Soft Target-Enhanced Matching Framework for Deep Entity Matching

Wenzhou Dou[1], Derong Shen[1]*, Xiangmin Zhou[2], Tiezheng Nie[1], Yue Kou[1], Hang Cui[3], Ge Yu[1]

[1]Northeastern University, China
[2]RMIT University, Australia
[3]University of Illinois at Urbana-Champaign, USA
wenzhoudou@gmail.com, {shenderong, nietiezheng, kouyue, yuge}@cse.neu.edu.cn
xiangmin.zhou@rmit.edu.au, hangcui2@illinois.edu

Abstract

Deep Entity Matching (EM) is one of the core research topics in data integration. Typical existing works construct EM models by training deep neural networks (DNNs) based on the training samples with onehot labels. However, these sharp supervision signals of onehot labels harm the generalization of EM models, causing them to overfit the training samples and perform badly in unseen datasets. To solve this problem, we first propose that the challenge of training a well-generalized EM model lies in achieving the compromise between fitting the training samples and imposing regularization, i.e., the *bias-variance tradeoff*. Then, we propose a novel **S**oft **T**arget-**E**nh**A**nced **M**atching (**STEAM**) framework, which exploits the automatically generated soft targets as label-wise regularizers to constrain the model training. Specifically, STEAM regards the EM model trained in previous iteration as a virtual teacher and takes its softened output as the extra regularizer to train the EM model in the current iteration. As such, STEAM effectively calibrates the obtained EM model, achieving the bias-variance tradeoff without any additional computational cost. We conduct extensive experiments over open datasets and the results show that our proposed STEAM outperforms the state-of-the-art EM approaches in terms of effectiveness and label efficiency.

Introduction

Entity Matching (EM) aims to identify the matching record pair that refer to the same real-world entity. As a fundamental research in data cleaning and integration, EM has been widely applied in many application scenarios including E-commerce and population census etc. Considering two data sources in Figure 1, EM determines whether the candidate record pairs (i.e., $a1$ and $b1$, $a1$ and $b2$) are matching or not, such that data integration and management tasks can be subsequently conducted based on the matching results. The comprehensive understanding of record semantics has become increasingly important, which requires deep EM as solutions in real scenarios.

Existing deep EM can be categorized into two types: deep learning-based (DL-based) (Mudgal et al. 2018; Fu et al. 2019; Li et al. 2020a; Fu et al. 2020) and pretrained language model-based (PLM-based) (Brunner and Stockinger 2020; Li et al. 2020b; Yao et al. 2022). DL-based approaches treat the elements of each record as a token sequence or a graph topology, which are then fed into deep learning models to predict the matching probability of the record pair from two given sources. And PLM-based approaches concatenate the record pair as a sentence in the format of $[cls]\ record_1\ [sep]\ record_2\ [sep]$, and feed it into the PLM to obtain the highly contextualized embeddings which improve the effectiveness of EM models. Although these approaches have good performance on training pairs, both types of EM models face the common problem of poor performance on unseen testing pairs. Due to the diversity of record distribution caused by unreliable sampling strategy (Goodfellow, Bengio, and Courville 2016), the information in training data can be far from that in testing data, causing the low quality of matching for testing data using the trained EM model. Thus, there is a requirement for designing an advanced method to well identify the matches over unseen record pairs.

To address this challenge, we need to generalize the obtained EM model to fit the unseen record pairs that belong to the item categories different from those for any training data. The popular approaches designed for generalizing EM models include model-oriented and data-oriented. Model-oriented approaches (e.g., model redesigning (Zhang et al. 2020; Fu et al. 2020)) introduce higher parameterized EM models that have higher capacity to learn universal features. Data-oriented approaches apply data augmentation (Li et al. 2020b, 2021b) or inject domain knowledge (Li et al. 2020b) to improve the data efficiency and avoid the model overfitting against the limited training data. However, both ap-

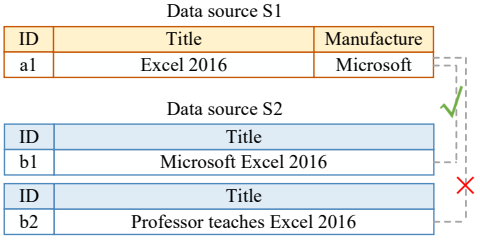


Figure 1: An example of entity matching task.

*Corresponding Author

proaches are highly empirical due to the requirement for model redesigning and data preprocessing, thus inapplicable in real EM scenarios. It is highly demanded that a cost-effective and easy-to-use solution is designed to generalize EM models.

