Deep Transfer Learning for Multi-source Entity Linkage via Domain Adaptation

Di Jin[1], Bunyamin Sisman[2], Hao Wei[2], Xin Luna Dong[2], and Danai Koutra[1]

[1]University of Michigan, Ann Arbor
[2]Product Graph, Amazon Research
{dijin,dkoutra}@umich.edu {bunyamis,wehao,lunadong}@amazon.com

Abstract

Multi-source entity linkage focuses on integrating knowledge from multiple sources by linking the records that represent the same real world entity. Existing pipelines mainly depend on supervised learning that requires abundant amounts of training data. However, collecting well-labeled training data from many sources is expensive and the trained models can easily overfit to specific data sources and fail to generalize to new sources due to significant differences in data/label distributions. To address these challenges, we present ADAMEL, a deep transfer learning framework that learns generic high-level knowledge to perform multi-source entity linkage. ADAMEL models attribute importance through an attribute-level self-attention mechanism, and leverages the massive unlabeled data from new data sources through domain adaptation to make it generic and data-source agnostic. Extensive experiments show that our framework achieves state-of-the-art results with $2.67\% - 11.24\%$ improvements over methods based on supervised learning. Also, it is more stable in handling different data sources with at least 3x less runtime.

1 Introduction

Entity linkage (EL), also known as entity resolution, entity matching, is a fundamental task in data mining, database, and knowledge integration with wide applications, including deduplication, data cleaning, user stitching, and more. The key idea is to identify records across different data sources (*e.g.*, databases, websites, knowledge base, etc.) that represent the *same* real-world entity. As newly-generated data surge over time, accurately consolidating the same entities across semi-structured web sources becomes increasingly important, especially in areas such as knowledge base establishment [7, 12] and personalization [15]. Methods for solving the entity linkage problem across data sources include rule reasoning [9, 28], computation of similarity between attributes or schemas [2], and active learning [27]. In particular, recent deep learning approaches that are based on heterogeneous schema matching or word matching [24, 25, 22] have been widely stuied. Their promising performance mainly comes from the sophisticated word-level operations such as RNN and Attention [24, 11] to represent token sequences under attributes as the summarization, or the usage of pretrained language models [19] to better learn the word semantics. However, all the above approaches implicitly assume that the "matching/non-matching" info for training records is available (*e.g.*, the music records in source 1 and source 2 shown in the two blue tables of Fig. 1) and can be queried through the learning process, which does not always hold in practice. In real-world knowledge integration scenarios, new data come incrementally and can be well-labeled (*e.g.*, through manual confirmation) or unlabeled. As the example shown in Fig. 1, a model trained on the high-quality labeled data (blue tables) would fail to generalize to the new data sources (red tables) with missing and different attribute values (*i.e.*, "Artist"), as well as new or rarely-seen attributes (*i.e.*, "Gender").

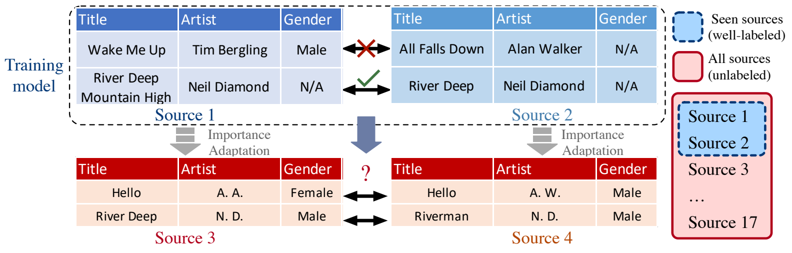


Figure 1: Our proposed framework, AdaMEL automatically learns the attribute importance that adapts to the massive unlabeled data from different sources (*i.e.*, red tables) during training, and then uses it as the transferable knowledge to perform matching.

Motivated by real-world knowledge integration settings, we consider three key challenges: (**C1**) missing attribute values from unseen data sources; (**C2**) new attributes from unseen data sources; and (**C3**) different value distribution in unseen data sources. Based on these challenges, we seek to tackle the following **MEL (multi-source entity linkage) problem**: Given labeled data from a limited set of sources, *what* knowledge can be learned and *how* can it be transferred to automatically handle multiple unseen data sources with different value distribution, missing values and new attributes?

To solve this task, we propose AdaMEL, a transfer learning framework that leverages both the labeled and massive unlabeled data to train the model for multi-source entity linkage while addressing the aforementioned challenges (**C1**-**C3**). We define the attribute importance in entity linkage as the high-level transferable knowledge and automatically learn it through a proposed attribute-level attention mechanism (*what* to transfer). More importantly, AdaMEL adopts domain adaptation (DA) to jointly update the attention scores for attributes in both the seen and unseen data as the basis for entity linkage (*how* to handle multiple sources). Unlike most transfer learning frameworks that fine-tune the pretrained models to handle new data, AdaMEL uses well-labeled data sampled from limited sources as well as unlabeled data sampled from a wide range of sources to jointly update the attribute importance during training. It is worth noting that AdaMEL highlights the role of "important" entity attributes in determining entity matching across data sources, and assumes that the similarity/dissimilarity of important attributes has higher impacts than unimportant attributes. While the widely-adopted NLP-based attribute summarization in existing works could accurately capture the word-level semantics using pretrained language models or domain knowledge, we claim that the impacts of word-level similarity/dissimilarity under some attributes are limited and even harmful in model performance if those attributes are not important. Experimental results show that the MEL performance is comparable and better by using fewer important attributes only, and AdaMEL could achieve consistency in terms of performance and reduces expensive model retraining time.

2 Related Work

Entity Linkage (EL). Early works in EL [12, 21, 6] are based on the similarity between entity attributes [9, 6] via resolving the data conflicts [7], linking relevant attributes via semantic matching or rule reasoning [28]. Blocking or hashing are normally applied to merge the candidate entities [4]. The major drawback is the dependence on prior knowledge as the useful attributes are normally manually-selected.Recently, EL models based on deep neural networks [24, 16] have been widely studied due to their capability in automatically deriving latent features and promising results in fields such as CV and NLP [1, 23, 10]. For example, DeepER [16] and DeepMatcher [24] propose to leverage RNN to compose the pre-trained word embeddings of tokens within all attribute values, and use them as features to conduct EL. There are also works that formulate entity linkage across different data sources as heterogeneous entity matching [22, 11, 25].

Transfer Learning. In transfer learning, models are trained on a source domain and applied to a related target domain to handle the same or a different task [26, 13]. The specific transferable knowledge that bridges the source and target domain has significant impact to model performance [33]. A popular approach is to adapt the pre-trained model for the new task through fine-tuning [19], or by adding new functions to tasks such as object detection [14]. A recent work in EL, Auto-EM [34] falls into this category, but it only applies to single data source. A specific type of transductive transfer learning that is most relevant to our work is known as *Domain Adaptation*, where the source and target domain share the same feature space with different distributions [29], and machine learning models are trained on the same task [32]. Many well-designed algorithms propose to map the original feature spaces to a shared latent feature space between domains [8, 3].

3 Preliminaries

Problem Definition An entity record is collected from a specific data source such as a website or a database, and is identified by its attributes. For example, a song record $r =$ ("Sweet Caroline", "Neil Diamond", "USA") is specified by the attributes $\mathcal{A} = \{\texttt{title, artist, country}\}$. In this paper, we conduct analysis based on entity pairs (r, r') instead of individual entity records. We now define the MEL problem, which is related to the heterogeneous entity matching[*] [25, 11]. Symbols and notions used in this paper are listed in Table 3 of Section 7.1 in the appendix.

Problem 1 (MEL: Multi-source Entity Linkage) *Given the labeled entity pairs $\{(r, r')\}_{seen}$ from a limited set of data sources $\mathcal{S}$ where each entity record r is associated with attributes $\mathcal{A}$, and previously unseen pairs $\{(r, r')\}_{unseen}$ from the new data sources $\mathcal{S}'$ with attributes $\mathcal{A}'$, Multi-source entity linkage aims to predict if $\{(r, r')\}_{unseen}$ represents the same real-world entity, where $(r, r')_{unseen} \in (\mathcal{S} \times \mathcal{S}') \cup (\mathcal{S}' \times \mathcal{S}')$, $|\mathcal{S}'| > |\mathcal{S}|$ (assuming that there are more unseen entities than seen). Since $\mathcal{S} \neq \mathcal{S}'$, certain attributes in $\mathcal{A}'$ could be missing (**C1**), new (**C2**), or associated with different values (**C3**), and thus $\mathcal{A} \neq \mathcal{A}'$.*

In Problem 1, the linkage is conducted on entity pairs sampled from a wider range of data sources than the labeled data in model training (ten or hundred orders of magnitude more in reality). Plus, the new data sources contain a rarely seen or unseen attribute ("Gender"). This issue can be addressed by the aligned ontology such as $\mathcal{A} \cup \mathcal{A}'$ with blank "dummy" attributes. Based on our definition, a solution to MEL should be able to (**G1**) make use of the massive unlabeled data from new sources, and (**G2**) further improve the linkage performance by leveraging a few labeled record pairs from new sources if available (*i.e.*, an additional support set). At a high level, we introduce an effective domain adaptation-based solution. Before presenting our framework, we discuss the necessary terminology.

