Labeled	# Pos.	# Attr.
Amazon-Google	(4344, 2999)	11460	962	3
BeerAdvo-RateBee	(1362, 3225)	450	68	4

Evaluation Datasets and Metric

We used two datasets, each contains two tables, and a list of golden matches. Following (Mudgal et al. 2018), the golden matches lists is an after-blocking candidate set using (Konda et al. 2016). For both datasets, we use the same 3:1:1 train/dev/test split as in (Mudgal et al. 2018).Their detailed statistics are summarized in Table 3.

- **Amazon-Google** contains software product data from Amazon and Google (Köpcke, Thor, and Rahm 2010). The tables contains 3 attributes: (TITLE, textual); (MANUFACTURES, textual); (PRICE, numerical). Attribute TITLE commonly contains rich and important information (*e.g.*, software version), which is descriptive and contains a lot of synonyms across matching pairs. Note that in this dataset, there are 135 records, over 962 positively labeled entities pairs (or roughly 14%), has more than one matching records (maximum matching records is 5).
- **BeerAdvo-RateBeer** contains beer product data from BeerAdvo and RateBeer websites (Mudgal et al. 2018). The tables contains 3 attributes: (BEER_NAME, textual); (BREW_FACTORY_NAME, textual); (STYLE, textual); (ABV, numerical). This dataset is more cleaner and shallower than Amazon-Google.

Following the previous research efforts, we reported precision (P), recall (R), and F1 score (F1) on test datasets.

Training Settings

For ER-GCN, the size of $\Theta^{(1)}$ was set to $|V| \times 300$, $|V|$ was the number of nodes in corresponding ER-Graph, and the size of $\Theta^{(2)}$ was 300×200. The textual window size was set to 20. Token embeddings were initiallized using 300-dimensional pretrained vectors of Glove[1], while unknown words were initialized with an embedding drawn from a uniform distribution $U(-0.25, 0.25)$. All the weight matrices in ER-GCN are initialized using Xavier initialization (Glorot and Bengio 2010) with gain 1. d_a in Eq. 10 was set to 350.

For the CNN used in aggregation layer, we took three filter widths $[1, 2, 3]$, each filter width having 150 kernels. For the final prediction layer, the number of hidden units of HighwayNet is set to 4000.

For optimization, we used Adam (Kinga and Adam 2015) with an initial learning rate 0.001, dropout rate as 0.5, and the gradient clipping to 5; the batch size to be 32 and 3 for *Amazon-Google* and *BeerAdvo-RateBeer* dataset, respectively; all other hyper-parameters were their default values.

We trained the model for a maximum of 100 epochs, and stopped training if the validation loss did not decrease by 10 consecutive epochs. Unless noted otherwise, we used the same hyper-parameter settings in all the experiments.

Baselines

We compared our model GraphER with three existing models, including two state-of-the-art DL-based models: RNN (Mudgal et al. 2018), and Hybrid (Mudgal et al. 2018), and one state-of-the-art ML-based (non-DL) ER system Magellan (Konda et al. 2016).

Main Results

Table 2 lists the results of our GraphER model compared with state-of-the-art models on entity resolution task. To bet-

[1]https://nlp.stanford.edu/projects/glove/

Table 4: Ablation study results compared with the full GraphER model.

Model	*Amazon-Google*				*BeerAdvo-RateBeer*			
	P (%)	R (%)	F1 (%)	Δ F1	P (%)	R (%)	F1 (%)	Δ F1
GraphER	69.11	67.13	**68.08**	-	79.34	80.81	**79.71**	-
- ER-GCN	66.72	49.61	56.91	-11.17	54.55	85.71	66.67	-13.04
- Cross-encoding	34.29	20.52	25.67	-42.41	50.00	28.57	36.36	-43.35
- Token-gating	68.64	64.53	66.52	-1.56	77.72	77.04	77.38	-2.33
- Pretrained embeddings	70.42	64.10	67.11	-0.97	79.30	75.83	77.53	-2.18

ter evaluate the perforamance and stability of the three DL-based model – GraphER, RNN and Hybrid, we ran each model 10 times and reported their mean $\pm$ standard deviation. For the ML-based model Magellan, we directly used performance reported in (Mudgal et al. 2018). From Table 2, we can see that:

1) Our model stably outperformed all existing models in terms of F1 score, achieving new state-of-the-art results on these two datasets. Compared with attribute-centric models RNN and Hybrid, GraphER significantly outperformed them by 7.57 percentage points (pts) and 8.37 pts on *Amazon-Google* and *BeerAdvo-RateBeer* dataset, respectively. This demonstrates our ER-GCN and token-centric strategy could greatly improve accuracy in matching entity records and account for its better performance. Compare with ML-based model Magellan, our model achieved 18.98 pts and 0.91 pt F1 improvements on the two datasets. This demonstrate DL-based model could beat traditional ML-based methods not only on semantically deep dataset but also on clean and shallow dataset.