Motivated by the above limitations, we propose a **S**oft **T**arget-**E**nh**A**nced **M**atching (**STEAM** for short) framework to generalize deep EM models. Different from model-oriented and data-oriented approaches, STEAM is a label-oriented solution. The superiority of STEAM lies in its high capability of adapting existing EM approaches without model redesigning or data preprocessing. STEAM achieves these goals by introducing a soft supervised learning process that a label-wise regularizer is used to achieve the compromise between fitting the training data and imposing regularization (Yuan et al. 2020; Zhou and Song 2021). Since the soft labels have higher entropy than the vanilla onehot labels, they provide informative supervision signals (e.g., matching confidence) to calibrate the learning phase, thus the obtained EM model can avoid the imbalance of bias and variance. Specifically, STEAM regards the EM model trained in previous iteration as a virtual teacher and takes its softened outputs as the soft labels to regularize the EM model in current iteration. For every follow-up iterations, the EM model is trained under the supervision of both onehot labels and automatically generated soft labels. The onehot labels mainly provide the optimization direction, and the soft labels regularize the EM model, achieving the balance of bias and variance. Our contributions are as follows:

- We are the first to propose a label-wise regularizer to generalize the EM model. This improves the generalization ability of EM model without the model redesigning or the training data preprocessing.
- We propose the STEAM framework to implement the label-wise regularization in training phase. It leverages the teacher-free knowledge distillation to automatically generate soft labels, which effectively calibrates the model training and well balances the bias and variance.
- We conduct extensive experiments on real-world EM datasets. The results demonstrate that our proposed STEAM framework outperforms the SOTA approaches in terms of effectiveness and label efficiency.

Related Works

Deep entity matching adopts deep neural networks (DNNs) as encoders to encode the similarity features into fixed length vectors. DeepER (Ebraheem et al. 2018) and DeepMatcher (Mudgal et al. 2018) are the earliest attempts to apply deep learning technologies to entity matching tasks. DeepER adopts LSTM to convert the record to a distributed representation to capture the similarity between them. And DeepMatcher is a design space for EM tasks, and reveals the advantages of DL technologies for EM, especially in dirty and textual scenarios. Later, a series of improved methods (Fu et al. 2019; Zhang et al. 2020; Fu et al. 2020) are proposed. MPM (Fu et al. 2019) designs various similarity measures and adaptively selects the optimal measure for heterogeneous attributes in an end-to-end manner. MCA (Zhang et al. 2020) fully exploits the semantic context of embeddings for the record pairs, which takes into account the multi-context attention like self-attention, pair-attention, and global-attention for three types of context. And HierMatcher (Fu et al. 2020) is an end-to-end hierarchical matching network for deep entity matching, which jointly matches the entities in three levels—token, attribute, and entity. Some other works (Li et al. 2020a; Cappuzzo, Papotti, and Thirumuruganathan 2020; Chen, Shen, and Zhang 2021) consider representing the record hierarchy as graph topology and adopt graph representation learning technologies (e.g., GCNs (Kipf and Welling 2016)) to obtain embeddings for record pairs. For example, GraphER (Li et al. 2020a) is a graph-based EM model, which adopts an Entity Record Graph Convolutional Network (ER-GCN) to embed the semantic and structural information of record pairs. EMBDI (Cappuzzo, Papotti, and Thirumuruganathan 2020) designs a compact tripartite graph which effectively represents the syntactic and semantic relationship between the cell values, and uses random walk to obtain the local embeddings for data integration tasks such as entity matching and schema matching.

With the development of pretrained language models, the pretraining-then-finetuning procedure has become a new paradigm for deep EM (Brunner and Stockinger 2020; Li et al. 2020b; Peeters and Bizer 2021; Li et al. 2021a; Dou et al. 2022; Yao et al. 2022). Ditto (Li et al. 2020b) improves the matching capability of PLM-based EM model by developing three optimization techniques including domain knowledge injection, text summarization, and data augmentation. JointBERT (Peeters and Bizer 2021) combines the tasks of entity matching and classification, and proves that this dual-object training paradigm effectively improves the matching quality. GTA (Dou et al. 2022) enhances PLMs for relational data representation by injecting additional hybrid matching knowledge from a hybrid matching graph. HierGAT (Yao et al. 2022) proposes a Hierarchical Graph Attention Transformer Network to capture multiple relationships among entities to improve matching quality.

This paper has a different starting point from the above. We focus on the challenge in training a well-generalized deep EM model with the label-wise perspective. To the best of our knowledge, this is the first effort to realize it in a label-wise way, which avoids the model redesigning and heavy data preprocessing.

Problem and Preliminary

Given a candidate record pair set C, entity matching aims to identify the matching pair set $M \subseteq C$, where each matching record pair refers to the same real-world entity (software, person, shop, and so on). Each record is a set of key-value pairs $r = \{(name_i, val_i)\}_{1 \le i \le k}$, where $name_i$ and val_i refer to the attribute name and the attribute value with textual or numerical type, respectively. The EM model $f(\cdot)$ includes a training phase and a testing phase. This paper aims to generalize EM model in the training phase via label-wise regularization.

We formulate EM as a binary classification task. In the training phase, the EM model $f(\cdot)$ is fed with a training set

Notation	Definition and description
C	The candidate record pair set
D	The training record pair set
M	The matching record pair set
x	The input record pair $x = \langle r_1, r_2 \rangle$, where $x \in C$
L	The label set $L = \{match, unmatch\}$
z_i^t, z_i^s	The output logit of teacher and student
τ	The softening temperature
p_i, p_i^τ	The original and softened output probability
y_i, y_i^τ	The onehot label and soft label
$p_{ce}, \bar{p}_{ce}$	The output matching probability under original supervised training and its average value
$p_{st}, \bar{p}_{st}$	The output matching probability under soft supervised training and its average value
λ, α	The weights of soft training loss

Table 1: Summary of the main notations.