Definition 1 (Source & target domain) *The source domain $\mathcal{D}_S$ refers to a set of labeled entity pairs $\{(r, r')\}$ sampled from limited data sources that the model is trained on. The target domain $\mathcal{D}_T$ refers to the set of unlabeled pairs where each pair has at least one entity sampled from the data sources unseen in $\mathcal{D}_S$.*

For clarity, we use the superscript * to indicate the data source(s) of a record/domain. Following Definition 1, the seen and unseen set of data sources in Problem 1 are formulated as $\mathcal{S} = \mathcal{D}_S^*$ and $\mathcal{S}' = \mathcal{D}_T^*$. An entity pair from $\mathcal{D}_T$ could either contain one entity sampled from the seen data sources in $\mathcal{D}_S^*$ and the other one from the unseen, *i.e.*, $(r, r')_T \in \mathcal{D}_S^* \times \mathcal{D}_T^*$, or it has both entities sampled from the completely unseen data sources, *i.e.*, $(r, r')_T \in \mathcal{D}_T^* \times \mathcal{D}_T^*$. In both cases, achieving **G1** requires data in $\mathcal{D}_T$. To achieve **G2**, we introduce the support set, which corresponds to the real-world scenario that a few newly incoming entity pairs are well-labeled (*e.g.*, on-the-fly human annotation).

Definition 2 (Support set) *The support set $\mathcal{S}_U$ refers to a small set of labeled entity pairs sampled from the same set of data sources as $\mathcal{D}_T^*$. It has at least one data source that is not contained in $\mathcal{D}_S^*$.*

4 Proposed framework

4.1 Formulation

In transfer learning, the generic transferable knowledge $\mathcal{K}$ is key to adapt the model trained on the source domain to the target domain. We denote our domain adaptation solution to MEL as the following binary classification task.

$$y = M(\mathcal{K}, (r, r')) \in \{0, 1\} \tag{1}$$

where M represents the deep model that generates the binary prediction y for the entity pair $(r, r') \in \mathcal{D}_T$, where 1 and 0 indicate the matching and non-matching, respectively. As mentioned in Problem Statement, the key difference between $\mathcal{D}_S$ and $\mathcal{D}_T$ lies in the difference in data sources, therefore $\mathcal{K}$ should be data-source agnostic to address **(C1)-(C3)**. To ensure $\mathcal{D}_T$ shares the same feature space as $\mathcal{D}_S$ (the prerequisite for domain adaptation), ADAMEL first aligns the ontology so that data sources $\mathcal{D}_S^*$ and $\mathcal{D}_T^*$ share the same attribute schema, but the attribute values (word tokens)

[*]In MEL, the entities come from different data sources, and thus there may be new or missing attributes. On the other hand, in heterogeneous entity matching, the schemas are heterogeneous (i.e., they have different attributes, which may not be aligned) and the entities do not necessarily come from different data sources.

can vary significantly. By doing so, entity records reveal the following properties that correspond to the aforementioned challenges: (**C1**) entity records in the source/target domain contain missing values, *i.e.*, $r[A] =$"" (empty string) for $r \in \mathcal{D}_S \cup \mathcal{D}_T$, (**C2**) certain attribute values are completely missing for records in $\mathcal{D}_S$, *i.e.*, $r[A_j] =$"" for $r \in \mathcal{D}_S$, but not in $\mathcal{D}_T$, and (**C3**) rich texts under some attributes in $\mathcal{D}_S$ but sparse in $\mathcal{D}_T$ or vice versa.

4.2 Feature Representation

Given entity pairs (r, r') with the aligned attributes $\mathcal{A}$, ADAMEL first parses each attribute A into 2 contrastive relational features, which are word tokens shared by r and r', and word tokens that only appear in one record but not the other. This is because the similarity or uniqueness of attribute between r and r' gives independent and complementary evidence for linkage [31]. Taking the attribute $A =$"Gender" as an example, a pair of music recordings sharing artist gender is a weak identifier for matching but it is strong for non-matching. In addition, looking into both the similarity and uniqueness in attribute A between entities would enrich the feature space and facilitate training the deep model. We describe the 2 contrastive relational features of an attribute A as follows.

$$sim(A) = \{w| \text{ for } w \in \{r[A] \cap r'[A]\}\}, uni(A) = \{w| \text{ for } w \in \{r[A] \cup r'[A] - r[A] \cap r'[A]\}\} \quad (2)$$

where w is the word token in attribute $r[A]$. For clarity, we uniformly denote shared/unique tokens $sim(A)/uni(A)$ as "features" that contribute independently to entity linkage. Clearly, there are $F = 2|\mathcal{A}|$ features for a pair of entities. To summarize the feature representation, ADAMEL simply sums up the embeddings of the cropped word tokens [17, 30, 24] without using more sophisticated operations. The embeddings of word tokens can be obtained using any pretraining language models such as BERT [19] or Fasttext [17]. For clarity, we use i as the index of entity pairs and j as the index of features. Thus, the token embedding vector of an entity pair (r, r') is denoted as:

$$\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_F] = [\text{emb}(sim(A_j)), \text{emb}(uni(A_j))] \text{ for } j = 1, \cdots, |\mathcal{A}| \quad (3)$$

Thus, we have $F = 2|\mathcal{A}|$ textual embedding features for an entity pair (r, r'). ADAMEL leverages per-feature non-linear affine transformation to project word embeddings for latent feature $\mathbf{x}$:

$$\mathbf{x} = [\sigma(\mathbf{V}_j \mathbf{h}_j + \mathbf{b}_j)] \text{ for } j = 1, \cdots, F \quad (4)$$

where $\mathbf{V}_j^{H \times D}$ and $\mathbf{b}_j^H$ are the learnable weight matrix and bias vector, respectively. σ denotes the non-linear activation function (*e.g.*, Relu). Thus, Equation (1) can be rewritten as: $y = M(\mathcal{K}, \mathbf{x}) \in \{0, 1\}$. Next we discuss how ADAMEL learns feature importance [†] as the transferable knowledge $\mathcal{K}$.

4.3 Feature Attention Embedding

Given a pair of entities denoted through F features, ADAMEL defines the attention coefficient of feature j as: $e_j = a(\mathbf{W}\mathbf{x}_j)$, where $\mathbf{x}_j$ is the H-dimensional representation of latent feature j, $\mathbf{W}^{H' \times H}$ is a shared linear transformation, and a represents the attention mechanism $\mathbb{R}^{H'} \to \mathbb{R}$, as a single-layer neural network (parameterized with $\mathbf{a}$). ADAMEL allows each feature to attend to the label y independently and computes coefficients using the softmax function such that the normalized scores are comparable across all features. Equation (5) computes the attention score of feature j:

$$g(\mathbf{x}_j) = \text{softmax}_j(e_j) = \frac{\exp\left(\mathbf{a}^T \tanh\left(\mathbf{W}\mathbf{x}_j\right)\right)}{\sum_{k=1}^{F} \exp\left(\mathbf{a}^T \tanh\left(\mathbf{W}\mathbf{x}_k\right)\right)} \quad (5)$$

Note that Equation (5) only generates the scalar attention score of feature j for an input vector $\mathbf{x}$. To compute the scores of all features, we introduce the attention embedding function f that learns attention scores of all F features as follows.

$$f(\mathbf{x}) = f([\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_F]) = [g(\mathbf{x}_1), g(\mathbf{x}_2), \cdots, g(\mathbf{x}_F)] \quad (6)$$

In Equation (6), all features share the same $\mathbf{W}$ and $\mathbf{a}$ to compute the attention scores. We denote $f(\mathbf{x})_j = g(\mathbf{x}_j)$, and $\sum_{j=1}^{F} f(\mathbf{x})_j = 1$. ADAMEL takes the generated feature importance vector $f(\mathbf{x})$ as the transferable knowledge $\mathcal{K}$ for the entity pair (r, r'), *i.e.*, $\mathcal{K} = f(\mathbf{x})$. In the learning process, ADAMEL feeds the feature representation coupled with its attention score to a 2-layer

[†]In this paper, we compute the feature attention as the transferable knowledge, feature importance.

feed-forward neural network Θ to perform the binary classification task: $\hat{y} = \Theta(\sigma(f(\mathbf{x}) \odot \mathbf{x})) = \Theta([\sigma(g(\mathbf{x}_1) \cdot \mathbf{x}_1), \cdots, \sigma(g(\mathbf{x}_F) \cdot \mathbf{x}_F)])$, where $\odot$ denotes the element-wise multiplication, σ denotes the non-linear activation (*e.g.*, Relu) and $\hat{y}$ denotes the inference score for matching. ADAMEL uses the same attention mechanism to handle all records in the training and leverages the cross-entropy loss to update the shared parameters $\mathbf{W}$, $\mathbf{a}$, as well as the learnable $\mathbf{V}$, $\mathbf{b}$ through back-propagation. The loss is $L_{base} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)$, where y_i denotes the label $\{0, 1\}$. To ensure that all learnable parameters can be updated correctly, ADAMEL initializes the missing attribute values (incurred by challenge **C1, C2**) with a fixed normalized non-zero vector. We name this solution **ADAMEL-*base*** as it learns f through the labeled data in $\mathcal{D}_S$. The attribute importance learned under the supervision of labeled data in $\mathcal{D}_S$ will be carried over to the unseen data sources and may not generalize well as there is always new data from seen or unseen sources with different distributions (**C3**) in MEL. Next we discuss how ADAMEL adopts $\mathcal{D}_T$ to alleviate this issue and make $\mathcal{K}$ data-source agnostic, and describes three variants that leverage domain adaptation to handle different learning scenarios.