2) On *Amazon-Google* dataset, the three DL-based models (RNN, Hybrid and GraphER) performed much better than the non-DL model Magellan. This is because the most informative attribute is TITLE, which is descriptive and contains a lot of synonyms across matching pairs. That is, they are semantically similar but may have large string similarity distance. Thus, it is better for DL-based models to capture semantic information on this dataset than feature-based model Magellan which mainly relies on computing string similarity. Among the three DL-based models, GraphER achieved the best performance. This indicates our token-centric representation and comparison strategy could capture fine-grained information within single (or few) informative attribute, which is helpful for information dilution problem.

3) On *BeerAdvo-RateBeer* dataset, the performance of the three DL-based models were not as better as on *Amazon-Google* dataset. Especially, RNN and Hybrid performed even worse than non-DL model Magellan. The reason lies in the fact that all the textual attributes are clean and shallow in this dataset, and thus, the numerical attribute ABV as an importance indicator makes more impact on final results. RNN and Hybrid suffer more from heterogeneous attribute types, since they do not distinguish different attribute types. By contrast, our model performs better than Magellan indicating our model could be type-sensitive and accounts for its better performance.

Detailed Analysis

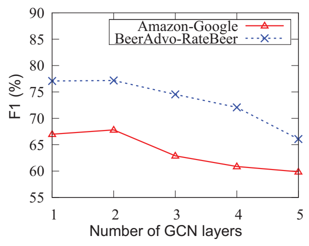
Ablation Study. To evaluate the contribution of ER-GCN, cross-encoding, token-gating, and pretrained embeddings components to final results, we conducted an ablation study by ablating a specific component from the full model once per test. The results are shown in Table 4. We can see that ablation on ER-GCN results in roughly 12 pts performance decline, which demonstrates the structural information is important and the ER-GCN layer is critical for model performance. The biggest performance gap happened in removing the cross-encoding component. The main reason is the model cannot effectively generate comparison features without cross-encoding, which is critical for detecting matching relation. Another observation is, there are only slight differences after removing pretrained embeddings, and hence demonstrates GraphER can achieve good result only using information in the target input dataset, without relying on too much external information. From above we can conclude that these components contribute significantly to the effectiveness of our model.

Table 5: Comparison of record-centric, attribute-centric and token-centric comparison strategy.

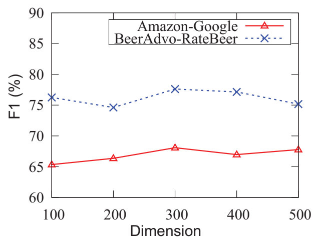
Model	*Amazon-Google*		*BeerAdvo-RateBeer*	
	F1 (%)	Δ F1	F1 (%)	Δ F1
Record-centric	45.54	-22.54	57.27	-22.44
Attribute-centric	47.87	-20.21	52.55	-27.16
Token-centric	**68.08**	-	**79.71**	-

Effects of Token-centric Comparison Strategy. To analysis the effectiveness of our token-centric representation and comparison strategy, we compare it with attribute-centric and record-centric strategy by applying the same comparison layer but on different granularity. The results are shown in Table 5. We can see that the token-centric strategy had a huge advantage (more than 20 pts) over others. We believe it is mainly caused by the information dilution problem when compare the matching pair in attribute and record levels. Another possible reason is the cross-encoding and token-gating operations are more suitable for token-centric comparison, while it may entail errors in attribute and record levels where soft-alignment are unnecessary. This again verified the effectiveness of our model.

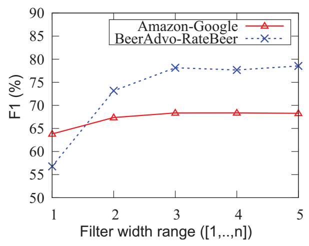
Parameter Sensitivity. In order to test the efforts of hyper-parameters on F1 scores, and offer an empirical way to tune its setting, we conducted a parameter sensitivity analysis. The results are shown in Fig. 3.



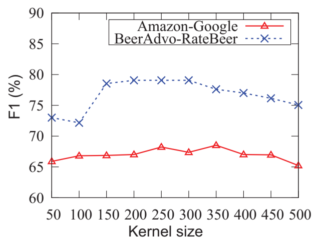
(a) F1 v.s. different number of GCN layers.



