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

Di Jin
University of Michigan
Ann Arbor, USA
dijin@umich.edu

Bunyamin Sisman
Amazon
Seattle, USA
bunyamis@amazon.com

Hao Wei
Amazon
Seattle, USA
wehao@amazon.com

Xin Luna Dong
Amazon
Seattle, USA
lunadong@amazon.com

Danai Koutra
University of Michigan
Ann Arbor, USA
dkoutra@umich.edu

ABSTRACT

Multi-source entity linkage focuses on integrating knowledge from multiple sources by linking the records that represent the same real world entity. This is critical in high-impact applications such as data cleaning and user stitching. The state-of-the-art entity linkage pipelines mainly depend on supervised learning that requires abundant amounts of training data. However, collecting well-labeled training data becomes expensive when the data from many sources arrives incrementally over time. Moreover, the trained models can easily overfit to specific data sources, and thus fail to generalize to new sources due to significant differences in data and 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 the attribute importance that is used to match entities 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. In addition, AdaMEL is capable of incorporating an additional set of labeled data to more accurately integrate data sources with different attribute importance. Extensive experiments show that our framework achieves state-of-the-art results with 8.21% improvement on average over methods based on supervised learning. Besides, it is more stable in handling different sets of data sources in less runtime.

PVLDB Reference Format:
Di Jin, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Danai Koutra. Deep Transfer Learning for Multi-source Entity Linkage via Domain Adaptation. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

1 INTRODUCTION

Entity linkage (EL), also known as entity resolution, record linkage, entity matching, is a fundamental task in data mining, database, and knowledge integration with numerous 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. For example, some music websites record the song "Hey Jude" by Paul McCartney with the name abbreviation (i.e., "P.M.") while others with the band name (i.e., "The Beatles"). 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, 13] and personalization [16].

Methods for solving the entity linkage problem across data sources include rule reasoning [9, 32], computation of similarity between attributes or schemas [2], and active learning [30]. In particular, recent deep learning approaches that are based on heterogeneous schema matching or word matching [23, 26, 27] have been widely studied. Their promising performance mainly comes from the sophisticated word-level operations such as RNN and Attention [11, 26] to represent token sequences under attributes as the summarization, or the usage of pretrained language models [20] to better learn the word semantics. However, all the above learning 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 it can be either well-labeled (*e.g.*, through manual confirmation) or unlabeled. While the existing frameworks can handle high-quality labeled data, they cannot deal with the massive volume of unlabeled and previously unseen data or missing values. 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 attributes or attributes that are rarely seen (*i.e.*, "Gender").

Motivated by real-world knowledge integration settings, we consider three key challenges in the multi-data source scenario: (**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*

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

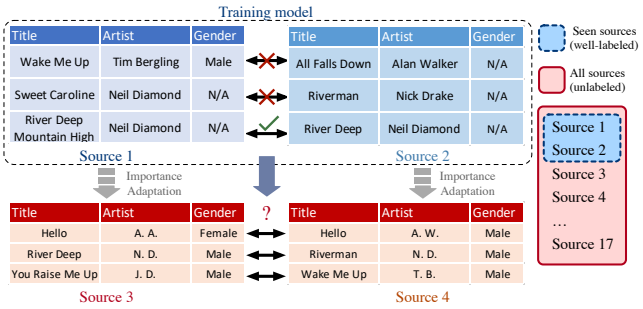


Figure 1: Well-labeled data sources (*e.g.*, blue tables) are generally outnumbered by massive unlabeled data in real-world knowledge integration scenarios. Entity linkage models trained only on well-labeled samples fail to handle new sources with different contexts or formats (*e.g.*, red tables). Our proposed framework, AdaMEL automatically learns the attribute importance that adapts to the massive unlabeled data from different sources during training, and then uses it as the transferable knowledge to perform matching.

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, human experts typically rely on prior domain knowledge to learn the attribute importance in the seen data sources, and then *transfer* it to match the unseen records based on the similarity of attribute values. As challenges (**C1-C3**) lead to different attribute importance, human experts would update their knowledge learned from the seen data sources and adapt to the unseen data sources. For example in Figure 1, when trying to link entities in the red table, the importance of "Artist" learned in the seen data sources (blue table) is down-weighted due to the fact that name abbreviation is less informative. On the other hand, even though it is a rarely-seen attribute in the seen sources, "Gender" would be more important because the gender difference between artists naturally leads to non-matching of music records regardless of the fact that entity pairs could share the same "Title" ("Hello") and very similar "Artist" values ("A. A." and "A. W."). This process, however, is tedious and does not scale to massive unlabeled data involved in real-world entity linkage problems, where large volumes of new data sources continuously arrive.