4.4 Domain Adaptation-based Variants

Unsupervised Domain Adaptation Our first idea is to adjust the learned attribute importance according to new distribution of unlabeled data. In Equation (6), the attention embedding function f contains the shared attention mechanism a parameterized by weight vector $\mathbf{a}$ and the shared transformation matrix $\mathbf{W}$. It only takes the feature embeddings $\mathbf{x}$ as input to compute the attention scores. Since $\mathbf{W}$ and $\mathbf{a}$ are shared across the input data, the attention score vector $f(\mathbf{x})$ can be seen as projecting the input feature embeddings $\mathbf{x}$ into a hyper-plane that is parameterized by $\mathbf{W}$ and $\mathbf{a}$. Without introducing extra information such as entity pair labeling, we can project data from $\mathcal{D}_T$ into the same space as $\mathcal{D}_S$, and it holds as long as the ontology of the unlabeled data aligns with the labeled data, *i.e.*, identical attribute schema between $\mathcal{D}_S$ and $\mathcal{D}_T$. Therefore, ADAMEL uses the KL divergence to measure the attention score distribution difference between the source and target domain as the regularization term to train the model. The loss is defined as $L_{\text{un}} = (1 - \lambda)L_{\text{base}} + \lambda L_{\text{target}}$, where λ is the hyperparameter that balances between L_{base} and L_{target}. λ also measures the amount of adaptation to the target domain $\mathcal{D}_T$. L_{target} is given as $L_{\text{target}} = \text{KL}(f(\mathbf{x}), \bar{f}(\mathbf{x}'))$, where $\bar{f}(\mathbf{x}')_j = \frac{1}{|\mathcal{D}_T|}\sum_{\mathbf{x}'_i \in \mathcal{D}_T} f(\mathbf{x}'_i)_j$ represents the attention score for feature j averaged over the unlabeled data. $\mathbf{x}$ and $\mathbf{x}'$ denote the feature vector in the source and target domain, respectively, and $f(\mathbf{x}_i)_j$ denotes the importance of the j-th feature in the i-th entity pair. In practice, ADAMEL adopts batch learning to improve the training efficiency, *i.e.*, minimizing the loss per batch instead of iterating through all records in the data. The unlabeled data could also come in batches, which makes $\bar{f}(\mathbf{x}')$ be the attention vector averaged over the batched unlabeled data instead of all in the target domain. By default, the batches are sampled randomly. We name this solution ***AdaMEL-zero*** as it is based on unsupervised domain adaption without using labeled data in $\mathcal{D}_T$ and performs linkage in the zero-shot manner. The detailed architecture and algorithm is given in Section 7.2 of the Appendix.

Semi-supervised Domain Adaptation In practice, a small number of labels may be available for the entity pairs coming from the target domain (*e.g.*, through on-the-fly human annotation). Entity pairs in this support set $\mathcal{S}_U$ are sampled from the wide range of data sources and provide clues about the data characteristics of the target domain. To leverage this set of labeled data (**G2**), ADAMEL updates the attention embedding function f under the supervision of $\mathcal{S}_U$ so that the projected feature attention vectors of entity pairs in $\mathcal{D}_S$ could match to those in $\mathcal{S}_U$. For this purpose, ADAMEL computes the centroid of the positive entity pairs in $\mathcal{D}_S$ as: $\mathbf{c}^{+}_{\mathcal{D}_S} = \frac{1}{|\mathcal{D}_S|}\sum_{(\mathbf{x}^{+}_i, y^{+}_i) \in \mathcal{D}_S} f(\mathbf{x}_i)$. The centroid of the negative pairs can be computed in a similar way with negative samples. Intuitively, entity pairs from the data sources unseen in $\mathcal{D}^{*}_{S}$ are more important in adaptation than those from the seen sources, and should be highlighted. ADAMEL measures such difference through the Euclidean-distance between $f(\mathbf{x})$ and $\mathbf{c}_{\mathcal{D}_S}$, as the deviating attention vectors are more likely to correspond to unseen data sources in the projected space. In the loss function shown in Equation (7), we compare the distance $d(f(\mathbf{x}), \mathbf{c}_{\mathcal{D}_S})$ with the "mean distance to cluster centroids" to give higher weights to entity pairs in $\mathcal{S}_U$ that are deviating from seen data sources.

$$L_{\text{support}} = \sum_{y_i=1} \frac{d(f(\mathbf{x}^{+}_i), \mathbf{c}^{+}_{\mathcal{D}_S})}{\bar{d}^{+}_{\mathcal{D}_S}} \log \hat{y}_i + \sum_{y_i=0} \frac{d(f(\mathbf{x}^{-}_i), \mathbf{c}^{-}_{\mathcal{D}_S})}{\bar{d}^{-}_{\mathcal{D}_S}} \log(1 - \hat{y}_i) \tag{7}$$

where d denotes the Euclidean distance, $\bar{d}^{+/-}$ represents the mean distance for all positive/negative pairs in $\mathcal{D}_S$ to the corresponding centroid. Thus, by integrating $L_{support}$ with L_{base}, the updated loss of ADAMEL in the supervised setting is denoted as $L_{\text{ssl}} = L_{\text{base}} + \phi L_{\text{support}}$, where $\phi \in (0, 1]$ is a hyperparameter that controls the impact of the labeled support set. The training process updates not only parameters in the neural network Θ for better classification performance, but also the attention embedding function f so that the projected positive and negative feature attentions are matched closer. In this process, feature importance from the new data sources unseen in $\mathcal{D}_S^*$ can be incorporated to update the centroids $\mathbf{c}^{+/-}$ in the supervised manner. We name this solution ***AdaMEL-few*** as it uses a few labeled data in $\mathcal{D}_T$, and depicts the process in Algorithm 2.

Hybrid Model We further propose a hybrid model that incorporates both the labeled support set and the unlabeled data from $\mathcal{D}_T$ in the training process. It can be seen as the composition of *AdaMEL-zero* and *AdaMEL-few*, and uses the loss L_{target} and L_{support} defined in Equation (7) as $L_{\text{hybrid}} = (1 - \lambda)L_{\text{base}} + \lambda L_{\text{target}} + \phi L_{\text{support}}$. We name this hybrid solution as ***AdaMEL-hyb*** that describes the algorithm in Algorithm 3.

5 Experiments

5.1 Experimental Setup

Data We use both the public dataset from the Data Integration to Knowledge Graphs (DI2KG) challenge [5] and two real-world datasets in different scales from Amazon. Both the public and private datasets are in the tabular form and the entities are associated with textual features. We provide the dataset statistics with source info in Table 1, and the description in Section 7.4 of the Appendix. Comparing with the public benchmark datasets [24], the above datasets are collected from larger ranges of real-world sources with heterogeneous schemas, which makes them more challenging.

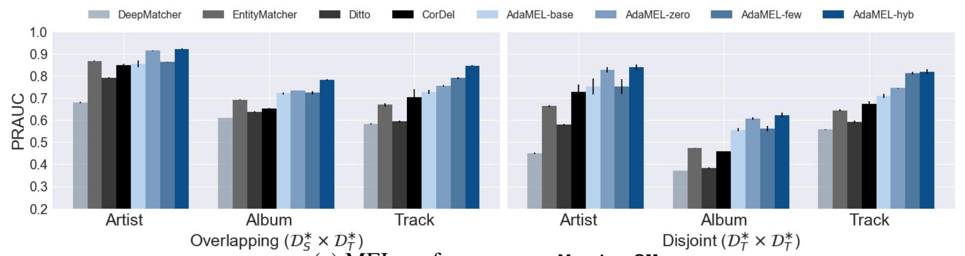
Baselines & Configuration We compare ADAMEL with the state-of-the-art deep learning baselines, most of them are proposed to handle heterogeneous entity linkage, namely: DeepMatcher [24], Entity-Matcher [11], Ditto [22] and CorDel [31]. These baselines are reported to achieve the state-of-the-art EL performance and outperform methods such as Seq2SeqMatcher [25] and DeepMatcher+ [18]. In our experiments, we follow the original paper and fine-tune the baseline approaches for optimal performance (see detailed configuration in Section 7.4.). To evaluate the effectiveness of our proposed framework, we configure ADAMEL with consistent setup as the baselines. The dimension of the projected embeddings per feature is $H = 64$, the hidden layer in f is $H' = 256$, and the hidden layer in Θ is $H_{\text{hidden}} = 256$. The activation σ is set to be Relu. We set $\lambda = 0.98$ and $\phi = 1.0$ for ADAMEL variants unless otherwise addressed. In training, we use Adam optimizer [20] for 100 epoches with learning rate $= 10^{-4}$ and batch size $= 16$. We evaluate the models using PRAUC (Area under the precision/recall curve) as it measures the precision-recall relation globally.