(b) F1 v.s. different embedding dimensions.



(c) F1 v.s. different filter width ranges.



(d) F1 v.s. different kernel sizes.

Figure 3: F1 scores by varying hyper-parameters on Dev set.

Fig. 3(a) shows F1 scores with different GCN layers depth on *Amazon-Google* and *BeerAdvo-RateBeer* datasets. We can see that the performance of two-layer GCN is slightly better than single layer GCN, but there is a significant performance drop after stacking more than 2 layers. This suggests that single layer GCN could not fully integrate useful information that outsides 1-hop neighbors, while stacking too many layers will make the value of each node tend to be the average over the entire graph under the influence of Laplacian smoothing, which will entail too much noisy thus will lower F1 score.

Fig. 3(b) depicts the F1 scores with different dimension of the-first layer embeddings. We found that low dimensional embeddings may not fit the model well, while high dimensional embeddings do not effectively improve F1 scores and may cost more training time. Thus, an appropriate dimension setting is among $200 - 400$.

Fig. 3(c) and Fig. 3(d) shows the F1 scores with different ranges of filter width and kernel sizes. We can see that, solely using 1-gram filter could not provide a good enough performance for both datasets. This indicates consecutive n-gram comparison features are important in identifying matching pairs. Considering the extra computational cost brought by a larger width range, a smaller range (*e.g.*, 2 or 3) is preferable. With the increase of the kernel size, F1 scores rise to a peak region and then has a slow drop down. The most appropriate kernel size is supposed to be among 150 - 350, and further, considering the training cost, we choose 150 as our setting.

Related Works

Existing works can be broadly classified into three categories: rule-based solutions, crowd-based solutions, and machine learning (ML) based solutions.

Rule-based solutions either rely on predefined declarative matching rules such as DNF (Hernández and Stolfo 1995; Arasu, Ré, and Suciu 2009) or dynamically synthesized entity matching rules (Singh et al. 2017) to find matching pairs. Rule-based solution is easily interpretable, however, heavily relying on domain experts and lacking enough flexibilities to handle heterogeneous data schema.

To alleviate the drawbacks of rule-based solutions, crowd-based solutions (Wang et al. 2012; Firmani, Saha, and Srivastava 2016) have been proposed to employ crowd-sourcing workers to manually identify matching tuples. However, the human labor cost is extremely expensive.

ML-based solutions try to automatically learn matching functions (Bilenko and Mooney 2003) with limited annotations. Traditional ML-based approaches mostly design different similarity measures (*e.g.*, jaccard similarity, ED-distance) (Konda et al. 2016), and then train a classifier on top of similarity features, such as SVM (Bilenko and Mooney 2003), activate learning (Sarawagi and Bhamidipaty 2002), or clustering-based classifier (Cohen and Richman 2002). In order to further refine the quality of learned matching functions, some other works (Stonebraker et al. 2013; Gokhale et al. 2014) leverage human experts to guide the learning process.

Most recent efforts (Thirumuruganathan, Tang, and Ouzzani 2018) have sought deep learning (DL) techniques for better attribute representation. Ebraheem *et al.* (2018) proposed a DL-based ER system DeepER, which represents each attribute as the aggregation of its words' representations using RNN and LSTM. Another DL-based ER model DeepMatcher (Mudgal et al. 2018) used a bidirectional RNN with attention mechanism for attribute summarization. The decoding steps for both DeepER and DeepMatcher are similar: compare attribute representations using various comparison functions (*e.g.*, cosine similarity, element-wise subtraction *etc.*), and then employ a dense layer to get the ER decision on top of attribute-level comparison features. The advantage of DL-based approaches is that they can capture semantic similarity better especially for textual attributes. Another closely related task is unstructured matching (*e.g.*, QA-matching) (Wang and Jiang 2017; Wang, Hamza, and Florian 2017). Since the important structural information are neglected by those methods, their performance on ER task is commonly not competitive to specifically designed ER models.

Existing DL-based ER models are attribute-centric, which need to manually align schema for different data sources, and too coarse-grained to capture subtle key information. By contrast, our work is the first effort to perform ER task in a token-centric manner with the help of GCN.

Conclusion

In this paper, we propose a new token-centric paradigm for DL-based ER model. Instead of representing and comparing entity records in attribute-level, our model first integrate schematic and structural information into token representations with ER-GCN, and the final ER decisions are fetched by directly aggregating token-level comparison features. Compared with the best state-of-the-art models, our model gains a 7.57 pts and 0.91 pt improvements on Amazon-Google and BeerAdvo-RateBeer dataset, respectively.