D consisting of many record pairs and labels, where each record pair $x = \langle r_1, r_2 \rangle$ has a corresponding label from $L = \{match, unmatch\}$. In the testing phase, the trained EM model is expected to output the reliable matching probabilities of unseen record pairs. The notations used in this paper are summarized in Table 1.

The proposed STEAM

This section proposes the soft target-enhanced matching framework, STEAM, for the deep entity matching tasks.

Framework Overview

Figure 2 depicts the proposed STEAM framework which consists of an entity matching pipeline and a model training pipeline. The entity matching pipeline contains an EM model to identify the matching pairs. It takes the unseen candidate set as input, and outputs the matching set. The model training pipeline takes the labeled samples as training set and optimizes the EM model in the manner of soft supervised training. The goal of STEAM is to obtain a well-generalized EM model to effectively identify the matching pairs from the unseen candidate set.

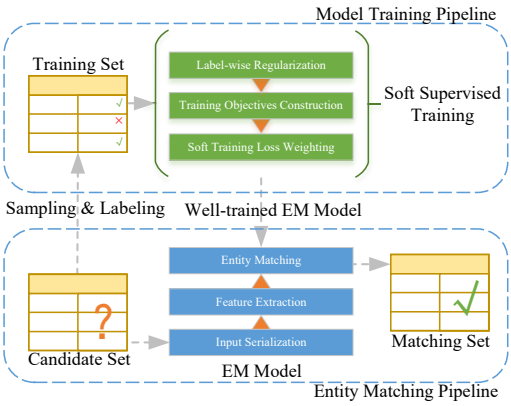


Figure 2: The overview of STEAM framework.

Entity Matching Pipeline in STEAM

The entity matching pipeline consists of an EM model, which has three layers—input serialization layer, feature extraction layer, and entity matching layer.

Input Serialization Layer. For each candidate pair $x = \langle r_1, r_2 \rangle$ as the input of EM model, we serialize it as $[cls]\ r_1\ [sep]\ r_2\ [sep]$. The special tokens $[cls]$ and $[sep]$ act as the separators to split the record pair. We further perform a finer-grained separation. For each record, we split the attributes in the format of $[nam]\ name\ [val]\ value$, where the special tokens $[nam]$ and $[val]$ are the additional attribute-level separators to split each attribute, which indicates the start of attribute names and values.

Feature Extraction Layer. The feature extraction layer is initialized with a PLM such as RoBERTa. It is based on the Transformer architecture and has been pretrained with the large corpus. Therefore it contains the universal language knowledge to understand the semantic features of record pairs. The attention mechanism of Transformer conducts token-level comparison and aggregation, and generates highly contextualized embeddings for record pairs.

Entity Matching Layer. We employ a mean pooling layer as a feature aggregator. It aggregates all the contextualized token embeddings into a similarity vector. We then feed it into a two-layer MLP with softmax to output the matching probability.

Model Training Pipeline in STEAM

The model training pipeline trains highly generalized EM models via our proposed soft supervised training, which consists of three parts—label-wise regularization, training objectives construction, and soft training loss weighting.

Label-wise Regularization. As is shown in Figure 3, we design two training targets for the EM model in STEAM. They are (1) the manual onehot labels y_i for the original supervised training, which is commonly used in most existing EM approaches, and (2) the automatically generated soft labels y_i^τ for the label-wise regularization, which calibrates the model training and balances the bias and variance. It is inspired by the idea of knowledge distillation (KD) that the outputs of the cumbersome model (i.e., teacher) provide rich inter-class relationship, which can be naturally considered as soft target to regularize the lightweight model (i.e., student) label-wisely (Hinton, Vinyals, and Dean 2015). Unlike the original KD, we address the distillation procedure in a teacher-free mode, which takes the previous iteration's EM model as a virtual teacher to regularize the EM model in current iteration. Therefore, for the EM model in current iteration, it takes two ways of supervision, one is the onehot labels y_i from the training set and the other is the soft labels y_i^τ from its virtual teacher. And the soft labels y_i^τ inherently take into account all the previously accumulated information to improve the generalization of obtained EM model.

Training Objectives Construction. We formulate the training objectives from two perspectives. The first is fitting the EM model to the ground truth (i.e., control the bias).

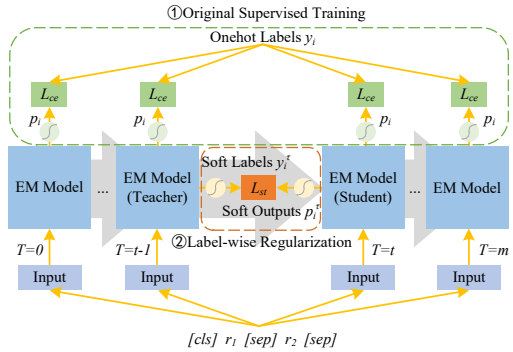


Figure 3: The training iterations of soft supervised training in STEAM.