5.2 Transfer Learning for MEL

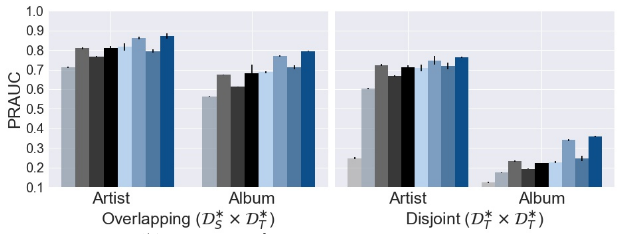
We first experiment the effectiveness of ADAMEL variants for the MEL task. We simulate two real-world scenarios: **(S1)** data in the target domain ($\mathcal{D}_T$) shares common data sources with the source domain ($\mathcal{D}_S$) (*i.e.*, $(r, r')_T \in \mathcal{D}_S^* \times \mathcal{D}_T^*$), and **(S2)** data sources in the target domain are disjointed from the source domain (*i.e.*, $(r, r')_T \in \mathcal{D}_T^* \times \mathcal{D}_T^*$). For the `Music` data, we use three data sources (*i.e.*, $\mathcal{D}_S = \{$*website 1, 2, 3*$\}$) to train our model and test on all 7 sources (overlapping scenario **S1**) or only the 4 remaining sources (disjoint scenario **S2**) as the target domain $\mathcal{D}_T$. In either scenario, we randomly collect 100 samples (50 positive and 50 negative) from the corresponding $\mathcal{D}_T$ as support set $\mathcal{S}_U$. For the public `Monitor` data, we use entity pairs from 5 sources (*i.e.*, $\mathcal{D}_S = \{$*ebay.com*, *catalog.com*, *best-deal-items.com*, *cleverboxes.com*, *ca.pcpartpicker.com*$\}$) to train the models. We use data in all 24 sources as $\mathcal{D}_T$ for **S1**, and the rest 19 data sources for **S2**, respectively. 100 samples are collected as $\mathcal{S}_U$ in the same way as `Music`. We report the results in Figure 2 with complete numerical results in Table 4 and 5 of Section 7.5 of the appendix. Our first observation is that all ADAMEL variants tend to outperform the baseline methods and our base model without adaptation, ADAMEL-*base*. Also, we observe that *AdaMEL-hyb* achieves the best performance in all cases

Table 1: Data statistics and properties

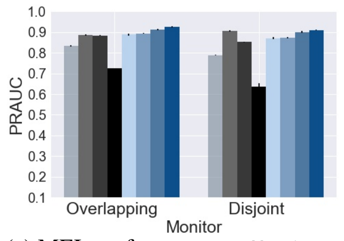
Data	# Records	Entity_types	$\lvert\mathcal{D}_S^*\rvert$	$\lvert\mathcal{D}_T^*\rvert$	$\lvert\mathcal{A}\rvert$
`Monitor`	66,795	Monitor	5	24	13
`Music-3K`	3,070	Artist, Album, Track	3	7	9
`Music-1M`	1,723,426	Artist, Album	3	7	9



(a) MEL performance on `Music-3K`



(b) MEL performance on `Music-1M`



(c) MEL performance on `Monitor`

Figure 2: MEL performance (PRAUC) comparison. ADAMEL variants outperform baseline methods in almost all cases. Particularly, *AdaMEL-hyb* performs the best on all entity types and datasets.

with $0.64\% \sim 5.50\%$ improvement in PRAUC than the second-best (*AdaMEL-zero* in most cases), which demonstrates its effectiveness in integrating both the labeled support set $\mathcal{S}_U$ and unlabeled info from the target domain $\mathcal{D}_T$. *AdaMEL-zero* performs better than *AdaMEL-few* on the "Artist" and "Album" type, while *AdaMEL-few* performs better on the "Track" type. This is likely due to the fact that the track records are more diverse than the other types as the digital-format music tracks can be remixed or covered by other artists. Thus, the high-quality labeled samples from $\mathcal{S}_U$ is of higher value. The improvement of ADAMEL variants over the baselines indicates the effectiveness of domain adaptation in incorporating data in $\mathcal{D}_T$. Overall, ADAMEL variants achieve better performance on the overlapping scenario (**S1**) than the disjoint scenario (**S2**). Besides, the performance of all approaches running on `Music-1M` is lower than `Music-3K`. The main reason is that the data is weakly labeled as it simply follows the hyperlinks from the websites, and does not distinguish the actual media of the music work (*i.e.*, the physical or digital copy). Table 5 gives the result on `Monitor`. We observe similar results that ADAMEL variants tend to outperform the baselines and *AdaMEL-hyb* performs the best with at least 0.51% improvement in PRAUC over the second best, EntityMatcher.

5.3 Attention Analysis

Here we showcase the learned feature importance through the attention scores produced by ADAMEL on two datasets: `Music-3K` and `Monitor`. We only report the artist type for brevity. *AdaMEL-hyb* is configured with $\lambda = 0.98, \phi = 1.0$. For `Monitor` dataset we observe the long "tail distribution" of feature importance, *i.e.*, the most important feature is "Page_title_shared" with significantly high scores, while the other features are with roughly the same low scores. On the other hand, we observe the more uniform distribution for the artist type in `Music-3K` dataset, which makes sense as all top features are related to the artist names. The learned attention scores on both datasets imply that the task of MEL could be addressed with some of the most remarkable features (importance inequality).

```
Monitor:  {Page_title_sim:  0.1635, Page_title_uni:  0.0595, Source_sim:  0.0535,
Manufacturer_uni:  0.0473, Manufacturer_sim:  0.0416}
Music-3K (artist):  {Main_performer_sim:  0.0739, Name_uni:  0.0697, Name_sim:
0.0628, Source_uni:  0.0597, Name_Native_Language_sim:  0.0583}
```

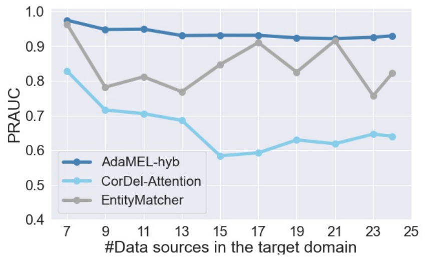
We further run *AdaMEL-hyb* on these selected important features only and compare the performance with the result using the other features, as well as all the features. For `Monitor`, we use 3 attributes (*i.e.*, "Page_title", "Source" and "Manufacturer"). For the artist type of `Music-3K`, we use the 3 name-related attributes (*i.e.*, "Main_performer", "Name", Name_Native_Language), and "Source". Similarly, for the other two types, we use their corresponding top important attributes, and report the results in Table 2. We observe that by using the selected important features only, ADAMEL is capable of achieving comparable and even slightly better performance than using all features with 2.21%, 0.87% and 2.92% improvement in PRAUC on `Monitor`, `Music-3K` (artist) and `Music-3K` (album), respectively. For `Music-3K` (track), using the top attributes only performs slightly worse than using all attributes, which is likely due to the diversity of track records. Nevertheless, these experimental results show that model training can further benefit from feature importance as using all the possible attributes could introduce irrelevant or noisy input to the model (*e.g.*, using album-related features when inferring the artist type).

Table 2: Performance (PRAUC) comparison using the selected important features vs. the other features and all features. Numbers in the parenthesis denote the counts of features.