Acknowledgements. This work is supported by ARC DPs 170103710 and 180103411, D2DCRC DC25002 and DC25003, NSFC Grant 61876038 and Dongguan University of Technology Project KCYKYQD2017003. The Titan V used for this research was donated by the NVIDIA Corporation.

References

Arasu, A.; Ré, C.; and Suciu, D. 2009. Large-scale deduplication with constraints using dedupalog. In *ICDE*, 952–963. IEEE.

Bhattacharya, I., and Getoor, L. 2007. Collective entity resolution in relational data. *TKDD* 1(1):5.

Bilenko, M., and Mooney, R. J. 2003. Adaptive duplicate detection using learnable string similarity measures. In *SIGKDD*, 39–48. ACM.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Chen, Y.-N.; Wang, W. Y.; Gershman, A.; and Rudnicky, A. 2015. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *ACL*, 483–494.

Cohen, W. W., and Richman, J. 2002. Learning to match and cluster large high-dimensional data sets for data integration. In *SIGKDD*, 475–480. ACM.

Dong, X. L., and Srivastava, D. 2013. Big data integration. In *ICDE*, 1245–1248. IEEE.

Ebraheem, M.; Thirumuruganathan, S.; Joty, S.; Ouzzani, M.; and Tang, N. 2018. Distributed representations of tuples for entity resolution. *VLDB* 11(11):1454–1467.

Firmani, D.; Saha, B.; and Srivastava, D. 2016. Online entity resolution using an oracle. *VLDB* 9(5):384–395.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, 249–256.

Gokhale, C.; Das, S.; Doan, A.; Naughton, J. F.; Rampalli, N.; Shavlik, J.; and Zhu, X. 2014. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, 601–612. ACM.

Hernández, M. A., and Stolfo, S. J. 1995. The merge/purge problem for large databases. In *ACM Sigmod Record*, volume 24, 127–138. ACM.

Jurek, A., and Deepak, P. 2018. It pays to be certain: unsupervised record linkage via ambiguity minimization. In *PAKDD*, 177–190. Springer.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, 1746–1751.

Kinga, D., and Adam, J. B. 2015. A method for stochastic optimization. In *ICLR*, volume 5.

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Konda, P.; Das, S.; Suganthan GC, P.; Doan, A.; Ardalan, A.; Ballard, J. R.; Li, H.; Panahi, F.; Zhang, H.; Naughton, J.; et al. 2016. Magellan: Toward building entity matching management systems. *VLDB* 9(12):1197–1208.

Kong, C.; Gao, M.; Xu, C.; Qian, W.; and Zhou, A. 2016. Entity matching across multiple heterogeneous data sources. In *DASFAA*, 133–146. Springer.

Köpcke, H.; Thor, A.; and Rahm, E. 2010. Evaluation of entity resolution approaches on real-world match problems. *VLDB* 3(1-2):484–493.

Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*, 2177–2185.

Mudgal, S.; Li, H.; Rekatsinas, T.; Doan, A.; Park, Y.; Krishnan, G.; Deep, R.; Arcaute, E.; and Raghavendra, V. 2018. Deep learning for entity matching: A design space exploration. In *SIGMOD*, 19–34. ACM.

Sarawagi, S., and Bhamidipaty, A. 2002. Interactive deduplication using active learning. In *SIGKDD*, 269–278.

Singh, R.; Meduri, V. V.; Elmagarmid, A.; Madden, S.; Papotti, P.; Quiané-Ruiz, J.-A.; Solar-Lezama, A.; and Tang, N. 2017. Synthesizing entity matching rules by examples. *VLDB* 11(2):189–202.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. In *ICML*.

Stonebraker, M.; Bruckner, D.; Ilyas, I. F.; Beskales, G.; Cherniack, M.; Zdonik, S. B.; Pagan, A.; and Xu, S. 2013. Data curation at scale: The data tamer system. In *CIDR*.

Thirumuruganathan, S.; Tang, N.; and Ouzzani, M. 2018. Data curation with deep learning [vision]: Towards self driving data curation. *arXiv preprint arXiv:1803.01384*.

Wang, S., and Jiang, J. 2017. A compare-aggregate model for matching text sequences. In *ICLR*.

Wang, J.; Kraska, T.; Franklin, M. J.; and Feng, J. 2012. Crowder: Crowdsourcing entity resolution. *VLDB* 5(11):1483–1494.

Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*, 4144–4150. AAAI Press.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *AAAI*, volume 33, 7370–7377.