The second is regularizing the EM model (i.e., control the variance). The training objectives vary across different training iterations. At the beginning of the training iterations ($T = 0$), the EM model takes the labeled pairs as inputs and obtains the sharp supervision signals from the onehot labels y_i. The training objective is to minimize the standard cross-entropy loss:

$$L_{ce} = \mathbb{E}[-y_i \log p_i], \tag{1}$$

where p_i is the model's output probability on the i th class. As the training iterations continue, the subsequent EM model ($1 \leq T \leq m$) additionally acquires soft labels y_i^τ from its previous one. The soft labels y_i^τ can be obtained via temperature softmax:

$$y_i^\tau = \frac{\exp(z_i^t/\tau)}{\sum_j \exp(z_j^t/\tau)}, \tag{2}$$

where z_i^t is the output logit of the i th class (i.e., match or unmatch) from the virtual teacher. The temperature τ controls the softening strength. A higher τ produces a smoother probability distribution over these two classes. STEAM formulates its total training objectives as two parts—cross-entropy loss L_{ce} and soft training loss L_{st}:

$$L_{total} = L_{ce} + \lambda L_{st}, \tag{3}$$

The former L_{ce} is computed using the same method as equation (1). And λ controls the weight of soft training loss L_{st}. As for the latter, the soft training loss L_{st} is computed as follows:

$$L_{st} = \tau^2 \mathbb{E}[-y_i^\tau \log p_i^\tau], \tag{4}$$

where p_i^τ is the softened output probability of EM model in the current iteration (i.e., student), which is computed using the same method in equation (2) with equal temperature τ:

$$p_i^\tau = \frac{\exp(z_i^s/\tau)}{\sum_j \exp(z_j^s/\tau)}, \tag{5}$$

where z_i^s is the student's output logit on the i th class. The soft training loss L_{st} is scaled with the parameter τ^2 since the magnitudes of the gradients produced by the soft labels scale as $1/\tau^2$. This enables the contribution of onehot labels y_i and soft labels y_i^τ to remain roughly unchanged when changing the temperature τ (Hinton, Vinyals, and Dean 2015). The hyperparameter λ controls the scale of soft training loss L_{st} in the total loss L_{total}.

The above method introduces the soft supervised training to impose label-wise regularization, which helps a lot in balancing the bias and variance and improving the generalization ability of EM model. The hyperparameter λ controls the regularization strength, balancing the bias and variance, that a lower λ leads to a higher variance, and vice versa. To go further, we additionally design a more flexible weighting method that the weight of soft training loss L_{st} are assigned adaptively.

Soft Training Loss Weighting. The sample-wise soft weighting (SSW) method is motivated by the fact that the bias and variance vary across training samples and change dynamically during training iterations (Zhou and Song 2021). The hard-to-fit samples produce overly softened labels and imposes excessive regularization for EM model, thus a lower weight should be assigned for the soft training loss L_{st} to alleviate this. To achieve it, STEAM framework quantifies the bias and variance from each training sample and adaptively reduces the weight of soft training loss L_{st} for those hard-to-fit samples.

For the input pair x in the training set D, we define the output under original supervised training as p_{ce} and the output under soft supervised training as p_{st}. Then, we can decompose the cross entropy loss L_{ce} and soft training loss L_{st} as follows (Heskes 1998; Zhou and Song 2021):

$$L_{ce} = \mathbb{E}_x[-y_i \log y_i] + D_{kl}(y_i, \bar{p}_{ce}) + \mathbb{E}_D[D_{kl}(\bar{p}_{ce}, p_{ce})], \tag{6}$$

$$L_{st} = \mathbb{E}_x[-y_i \log y_i] + D_{kl}(y_i, \bar{p}_{ce}) + \mathbb{E}_x[y_i \log(\frac{\bar{p}_{ce}}{\bar{p}_{st}})] + \mathbb{E}_{D,\tau}[D_{kl}(\bar{p}_{st}, p_{st})], \tag{7}$$

where $\bar{p}_{ce}$ and $\bar{p}_{st}$ refer to the averages of p_{ce} and p_{st}, and $D_{kl}(\cdot|\cdot)$ is the Kullback-Leibler divergence. Then the soft training loss L_{st} can be rewritten as follows:

$$\begin{aligned} L_{st} &= L_{ce} + L_{st} - L_{ce} \\ &= L_{ce} + \mathbb{E}_x[y_i \log(\frac{\bar{p}_{ce}}{\bar{p}_{st}})] + \\ &\quad \mathbb{E}_{D,\tau}[D_{kl}(\bar{p}_{st}, p_{st})] - \mathbb{E}_D[D_{kl}(\bar{p}_{ce}, p_{ce})]. \end{aligned} \tag{8}$$

It can be easily known that L_{ce} leads to bias reduction that the EM model's average output $\bar{p}_{ce}$ converges to corresponding onehot labels y_i, and $\bar{p}_{st}$ converges to soft labels y_i^τ. Thus we can know, $\bar{p}_{ce}$ is closer to the ground truth labels y_i than $\bar{p}_{st}$, which means $\mathbb{E}_x[y \log(\frac{\bar{p}_{ce}}{\bar{p}_{st}})] \geq 0$. There is an assumption that the variance of soft supervised training is smaller than original supervised training under onehot labels, thus $\mathbb{E}_{D,\tau}[D_{kl}(\bar{p}_{st}, p_{st})] - \mathbb{E}_D[D_{kl}(\bar{p}_{ce}, p_{ce})] \leq 0$. Therefore, $L_{st} - L_{ce}$ can be treated as an additional adversarial item which causes that the bias increases by $\mathbb{E}_x[y_i \log(\frac{\bar{p}_{ce}}{\bar{p}_{st}})]$ and the variance decreases by $\mathbb{E}_D[D_{kl}(\bar{p}_{ce}, p_{ce})] - \mathbb{E}_{D,\tau}[D_{kl}(\bar{p}_{st}, p_{st})]$. Accordingly, for