Dataset	Top Attributes (#)	Other Attributes (#)	All Attributes (#)
`Monitor`	**0.9479** $\pm$ **0.0007** (3)	0.4276 $\pm$ 0.0015 (10)	0.9258 $\pm$ 0.0025 (13)
`Music-3K`, artist	**0.9298** $\pm$ **0.0036** (4)	0.7966 $\pm$ 0.0005 (5)	0.9211 $\pm$ 0.0040 (9)
`Music-3K`, album	**0.8125** $\pm$ **0.0011** (4)	0.4692 $\pm$ 0.0009 (5)	0.7833 $\pm$ 0.0031 (9)
`Music-3K`, track	0.8398 $\pm$ 0.0004 (3)	0.7026 $\pm$ 0.0006 (6)	**0.8454** $\pm$ **0.0040** (9)

5.4 Data Sources Analysis

We also experiment the stability of AdaMEL in handling new data sources that arrive incrementally in the process of real-world knowledge integration (**Q4**). We use the public `Monitor` dataset and compare *AdaMEL-hyb* ($\lambda = 0.98, \phi = 1.0$) with the best-performing baseline, EntityMatcher, and the fastest baseline, CorDel-Attention. We use 1500 entity pairs from the same 5 data sources as mentioned in Section 5.2 to train the models. To test the performance on MEL, we first randomly select 200 entity pairs from each of 5 unseen data sources and form totally 1000 pairs to create the target domain. Then, we incrementally add up to 200 entity pairs from 2 new sources ($\Delta\mathcal{D}_T$) to $\mathcal{D}_T$, such that $\mathcal{D}_T = \mathcal{D}_T \cup \Delta\mathcal{D}_T$. We randomly select 100 labeled samples from $\mathcal{D}_T$ to create $\mathcal{S}_U$ and fix it throughout each run of the experiment so that its impact is consistent. We also record the average runtime as an empirical study of the model efficiency. As shown in the Figure 3, *AdaMEL-hyb* is more stable than both EntityMatcher and CorDel-Attention with significantly higher performance in handling the incrementally incoming data sources. This is due to the fact that *AdaMEL-hyb* continuously updates parameters in the attention embedding function f to adapt to new data sources in $\mathcal{D}_T$. Comparing with CorDel-Attention, EntityMatcher performs better and could occasionally compete with *AdaMEL-hyb* under some scenarios ($|\mathcal{D}^*_T| = 17, 21$), but it is not stable as the performance fluctuates. Also, *AdaMEL-hyb* takes less time to train than both baselines. The empirical runtime comparison corresponds to our conjecture as *AdaMEL-hyb* does not require sophisticated operations on word-level embeddings and thus having relatively less parameters to train. These findings demonstrate the capability of AdaMEL in consistently handing MEL with a variety of incoming data sources, while being more robust. Also, they strengthen our claim that finding important features as the transferable knowledge in MEL could benefit the model performance with reduced computational complexity.



Method	Runtime (s)
AdaMEL-hyb	319.20 $\pm$ 7.20
CorDel	906.19 $\pm$ 46.35
EntityMatcher	2500.43 $\pm$ 17.56

Figure 3: *AdaMEL-hyb* performs more stably as #data sources increases in $\mathcal{D}_T$ with less runtime.

6 Conclusion

In this work, we tackle the problem of multi-source entity linkage (MEL) and described a deep learning solution based on domain adaptation, AdaMEL. AdaMEL highlights the impact of important attributes in MEL and automatically learns feature importance that adapts to the both seen and unseen data sources as the generic transferable knowledge. We propose a series of AdaMEL variants to handle different real-world learning scenarios, depending on the availability of labeled entity pairs from the target domain. Comparing to heterogeneous schema matching baselines, AdaMEL is able to handle hard transfer learning cases such as unseen data sources in the target domain and training on weakly-labeled data, while achieving up to 11.24% improvement than the baselines based on supervised learning in PRAUC score for the multi-source entity linkage task. We also provide the analysis on the learned feature attention and experiment the impact of data sources.

Customer Impact

This work focuses on linking entities from multiple external WEB sources to help Alexa answer queries about them. The aggregated knowledge will reduce the number of customer queries where Alexa fails to answer due to missing data. The quality of entity linkage directly impacts the quality of answers as such any improvements in linkage quality is expected to enhance overall customer experience. The research is not expected to put any customer segments at disadvantage. We do not leverage any biases in the data.

References

[1] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36, 2012.

[2] Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48, 2003.

[3] Rita Chattopadhyay, Qian Sun, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Multisource domain adaptation and its application to early detection of fatigue. *ACM TKDD*, 6(4):1–26, 2012.

[4] William W Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *KDD*, pages 475–480, 2002.

[5] di2kg. 2nd international workshop on challenges and experiences from data integration to knowledge graphs. http://di2kg.inf.uniroma3.it/2020/, 2020.

[6] AnHai Doan and Alon Y Halevy. Semantic integration research in the database community: A brief survey. *AI magazine*, 26(1):83–83, 2005.

[7] Xin Luna Dong and Felix Naumann. Data fusion: resolving data conflicts for integration. *Proceedings of the VLDB Endowment*, 2(2):1654–1655, 2009.

[8] Lixin Duan, Dong Xu, and Ivor Wai-Hung Tsang. Domain adaptation from multiple sources: A domain-dependent regularization approach. *IEEE Transactions on neural networks and learning systems*, 23(3):504–518, 2012.

[9] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. Reasoning about record matching rules. *Proceedings of the VLDB Endowment*, 2(1):407–418, 2009.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[11] Cheng Fu, Xianpei Han, Jiaming He, and Le Sun. Hierarchical matching network for heterogeneous entity resolution. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3665–3671, 2020.

[12] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE ICCV*, pages 2961–2969, 2017.

[15] Di Jin, Mark Heimann, Ryan A Rossi, and Danai Koutra. node2bits: Compact time-and attribute-aware node representations for user stitching. In *ECML-PKDD*, pages 483–506. Springer, 2019.

[16] Muhammad Ebraheem Saravanan Thirumuruganathan Shafiq Joty and Mourad Ouzzani Nan Tang. Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 11(11), 2018.

[17] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of ACL: Volume 2, Short Papers*, pages 427–431, 2017.

[18] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5851–5861, 2019.

[19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010.

[22] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, 14(1):50–60, 2020.

[23] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on EMNLP*, pages 1412–1421, 2015.

[24] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34, 2018.

[25] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 629–638, 2019.

[26] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[27] Kun Qian, Lucian Popa, and Prithviraj Sen. Active learning for large-scale entity resolution. In *Proceedings of the 2017 ACM CIKM*, pages 1379–1388, 2017.

[28] Rohit Singh, Venkata Vamsikrishna Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. Synthesizing entity matching rules by examples. *VLDB*, 11(2):189–202, 2017.

[29] Shiliang Sun, Honglei Shi, and Yuanbin Wu. A survey of multi-source domain adaptation. *Information Fusion*, 24:84–92, 2015.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in NIPS*, pages 5998–6008, 2017.

[31] Zhengyang Wang, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Shuiwang Ji. Cordel: A contrastive deep learning approach for entity linkage. *arXiv preprint arXiv:2009.07203*, 2020.

[32] Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM TIST*, 11(5):1–46, 2020.

[33] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *ICML*, pages 5085–5094, 2018.

[34] Chen Zhao and Yeye He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *WWW*, pages 2413–2424, 2019.

7 Supplementary Materials

7.1 Definition of Symbols and Notions

In this section, we list the symbols and notations used in this paper as well as their definitions in Table 3.

Table 3: Summary of notation

Symbol	**Definition**
$\mathcal{A} = \{A_j\}$	a set of pre-defined textual attributes (data source schema)
$r, r[A]$	an entity record and the value (word tokens) of attribute A
$\mathcal{D}_S, \mathcal{D}_T$	source and target domain, respectively
$(r, r')_{S/T}$	an entity pair in the source and target domain, respectively
S, S'	set of data sources in general
r^*	the data source that record r is sampled from
$\mathcal{D}^*$	set of data sources in a domain, *e.g.*, $\mathcal{D}_S^* = \{r^*\}_{r \in \mathcal{D}_S}$
F	the number of relational features, $F = 2\|\mathcal{A}\|$
$\mathbf{x}, y$	H-dim latent feature vector of an entity pair and its label
$\mathbf{h}_j$	D-dim token embedding of feature j
f	attention embedding function $\mathbb{R}^{D \times F} \to \mathbb{R}^F$

7.2 Detailed Algorithms

Algorithm 1 *AdaMEL-zero*

```
Input: $\mathcal{D}_S = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{D}_T = \{\mathbf{h}_i\}$, $\lambda$, batch size $B$
Output: Predicted $\hat{y}_i$ for $\mathbf{h}_i \in \mathcal{D}_T$, updated $\mathbf{a}, \mathbf{W}$
1: Initialize $\mathbf{a}, \mathbf{W}$ and $\mathbf{V}, \mathbf{b}$
2: loop training epochs
3:     for $\mathbf{h} \in \mathcal{D}_S \cup \mathcal{D}_T$ do
4:         Form $\mathbf{x}$ with $\mathbf{V}, \mathbf{b}$                                ▷ Eq. (4)
5:     $\bar{f}(\mathbf{x}') \leftarrow \frac{1}{|\mathcal{D}_T|} \sum_{\mathbf{x_i} \in \mathcal{D}_T} f(\mathbf{x_i})$
6:     $J \leftarrow 0$                                                ▷ Initialize loss
7:     $\mathcal{S}_{\text{batch}} \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_S, B)$
8:     for $(\mathbf{x}, y) \in \mathcal{S}_{\text{batch}}$ do
9:         $L_{\text{un}} \leftarrow (1-\lambda) L_{\text{base}} + \lambda L_{\text{target}}$            ▷ Loss function
10:        $J \leftarrow J + \nabla L_{\text{un}}$                          ▷ Update $\mathbf{a}, \mathbf{W}, \mathbf{V}, \mathbf{b}$
11: end loop
12: Form $\mathbf{x}, f$ with updated $\mathbf{a}$, $\mathbf{W}$, $\mathbf{V}, \mathbf{b}$                 ▷ Eq. (6)
13: $\hat{\mathbf{y}} \leftarrow \emptyset$
14: for $\mathbf{x}_i \in \mathcal{D}_T$ do
15:     $\hat{\mathbf{y}}_i \leftarrow \Theta(\sigma(f(\mathbf{x}_i) \odot \mathbf{x}_i))$
16: return $\hat{\mathbf{y}}$, $\mathbf{a}$, $\mathbf{W}$
```

In this section we depict algorithms for *AdaMEL-zero* (Algorithm 1), *AdaMEL-few* (Algorithm 2) and *AdaMEL-hyb* (Algorithm 3). Particularly, line 3-4 project the affine transformation of entity pairs from both $\mathcal{D}_S$ and $\mathcal{D}_T$. Line 5 computes $\bar{f}(\mathbf{x}')$, the attention vector averaged over entity pairs in the target domain. Line 8-10 computes each attention vector in the sampled batch $f(\mathbf{x}_i)$ and adapt it to $\bar{f}(\mathbf{x}')$ to compute the loss L_{target}. ADAMEL minimizes both the inference loss L_{base} and L_{target} to train the parameters in f and form the transferrable knowledge $\mathcal{K} = f(\mathbf{x}_i)$ for $\mathbf{x}_i \in \mathcal{D}_T$ (Line 12). Line 14- 15 denote the inference process.

In Algorithm 2, line 7- 8 denote the training process of f to minimize the loss L_{base}, and line 10- 11 denote the process of further training under the supervision of labeled samples in $\mathcal{S}_U$. Algorithm 3 integrates both $\mathcal{S}_U$ and the unlabeled data from $\mathcal{D}_T$.

Algorithm 2 *AdaMEL-few*

```
Input: $\mathcal{D}_S = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{S}_U = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{D}_T = \{\mathbf{h}_i\}$, $\phi$, $B$
Output: Predicted $\hat{y}_i$ for $\mathbf{h}_i \in \mathcal{D}_T$, updated $\mathbf{a}, \mathbf{W}$
1: Initialize $\mathbf{a}, \mathbf{W}$ and $\mathbf{V}, \mathbf{b}$
2: loop training epochs
3:     for $\mathbf{h} \in \mathcal{D}_S \cup \mathcal{D}_T$ do
4:         Form $\mathbf{x}$ with $\mathbf{V}, \mathbf{b}$                          ▷ Eq. (4)
5:     $J \leftarrow 0$                                                  ▷ Initialize loss
6:     $\mathcal{S}_{\text{batch}} \leftarrow$ RANDOMSAMPLE($\mathcal{D}_S, B$)
7:     for $(\mathbf{x}, y) \in \mathcal{S}_{\text{batch}}$ do
8:         $J \leftarrow J + \nabla L_{\text{base}}$                          ▷ Loss of ADAMEL-base
9:     Form $f$ with updated $\mathbf{a}, \mathbf{W}$                         ▷ Eq. (6)
10:    Compute $\mathcal{D}_S^+, \mathcal{D}_S^-, \bar{d}_{\mathcal{D}_S}^+, \bar{d}_{\mathcal{D}_S}^-$
11:    $L_{\text{ssl}} \leftarrow L_{\text{base}} + \phi L_{\text{support}}$                    ▷ Loss function
12:    $J \leftarrow J + \nabla L_{\text{ssl}}$                               ▷ Update $\mathbf{a}, \mathbf{W}, \mathbf{V}, \mathbf{b}$
13: end loop
14: Infer $\hat{\mathbf{y}}$                                   ▷ Same as Line 13- 15 of Algorithm 1
15: return $\hat{\mathbf{y}}, \mathbf{a}, \mathbf{W}$
```

Algorithm 3 *AdaMEL-hyb*

```
Input: $\mathcal{D}_S = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{S}_U = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{D}_T = \{\mathbf{h}_i\}$, $\phi$, $B$
Output: Predicted $\hat{y}_i$ for $\mathbf{h}_i \in \mathcal{D}_T$, updated $\mathbf{a}, \mathbf{W}$
1: Initialize $\mathbf{a}, \mathbf{W}$ and $\mathbf{V}, \mathbf{b}$
2: loop training epochs
3:     for $\mathbf{h} \in \mathcal{D}_S \cup \mathcal{D}_T \cup \mathcal{S}_U$ do
4:         Form $\mathbf{x}$ with $\mathbf{V}, \mathbf{b}$                          ▷ Eq. (4)
5:     $\bar{f}(\mathbf{x}') \leftarrow \frac{1}{|\mathcal{D}_T|} \sum_{\mathbf{x_i} \in \mathcal{D}_T} f(\mathbf{x_i})$
6:     $J \leftarrow 0$                                                  ▷ Initialize loss
7:     $\mathcal{S}_{\text{batch}} \leftarrow$ RANDOMSAMPLE($\mathcal{D}_S, B$)
8:     for $(\mathbf{x}, y) \in \mathcal{S}_{\text{batch}}$ do
9:         $L_{\text{un}} \leftarrow (1-\lambda) L_{\text{base}} + \lambda L_{\text{target}}$             ▷ Loss of AdaMEL-zero
10:        $J \leftarrow J + \nabla L_{\text{un}}$
11:    Form $f$ with updated $\mathbf{a}, \mathbf{W}$                         ▷ Eq. (6)
12:    Compute $\mathcal{D}_S^+, \mathcal{D}_S^-, \bar{d}_{\mathcal{D}_S}^+, \bar{d}_{\mathcal{D}_S}^-$
13:    $L_{\text{hybrid}} = L_{\text{un}} + \phi L_{\text{support}}$                    ▷ Loss function
14:    $J \leftarrow J + \nabla L_{\text{hybrid}}$                            ▷ Update $\mathbf{a}, \mathbf{W}, \mathbf{V}, \mathbf{b}$
15: end loop
16: Infer $\hat{\mathbf{y}}$                                   ▷ Same as Line 13- 15 of Algorithm 1
17: return $\hat{\mathbf{y}}, \mathbf{a}, \mathbf{W}$
```

7.3 Parameter Complexity

We measure the parameter complexity of ADAMEL in terms of the numbers of learnable parameters that comes from three parts: (i) per-feature non-linear affine operations that transform the word token embeddings to the latent feature vectors, (ii) the shared feature attention embedding function f, which includes learning $\mathbf{W}$ and $\mathbf{a}$, and (iii) the multilayer perceptron (MLP) Θ with 1 hidden layer for classification. For (i), there are totally F features, each feature is associated with $\mathbf{V}^{H \times D}$ and $\mathbf{b}$, thus leading to $\mathcal{O}(FDH)$ learnable parameters. For (ii), as $\mathbf{W}^{H' \times H}$ and $\mathbf{a}^{H'}$ are shared across all features, there are totally $\mathcal{O}(HH')$ parameters. The neural network Θ in (iii) takes the concatenated FH'-dim features as input with one H_{hidden}-dim hidden layer. Therefore, ADAMEL has totally $\mathcal{O}(FDH + HH' + FH'H_{\text{hidden}})$ parameters to learn. We discuss the setup values of H, H' and H_{hidden} in the configuration of Section 5.

7.4 Detailed Experimental Configuration

Data. Here we list the detailed description of the datasets used in our experiments.

- **Music-1M** is a weakly labeled corpus crawled from 7 public music websites. We name them *website 1-7* for confidentiality. There are 2 entity types: artists and albums. Entity pairs are labeled via the hyperlinks in pages, so there might be mixed-type matching errors, *e.g.*, matching an artist to her album.
- **Music-3K** is a manually labeled corpus containing the same data sources as **Music-1M**. It has three types: artist, album and tracks. The manual annotation is based on 9 attributes such as the artist name and album title. Errors such as mixed-type matching are carefully corrected.
- **Monitor** contains monitor data from 24 sales websites such as *ebay.com* and *shopmania.com*. We filter out attributes with $> 60\%$ empty records, and get totally 13 attributes such as product description, manufacturer info, condition status, etc.

The attribute values in the above datasets are generally longer with diverse characters, which makes it harder to summarize the attribute representation. For example, for the artist type of `Music-3K` dataset, the averaged attribute length is 25.75 word tokens, and for `Monitor`, the averaged attribute length is 11.73 word tokens. On the contrary, this number is 6.26 and 5.21 word tokens for the "dirty" and "heterogeneous" `Walmart-Amazon` dataset [11] from the benchmark, respectively. In terms of the `Music` datasets, as the music works come from different countries, many entities are recorded with non-English characters & phrases for attributes such as the title, album and artist names. Unlike `Music-1M` that labels entity pairs through website hyperlinks, `Music-3K` also inspects whether the pair of records indicate the music work from the same physical copy (*i.e.*, "Album"), or from the same digital copy in formats such as remix or cover (*i.e.*, "Track"). The `Monitor` dataset is highly imbalanced with more than 99% entity negative pairs.