soft training loss L_{st}, the bias-variance tradeoff reflects the change of L_{ce} and $L_{st} - L_{ce}$. If the gradient $a = \frac{\partial L_{ce}}{\partial z_i}$ lower than $b = \frac{\partial (L_{st} - L_{ce})}{\partial z_i}$, the variance dominates the overall optimization of total loss L_{total}. We call them hard-to-fit samples, which imposes too much label-wise regularization via soft training loss L_{st}, and confuses a lot to the optimization direction. Thus, we should assign lower weights of L_{st} for these samples, and vice versa.

To make the SSW method independent of temperature τ, we set $\tau = 1$, so that $a = p^s_{i,1} - y_i$, and $b = y_i - p^t_{i,1}$, where $p^s_{i,1}$ and $p^t_{i,1}$ refer to the output of the student and corresponding virtual teacher. Then the comparison between a and b has been converted into the comparison between $p^s_{i,1}$ and $p^t_{i,1}$. If the student EM model performs better on a training sample than the corresponding virtual teacher which means $p^s_{i,1} > p^t_{i,1}$, a smaller weight should be assigned to the soft training loss L_{st} of this training sample. The final total loss L_{total} can be formulated as follows:

$$\begin{aligned} L_{total} &= L_{ce} + \lambda \alpha L_{st} \\ &= L_{ce} + \lambda (1 - \exp(-\frac{\log p^s_{i,1}}{\log p^t_{i,1}})) L_{st} \\ &= L_{ce} + \lambda (1 - \exp(-\frac{\log L^s_{ce}}{\log L^t_{ce}})) L_{st}, \end{aligned} \tag{9}$$

where α is the additional sample-wise soft weight of L_{st} that changes dynamically across training samples and iterations. And L^s_{ce} and L^t_{ce} refer to the standard cross-entropy loss of student EM model and its virtual teacher EM model respectively.

Experiments

In this section, we evaluate our proposed STEAM framework on two open EM benchmarks (eight datasets) to demonstrate its performance against existing SOTA methods.

Benchmarks and Metrics

We evaluate STEAM framework on WDC benchmark (Primpeli, Peeters, and Bizer 2019) and DeepMatcher benchmark (Mudgal et al. 2018). The summaries of them are shown in Table 2 and Table 3.

For WDC benchmark, we split the training/validation sets with the ratio of 4:1, which is the same as Ditto (Li et al. 2020b). We evaluate STEAM on all the WDC subsets—computers, cameras, watches, and shoes. All the WDC subsets contain four attributes—*title*, *description*, *brand*, and *specTableContent*. For the sake of fairness, we only use the *title* attribute following Ditto (Li et al. 2020b), so that all the results can be fairly compared. We evaluate STEAM on all the versions (i.e., Small, Medium, Large, and xLarge) and comprehensively compare the results with Hybrid (Mudgal et al. 2018) and Ditto (Li et al. 2020b) to show the EM performance under different training scales.

For DeepMatcher benchmark, we split the training/validation/testing sets with the ratio of 3:1:1, which is the same as existing methods like DeepMatcher (Mudgal et al. 2018), Ditto (Li et al. 2020b), and HierGAT (Yao et al. 2022). In all the subsets of this benchmark, we choose the challenging and representative datasets—Amazon-Google (A-G), Walmart-Amazon (W-A) with its dirty version W-A*, and Abt-Buy (A-B). The A-G dataset contains three attributes—*title*, *manufacturer*, and *price*. And W-A and W-A* contain five attributes—*title*, *category*, *brand*, *modelno*, and *price*. The W-A* dataset is manually generated by randomly injecting other attribute values into *title* to simulate a common kind of dirty data seen in the wild. And the A-B dataset contains three attributes—*name*, *description*, and *price*, at least one of which is long text. These datasets are representative enough to evaluate STEAM in common EM scenarios.

Following previous researches, we report the F1 score to evaluate the EM performance of proposed STEAM. All the compared results are obtained from the original paper.

	Small (1/20)	Medium (1/8)	Large (1/2)	xLarge (1/1)
Computers	2,834	8,094	33,359	68,461
Cameras	1,886	5,255	20,036	42,277
Watches	2,255	6,413	27,027	61,569
Shoes	2,063	5,805	22,989	42,429

Table 2: Statistics of WDC benchmark.

Dataset	Domain	Type	#Rec	#Match
A-G	Software	Structured	11,460	1,167
W-A	Electronics	Structured	10,242	962
W-A*	Electronics	Dirty	10,242	962
A-B	Product	Textual	9,575	1,028

Table 3: Statistics of DeepMatcher benchmark.