Baselines. For DeepMatcher, we use its hybrid variant (bi-directional RNN with attention) to summarize attributes and 2-layer highway neural network with 300-dim hidden layer. The training epoches is set to 40 with batch size = 16. For EntityMatcher, we use the full matching model that uses bi-GRU (hidden size= 150) to embed attribute word sequences with cross-attribute token-level alignment. The training epoch is set to 20 with batch size = 16. For CorDel, we use the attention-based variant of CorDel that learns the importance of words within the same attribute to validate the effectiveness of our attribute-level attention module. Moreover, CorDel-Attention was shown to perform best on long textual attribute values, which matches the property of our input data. All these three baselines use the pretrained FastText [17] to derive the 300-dimensional embeddings for word tokens in each attribute. We set the cropping size = 20 and sum the embeddings of word tokens as the feature embeddings for CorDel. The training epoch is set to 20 with learning rate = 10^{-4} and batch size = 16. For Ditto, we tested its optimization strategies and adopted the "token span deletion" for data augmentation, "general" domain knowledge and retaining high TF-IDF tokens to summarize the input sequences. We also tested all pretrained language models, *i.e.*, bert, distilbert, and albert, and ended up using bert. The training epoch is set to 40 with batch size= 64 and learning rate= 3×10^{-5}.

We conduct all experiments 3 times and report the mean and std. We run these experiments on the Linux platform with 2.5GHz Intel Core i7, 256GB memory and 8 NVIDIA K80 GPUs.

7.5 Complete Experimental Results of MEL

In this section we provide the complete numerical result shown in Section 5.2. The results on the `Music` dataset are shown in Table 4. We observe that *AdaMEL-hyb* performs the best out of all AdaMEL variants, while *AdaMEL-few* and *AdaMEL-hyb* tend to perform second to the best. Table 5 records the `Monitor` dataset, where we observe similar findings.

7.6 Effectiveness of Adaptation

Setup. To evaluate how well AdaMEL learns feature importance adapted to the target domain (**Q2**), we study the effectiveness of λ adopted in *AdaMEL-zero* and *AdaMEL-hyb* as it controls the weight of adapting to unlabeled data in the training process (larger λ leads to more adaptation). We run both variants of AdaMEL on the `Music-3K` dataset and report the performance on MEL. As discussed in Section 4.4, records from both the source and target domains are projected into the same space using the shared attention embedding function, and AdaMEL attempts to adapt the model to match these feature importance distribution. Intuitively, with sufficient adaptation, feature importance vectors

Table 4: ADAMEL performance (PRAUC) of multi-source entity linkage on the `Music` data. The best score of each entity type is marked in bold. Out of ADAMEL variants, *AdaMEL-hyb* performs the best with $0.64\% \sim 5.50\%$ improvement over the second-best variant (marked with $*$) in PRAUC. ADAMEL outperforms the baseline with 2.67% (ADAMEL-base) to 11.24% (*AdaMEL-hyb*) improvement on average.

	Method	Overlapping ($\mathcal{D}_S^* \times \mathcal{D}_T^*$)			Disjoint ($\mathcal{D}_T^* \times \mathcal{D}_T^*$)		
		Artist	Album	Track	Artist	Album	Track
`Music-3K`	DeepMatcher	0.6794 ± 0.0022	0.6093 ± 0.0009	0.5826 ± 0.0017	0.4492 ± 0.0021	0.3710 ± 0.0012	0.5572 ± 0.0014
	EntityMatcher	0.8682 ± 0.0017	0.6922 ± 0.0021	0.6694 ± 0.0084	0.6629 ± 0.0032	0.4733 ± 0.0014	0.6446 ± 0.0032
	Ditto	0.7920 ± 0.0032	0.6373 ± 0.0042	0.5938 ± 0.0051	0.5786 ± 0.0039	0.3832 ± 0.0027	0.5914 ± 0.0055
	CorDel-Attention	0.8489 ± 0.0047	0.6531 ± 0.0019	0.7032 ± 0.0364	0.7280 ± 0.0315	0.4586 ± 0.0002	0.6738 ± 0.0121
	ADAMEL-base	0.8545 ± 0.0143	0.7204 ± 0.0033	0.7277 ± 0.0077	0.7516 ± 0.0367	0.5569 ± 0.0072	0.7107 ± 0.0093
	AdaMEL-zero	0.9142 ± 0.0018*	0.7338 ± 0.0001*	0.7547 ± 0.0027	0.8263 ± 0.0121*	0.6071 ± 0.0072*	0.7453 ± 0.0012
	AdaMEL-few	0.8633 ± 0.0011	0.7241 ± 0.0080	0.7904 ± 0.0048*	0.7510 ± 0.0331	0.5619 ± 0.0119	0.8129 ± 0.0057*
	AdaMEL-hyb	**0.9211 ± 0.0040**	**0.7833 ± 0.0031**	**0.8454 ± 0.0040**	**0.8390 ± 0.0125**	**0.6229 ± 0.0115**	**0.8193 ± 0.0097**
`Music-1M`	DeepMatcher	0.7132 ± 0.0033	0.5629 ± 0.0021		0.6033 ± 0.0045	0.1742 ± 0.0013	
	EntityMatcher	0.8098 ± 0.0043	0.6731 ± 0.0024		0.7239 ± 0.0038	0.2331 ± 0.0031	
	Ditto	0.7663 ± 0.0025	0.6123 ± 0.0022		0.6678 ± 0.0019	0.1933 ± 0.0027	
	CorDel-Attention	0.8118 ± 0.0087	0.6811 ± 0.0432		0.7129 ± 0.0096	0.2224 ± 0.0010	
	ADAMEL-base	0.8165 ± 0.0184	0.6872 ± 0.0053	-	0.7086 ± 0.0180	0.2269 ± 0.0050	-
	AdaMEL-zero	0.8607 ± 0.0066*	0.7693 ± 0.0038*		0.7469 ± 0.0228*	0.3407 ± 0.0056*	
	AdaMEL-few	0.7942 ± 0.0090	0.7126 ± 0.0102		0.7177 ± 0.0171	0.2473 ± 0.0131	
	AdaMEL-hyb	**0.8710 ± 0.0130**	**0.7942 ± 0.0015**		**0.7632 ± 0.0034**	**0.3582 ± 0.0043**	

Table 5: ADAMEL performance (PRAUC) on `Monitor`. All variants outperform the baseline, *AdaMEL-hyb* performs the best (marked in bold) with at least 1.01% improvement over the second-best ($*$).

	Method	Overlapping	Disjoint
`Monitor`	DeepMatcher	0.8336 ± 0.0032	0.7884 ± 0.0011
	EntityMatcher	0.8858 ± 0.0034	0.9051 ± 0.0042
	Ditto	0.8841 ± 0.0010	0.8518 ± 0.0023
	CorDel-Attention	0.7240 ± 0.0026	0.6353 ± 0.0165
	ADAMEL-base	0.8884 ± 0.0057	0.8711 ± 0.0050
	AdaMEL-zero	0.8930 ± 0.0013	0.8719 ± 0.0030
	AdaMEL-few	0.9127 ± 0.0035*	0.9005 ± 0.0059*
	AdaMEL-hyb	**0.9258 ± 0.0025**	**0.9106 ± 0.0029**

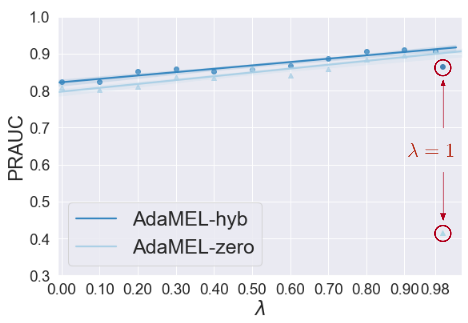
from both domains should align well. We study the performance of *AdaMEL-zero* and *AdaMEL-hyb* given different values of λ on the "artist" and "album" type from the `Music-3K` dataset.

Results. To evaluate the impact of adaptation to the linkage results, in Figure 4 we show the performance of our variants with different λ values. We observe that as λ approaches (but not equals) to 1, the general performance in terms of PRAUC improves for both *AdaMEL-zero* (0.8014 - 0.9091) and *AdaMEL-hyb* (0.8242 - 0.9201), which again demonstrates the effectiveness of adaptation. It is worth noting that when $\lambda = 1$, both *AdaMEL-zero* and *AdaMEL-hyb* perform worse without giving meaningful results. This is because at this point, *AdaMEL-zero* is trained without supervision of the labeling in $\mathcal{D}_S$, and the only term in the loss function is the regularization. *AdaMEL-hyb* is better as labeling in $\mathcal{S}_U$ is still used, but the overall performance deteriorates due to the lack of labeling from $\mathcal{D}_S$. As a result, the parameters trained (*i.e.*, $\mathbf{a}, \mathbf{W}$) would tend to only "match" the feature distribution between $\mathcal{D}_S$ and $\mathcal{D}_T$ that are not tailored to classification.

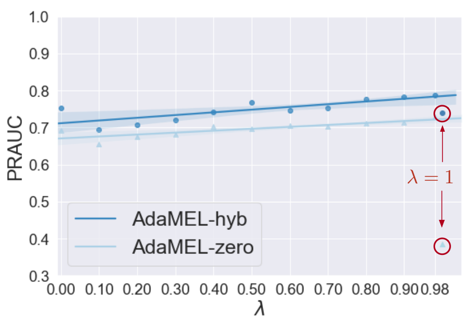
In Figure 5, we observe that for both variants, feature attention vectors from $\mathcal{D}_S$ and $\mathcal{D}_T$ align better when $\lambda = 0.98$ than $\lambda = 0$, which confirms the effectiveness of adaptation. In addition, we observe that comparing with *AdaMEL-zero* (Figure 5b), *AdaMEL-hyb* (Figure 5d) generates better adapted results as the projected records from $\mathcal{D}_S$ and $\mathcal{D}_T$ are almost indistinguishable, which is as expected as the labeled support set is leveraged.

7.7 Effectiveness of Support Set

Setup. To better understand the effectiveness of the labeled support set (**Q5**), we perform the sensitivity analysis with incrementally increasing numbers of labeled samples in the support set $\mathcal{S}_U$. Following Section 5.2, we randomly select 200 additional samples from $\mathcal{D}_T$ of the public `Monitor` dataset and create the support set with totally 300 labeled samples. We run two ADAMEL variants that leverage the support set, *AdaMEL-few* ($\phi = 1.0$) and *AdaMEL-hyb* ($\lambda = 0.98, \phi = 1.0$) in this experiment with $|\mathcal{S}_U|$ ranging from 1 to 300 with step size $=$ 20 (specifically, we "zoom in" the smaller values and have $|\mathcal{S}_U| = \{1, 5, 10, 20, 40, 60, \cdots, 300\}$). In each run, the samples in $\mathcal{S}_U$ are randomly selected.

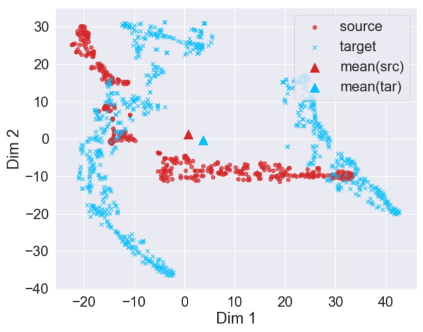


(a) ADAMEL performance change on `Music-3K`, artist type

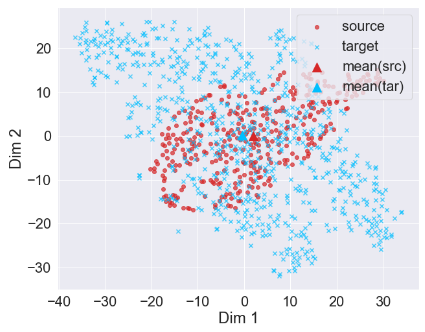


(b) ADAMEL performance change on `Music-3K`, album type

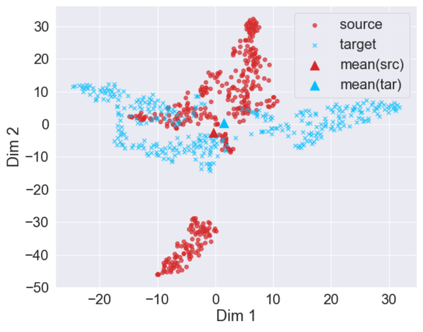
Figure 4: *AdaMEL-zero* and *AdaMEL-hyb* performance improve with increasing λ from 0 to 0.98 (fitted with linear regression). The performance drops when $\lambda = 1$ as no labeled data in $\mathcal{D}_S$ is used.



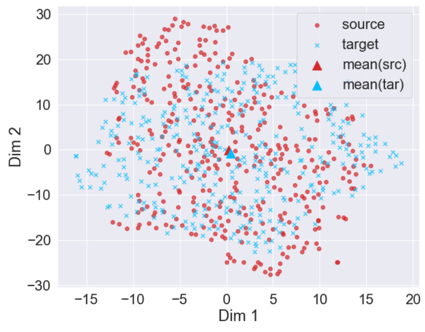
(a) *AdaMEL-zero* with no adaptation ($\lambda = 0$)



(b) *AdaMEL-zero* with adaptation ($\lambda = 0.98$)



(c) *AdaMEL-hyb* with no adaptation ($\lambda = 0$)



(d) *AdaMEL-hyb* with adaptation ($\lambda = 0.98$)

Figure 5: Source and target domain feature attention vectors are better aligned with high value of λ for both *AdaMEL-few* and *AdaMEL-hyb* (visualized with TSNE, dim=2).

Result. The experimental result is shown in Figure 6. Our first observation is that at the initial stage of the experiment, the performance of both *AdaMEL-few* and *AdaMEL-hyb* improves as the number of used labeled samples from $\mathcal{S}_U$ increases. Particularly, we observe $\sim 1\%$ performance improvement from $|\mathcal{S}_U| = 1$ to $|\mathcal{S}_U| = 140$ for *AdaMEL-few* and $2\% \sim 3\%$ improvement for *AdaMEL-hyb*. This overall performance improvement is as expected since an increasing amounts of labeled samples from $\mathcal{D}_T$ are used to supervise the learning process. In the late stage ($|\mathcal{S}_U| > 140$), we observe that the performance fluctuates in each run and the overall performance saturates. This indicates that the feature importance learned by ADAMEL has sufficiently adapted and does not significantly change as more labeled data are collected in $\mathcal{S}_U$. Moreover, comparing with *AdaMEL-few*, *AdaMEL-hyb* performs similarly when the size of support set is small ($|\mathcal{S}_U| \leq 60$), and it consistently outperforms when $|\mathcal{S}_U| > 60$. This is likely due to the bias of feature importance brought by particular labeled samples selected when $|\mathcal{S}_U|$ is small. When $\mathcal{S}_U$ contains more samples, the learned feature importance becomes stable and sufficiently adapted to $\mathcal{S}_U$, and the outperformance given by *AdaMEL-hyb* over *AdaMEL-few* comes from the unlabeled samples from $\mathcal{D}_T$. As a rule of thumb, Figure 6 indicates that a small support set with $|\mathcal{S}_U| = 100 \sim 200$ labeled samples from $\mathcal{D}_T$ is beneficial to learn feature

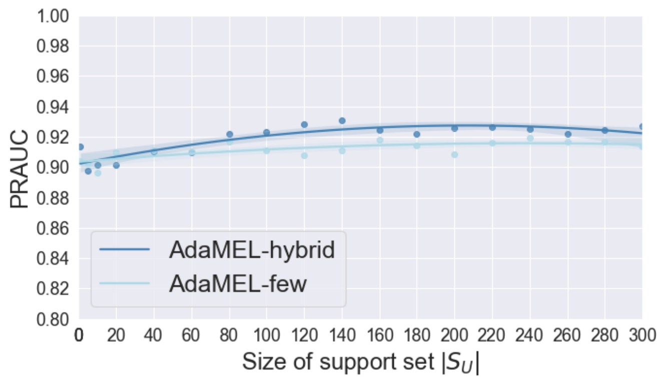


Figure 6: Sensitivity analysis of the size of support set $|\mathcal{S}_U|$ fitted with order-2 polynomial regression on *AdaMEL-few* and *AdaMEL-hyb*. As more labeled samples are included in $\mathcal{S}_U$, the model performance (PRAUC) increases initially and then saturates.

importance and to improve the MEL performance of ADAMEL. Too few samples would incur bias to the trained model, while too many samples would be expensive to obtain in practice, and does not necessarily help improve the model.

7.8 Additional Related Work

Here we describe additional works related to this paper in detail.

In addition to DeepER [16] and DeepMatcher [24], CorDel [31] proposes to compare and contrast the pairwise input records before getting the embeddings so that small but critical differences between attributes can be modeled effectively. There are also recent works that formulate entity linkage across different data sources as heterogeneous entity matching [22, 11, 25], for example, EntityMatcher [11] proposes a hierarchical matching network that jointly match entities in the token, attribute, and entity level. Ditto [22] proposes to leverage the pretrained language model [16, 19, 22] such as BERT or DistilBERT, as well as domain knowledge and data augmentation to improve the matching quality. The basis of these above deep models for heterogeneous schema matching is to accurately summarize the attribute words through advanced NLP techniques such as word token-level RNN (with attention) or using pretrained language models. On the contrary, our proposed method does not require sophisticated computation to summarize words in each attribute. ADAMEL focuses on the impact of important attributes in matching and explicitly models their importance using the soft attention mechanism as the transferable knowledge. Such attribute-level importance is agnostic to specific data sources and generalizes better than individual words in the transfer learning paradigm.
In transfer learning, DeepMatcher+ [18] extends DeepMatcher with the combination of transfer learning and active learning to achieve comparable performance with fewer samples. However, this work aims at dataset adaptation rather than the attribute matching, and the focus is not improving the matching performance. ADAMEL explicitly learns feature importance by adapting to the massive unlabeled data from unseen sources as the transferable knowledge for the multi-source EL task.