Implementation and Training Details

We implement STEAM framework with PyTorch and HuggingFace libraries. All the BERT-like PLMs (e.g., BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019), DistilBERT (Sanh et al. 2019)) are supported in STEAM framework. We adopt RoBERTa-base as the default PLM and report its results in all the benchmarks. The size of mini-batch is 64, and the maximum length of the input is limited to 128 (256 for A-B dataset) that any tokens beyond that are truncated. We train STEAM using Adam optimizer and the learning rate is 3e-5. The maximum training epoch is 50, and we adopt the early stop strategy with the patients varying from 5 to 15 according to the datasets. We adopt data augmentation (e.g., dropping token, swapping record, and swapping attribute values) and dropout strategy with the probability of 0.5. For the part of soft supervised training, we set the temperature $\tau = 10$ to automatically generate the soft labels. And the hyperparameter λ is set to be 1.

Comparisons of Effectiveness and Label Efficiency

Table 4 and 5 show the experimental results of our proposed STEAM framework compared with existing baselines. As can be seen, for both WDC and DeepMatcher benchmarks, STEAM achieves the leading F1 score and becomes

	xLarge (1/1)			Large (1/2)			Medium (1/8)			Small (1/20)		
	Hybrid	Ditto	STEAM	Hybrid	Ditto	STEAM	Hybrid	Ditto	STEAM	Hybrid	Ditto	STEAM
Computers	90.80	95.45	96.46	89.55	91.70	94.82	77.82	88.62	93.20	70.55	80.76	89.62
$\Delta F1$			+1.01			+3.12			+4.58			+8.86
Cameras	89.21	93.78	94.77	87.19	91.23	94.48	76.53	88.09	92.00	68.59	80.89	88.06
$\Delta F1$			+0.99			+3.25			+3.91			+7.17
Watches	93.45	96.53	96.83	91.28	95.69	96.67	79.31	91.12	94.53	66.32	85.12	93.86
$\Delta F1$			+0.30			+0.98			+3.41			+8.74
Shoes	92.61	90.11	92.99	90.38	88.07	90.47	79.48	82.66	85.53	73.86	75.89	82.84
$\Delta F1$			+2.88			+2.40			+2.87			+6.95
$\Delta F1_{ave}$			+1.30			+2.44			+3.69			+7.93

Table 4: F1 scores on the WDC benchmark. We calculate $\Delta F1$ and $\Delta F1_{ave}$ against Ditto.

	A-G	W-A	W-A*	A-B
Magellan (Konda et al. 2016)	49.1	71.9	37.4	43.6
RNN (Mudgal et al. 2018)	59.9	67.6	39.6	39.4
Attention (Mudgal et al. 2018)	61.1	50.0	53.8	56.8
Hybrid (Mudgal et al. 2018)	69.1	66.9	46.0	62.8
MPM (Fu et al. 2019)	70.7	73.6	-	-
Seq2SeqM (Nie et al. 2019)	-	78.2	68.3	-
GraphER (Li et al. 2020a)	68.08	-	-	-
MCA (Zhang et al. 2020)	70.3	73.4	-	69.4
HierMatcher (Fu et al. 2020)	74.9	81.6	68.5	-
BERT-ER (Li et al. 2021a)	75.3	-	-	-
Ditto (Li et al. 2020b)	75.63	86.97	85.69	89.33
HierGAT (Yao et al. 2022)	76.4	88.2	86.3	89.8
STEAM-BASE	73.85	84.32	83.33	88.83
STEAM-FSW	75.10	87.07	85.42	90.27
STEAM-SSW	77.91	88.41	87.23	91.09

Table 5: F1 scores on the DeepMatcher benchmark.

the SOTA solution in these two benchmarks. The details are as follows.

In Table 4, we show STEAM on WDC benchmark with the compared DL-based EM baseline (i.e., Hybrid (Mudgal et al. 2018)) and PLM-based EM baseline (i.e., Ditto (Li et al. 2020b)) and calculate the $\Delta F1$ and $\Delta F1_{ave}$ against Ditto. STEAM obtains the leading performances in all the subsets of WDC benchmark, and it is worth mentioning that the average improvement $\Delta F1_{ave}$ are 1.30, 2.44, 3.69, and 7.93, which effectively demonstrates the effectiveness and label efficiency of STEAM framework. Figure 4 further reflects this, when using only 1/20 training data (WDC-Small), STEAM is still competitive with Ditto using 1/8 training data (WDC-Medium), and this happens in other cross-level comparisons (i.e., Medium vs Large, Large vs xLarge). These exciting results effectively show the great improvement of our proposed STEAM framework. Despite using fewer training samples, our STEAM can still be competitive to Ditto.

In Table 5, we consider more compared baselines, and split them into ML-based baseline (i.e., Magellan), DL-based (non PLM) baselines (e.g., Hybrid), and PLM-based baselines (e.g., Ditto). For all the structured scenario (i.e., A-G and W-A), dirty scenario (i.e., W-A*), and textual scenario (i.e., A-B), STEAM outperforms all the compared baselines.

Ablation Study

We design three versions of STEAM to comprehensively analyze the role of pretrained language model and soft supervised training. STEAM-BASE refers to the version that directly finetunes the PLM on onehot labels without any optimization. STEAM-FSW refers to the version that adopts fixed soft weighting (FSW) method to weight the soft training loss L_{st}. And STEAM-SSW is the complete version that adopts sample-wise soft weighting (SSW) method to compute the weight of soft training loss L_{st} adaptively. As is shown in Table 5, the basic version STEAM-BASE shows superior performance against other non PLM-based methods, this demonstrates that PLMs are powerful encoders for relational records and have become the de facto best solution in deep EM tasks. And STEAM-FSW is competitive against baselines, which demonstrates that the label-wise regularization can also improve the generalization ability of EM model like existing model-oriented and data-oriented approaches. And the complete version STEAM-SSW achieves leading performance on all the datasets. The results show that the soft supervised training with sample-wise soft weighting can effectively improve the generalization ability of EM model to obtain the promising matching results for unseen record pairs.

Detail Analysis

In this section, we conduct the deep insight of our proposed STEAM. To control the variables, we take our basic version STEAM-BASE as a baseline, and analyze the results of STEAM-SSW with it to explain the obtained improvements in soft supervised training.

Tradeoff between Bias and Variance. Figure 5 (a) shows the tradeoff between bias and variance in WDC-Computers dataset, where the bias is computed as

$$bias^2(x) = (\bar{f}(x) - y)^2, \tag{10}$$

and the variance is computed as

$$var(x) = \mathbb{E}_D[(f(x) - \bar{f}(x))^2]. \tag{11}$$

With the reduction of bias (deep and light green lines), the outputs of STEAM converge to ground truth. What we need to pay more attention is the change of variance (deep and

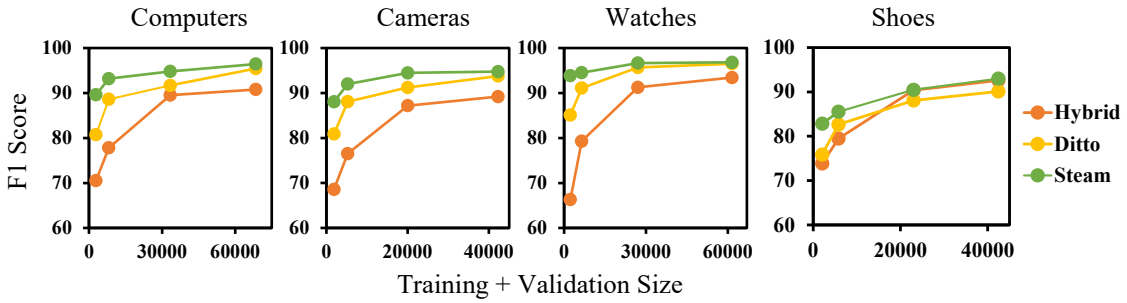
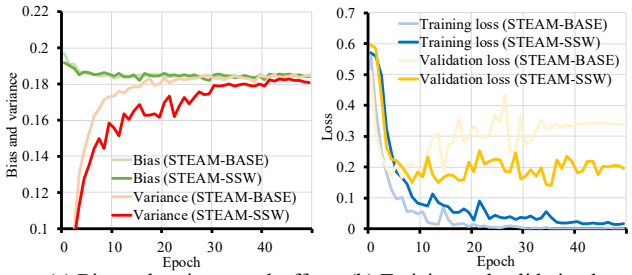


Figure 4: Label efficiency of STEAM against Hybrid and Ditto. The training + validation size changes as Small (1/20), Medium (1/8), Large (1/2), and xLarge (1/1).



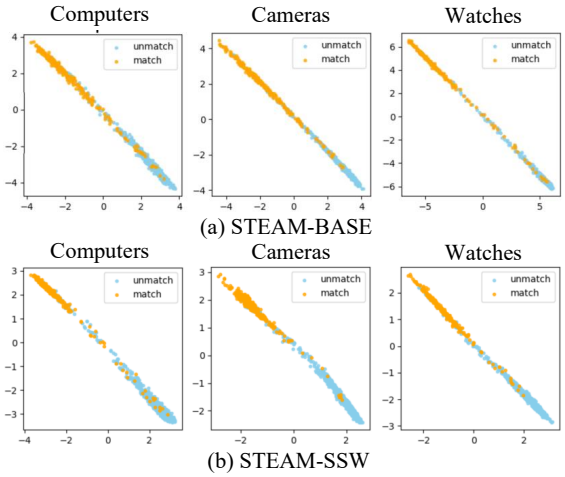
(a) Bias and variance tradeoff (b) Training and validation loss

Figure 5: (a) Bias and variance tradeoff of STEAM-SSW and STEAM-BASE. (b) Training and validation loss of STEAM-SSW and STEAM-BASE.

light red lines). For STEAM-BASE, the unregularized supervision signals from onehot labels make the obtained EM model overfit the training samples, causing the variance uncontrolled (light red line). As a contrast, the variance of STEAM-SSW is better controlled (deep red line) via the soft supervised training. It shows that the soft supervised training of STEAM-SSW can effectively control the bias and variance simultaneously, which helps a lot in improving the generalization of EM model.

Changes of Training and Validation Loss. Figure 5 (b) shows the training and validation loss in WDC-Computers dataset. As can be seen, at the beginning of the training iterations, STEAM-BASE converges quickly (light blue line). However, the unregularized sharp supervision signals make the EM model overfit the training samples and harm the effectiveness on validation samples (light yellow line). As a comparison, despite the training loss of STEAM-SSW (deep blue line) is higher than STEAM-BASE (light blue line), the validation loss of STEAM-SSW (deep yellow line) is much lower. This demonstrates that the soft supervised training method effectively prevents STEAM-SSW from overfitting to the training samples and improves the generalization when facing unseen samples.

Visualization. Figure 6 shows the visualization of our proposed STEAM-BASE and STEAM-SSW on WDC datasets (Computers, Cameras, and Watches). As can be seen, the



(a) STEAM-BASE

(b) STEAM-SSW

Figure 6: Visualization of STEAM-BASE and STEAM-SSW.

clusters of STEAM-BASE are much thinner and longer, meanwhile the boundary is more blurred. This leads to a worse EM performance, since more samples are difficult to be distinguished into matching or unmatching. As a contrast, the clusters of STEAM-SSW are much tighter and the boundary is more distinct, which leads to a better matching quality.

Conclusions

In this paper, we propose a novel STEAM framework to improve the generalization of EM model. It leverages the teacher-free knowledge distillation to automatically generate soft targets to conduct a label-wise regularization. By taking the softened outputs from the previous iteration's virtual teacher as additional regularizers to train the current iteration's EM model, STEAM effectively calibrates the EM model and balances the bias and variance without additional computational cost. Extensive experiments on two open EM benchmarks (eight datasets) show that our proposed STEAM framework effectively improves the performance of obtained EM model on unseen testing data in terms of effectiveness and label efficiency.

Acknowledgments

This work is supported by National Natural Science Foundation of China (62172082, 62072084, 62072086), and the Fundamental Research Funds for the Central Universities (N2116008).

References

Brunner, U.; and Stockinger, K. 2020. Entity matching with transformer architectures-a step forward in data integration. In *23rd International Conference on Extending Database Technology, Copenhagen, 30 March-2 April 2020*. OpenProceedings.

Cappuzzo, R.; Papotti, P.; and Thirumuruganathan, S. 2020. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 1335–1349.

Chen, R.; Shen, Y.; and Zhang, D. 2021. GNEM: a generic one-to-set neural entity matching framework. In *Proceedings of the Web Conference 2021*, 1686–1694.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186.

Dou, W.; Shen, D.; Nie, T.; Kou, Y.; Sun, C.; Cui, H.; and Yu, G. 2022. Empowering Transformer with Hybrid Matching Knowledge for Entity Matching. In *International Conference on Database Systems for Advanced Applications*, 52–67. Springer.

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

Fu, C.; Han, X.; He, J.; and 0001, L. S. 2020. Hierarchical Matching Network for Heterogeneous Entity Resolution. In *IJCAI*, 3665–3671.

Fu, C.; Han, X.; Sun, L.; Chen, B.; Zhang, W.; Wu, S.; and Kong, H. 2019. End-to-end multi-perspective matching for entity resolution. In *IJCAI*.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. Deep Learning. https://www.deeplearningbook.org/. Accessed: 2022-6-1.

Heskes, T. 1998. Bias/variance decompositions for likelihood-based estimators. *Neural Computation*, 10(6): 1425–1433.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the Knowledge in a Neural Network. *stat*, 1050: 9.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Konda, P.; Das, S.; GC, P. S.; 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. *Proceedings of the VLDB Endowment*, 9(12).

Li, B.; Miao, Y.; Wang, Y.; Sun, Y.; and Wang, W. 2021a. Improving the efficiency and effectiveness for bert-based entity resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 13226–13233.

Li, B.; Wang, W.; Sun, Y.; Zhang, L.; Ali, M. A.; and Wang, Y. 2020a. GraphER: token-centric entity resolution with graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8172–8179.

Li, Y.; Li, J.; Suhara, Y.; Doan, A.; and Tan, W.-C. 2020b. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1): 50–60.

Li, Y.; Wang, X.; Miao, Z.; and Tan, W.-C. 2021b. Data augmentation for ml-driven data preparation and integration. *Proceedings of the VLDB Endowment*, 14(12): 3182–3185.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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 *Proceedings of the 2018 International Conference on Management of Data*, 19–34.

Nie, H.; Han, X.; He, B.; Sun, L.; Chen, B.; Zhang, W.; Wu, S.; and Kong, H. 2019. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 629–638.

Peeters, R.; and Bizer, C. 2021. Dual-objective fine-tuning of BERT for entity matching. *Proceedings of the VLDB Endowment*, 14(10): 1913–1921.

Primpeli, A.; Peeters, R.; and Bizer, C. 2019. The WDC training dataset and gold standard for large-scale product matching. In *Companion Proceedings of The 2019 World Wide Web Conference*, 381–386.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Yao, D.; Gu, Y.; Cong, G.; Jin, H.; and Lv, X. 2022. Entity Resolution with Hierarchical Graph Attention Networks. In *Proceedings of the 2022 International Conference on Management of Data*, 429–442.

Yuan, L.; Tay, F. E.; Li, G.; Wang, T.; and Feng, J. 2020. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3903–3911.

Zhang, D.; Nie, Y.; Wu, S.; Shen, Y.; and Tan, K.-L. 2020. Multi-context attention for entity matching. In *Proceedings of The Web Conference 2020*, 2634–2640.

Zhou, H.; and Song, L. 2021. Rethinking Soft Labels for Knowledge Distillation: A Bias–Variance Tradeoff Perspective. In *Proceedings of International Conference on Learning Representations (ICLR)*.