Following this intuition, 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). In general, as transfer learning aims to transfers knowledge learned from the domain with abundant training data to a related target domain with limited data, the existing works either rely on increasing the labeling volume by introducing the external data (*e.g.*, public knowledge bases) [40] or reusing the seen training data [34]. On the contrary, 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), so that the knowledge is adaptive to the continuously incoming data sources. In addition, the insightful feature importance as transferrable knowledge is explicitly defined by AdaMEL to benefit both human interpretation and the performance of learning tasks, which is also different from methods that incorporates the knowledge into pretrained models like "black-boxes", such as the contextual word/character embeddings. While the widely-adopted NLP-based attribute summarization in existing works [11, 23, 27] could accurately capture the word-level semantics using some pretrained language models or domain knowledge for all attributes, they are too computationally expensive for the practical scenario. On the contrary, the feature-level attention is much faster to obtain and we claim that the impact of word-level similarity under some attributes is limited and even harmful for model performance if those attributes are not important.

AdaMEL follows the real-world scenario and assumes that new data sources come from the same or neighboring domains in batches (*e.g.*, music from different websites). Transferring knowledge between irrelevant domains (*e.g.*, celebrities and products) does not produce meaningful outputs and is out of the scope of this paper. We also propose a series of AdaMEL variants for different learning scenarios in practice. Our contributions are summarized as follows.

- We formulate the problem of MEL in real-world knowledge integration where the incoming data of unseen data sources are associated with missing values, unseen attributes and different value distributions.
- We propose a deep transfer-learning framework that learns the attribute-level importance as the high-level knowledge, and incorporates massive unlabeled data across multiple unseen sources through domain adaptation to make it agnostic and transferable.
- We apply AdaMEL to multi-source entity linkage over both industrial and public datasets, and show that it achieves at least 5.92% improvement in terms of mean average precision compared to the state-of-the-art deep learning EL methods.

2 RELATED WORK

Entity Linkage (EL). Entity linkage has been and continues being a fundamental problem in the field of database, data mining and knowledge integration [6, 13, 22]. Early works are based on the similarity between entity attributes [6, 9] through resolving the data conflicts [7], linking relevant attributes through semantic matching or rule reasoning [32]. Techniques such as blocking or hashing are normally applied to merge the candidate entities [4]. The major drawback of these methods is the dependence on prior knowledge as the useful attributes are normally selected through human efforts. Recently, EL models based on deep neural networks [17, 26] have been widely studied due to their capability in automatically deriving latent features and promising results in fields such as CV and NLP [1, 10, 24]. For example, DeepER [17] proposes to leverage RNN to compose the pre-trained word embeddings of tokens within all attribute values, and use them as features to conduct EL as the binary classification task. DeepMatcher [26] also takes the embeddings of attribute words as the input and uses RNN to obtain the attribute similarity representation.CorDel [37] 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 [11, 23, 27], for example, EntityMatcher [11] proposes a hierarchical matching network that jointly match entities in the token, attribute, and entity level. Ditto [23] proposes to leverage the pretrained language model [17, 20, 23] such as BERT or DistilBERT, as well as domain knowledge and data augmentation to improve the matching quality. The attention mechanisms [24, 35, 36] are generally adopted by these deep models, where the goal is to improve the linkage performance by highlighting valuable embeddings, *e.g.*, word embeddings within the attributes. 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 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.

Transfer Learning. In the transfer learning scenario, models are trained on a source domain and applied to a related target domain to handle the same or a different task [14, 29]. The specific transferable knowledge that bridges the source and target domain has significant impact to model performance [39]. A popular approach is to adapt the pre-trained model for the new task through fine-tuning [20], or by adding new functions to specific tasks such as object detection [15]. In terms of EL, TLER [34] is a non-deep method that reuses and adopts seen data from the source domain to train models for the new domain. Auto-EM [40] proposes to pre-train models for both attribute-type (*i.e.*, schema) and attribute value matching based on word- and character-level similarity. However, Auto-EM assumes the typed entities are from a single data source and the attributes are seen during training, and thus cannot handle the multi-source scenario with unseen attributes. 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 [33], and models are trained on the same task [38]. Many well-designed algorithms propose to map the original feature spaces to a shared latent feature space between domains [3, 8]. DeepMatcher+ [19] 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. Another line of works proposes to pre-train models on the source and target domain (if labeling available) and then combine them through specific weighting schemes [31]. The process of applying the trained model to handle previously unseen data is also known as zero-shot learning [28]. Unlike the above approaches, 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.

Table 1: 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 $\mathbb{R}^{D \times F} \rightarrow \mathbb{R}^F$

3 PRELIMINARIES

In this section, we first formally define the problem, and then provide several key notions relevant to our proposed solution. Symbols and notations used in this paper are listed in Table 1.

3.1 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 $\mathcal{A}$ = {`title, artist, country`}. We start with the formal definition of entity linkage.

Problem 1 (EL: Entity Linkage). *Given two entity records r and r' associated with the same set of attributes $\mathcal{A}$ (schema), entity linkage aims to predict if r and r' refers to the same real-world entity.*

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 heterogeneous entity matching[1] [11, 27].

Problem 2 (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 each pair in $\{(r, r')\}_{unseen}$ represents the same real-world entity, where $(r, r')_{unseen} \in (\mathcal{S} \bigtimes \mathcal{S}') \cup (\mathcal{S}' \bigtimes \mathcal{S}')$, $|\mathcal{S}'| > |\mathcal{S}|$. Since $\mathcal{S} \neq \mathcal{S}'$, certain attributes in $\mathcal{A}'$ could be missing (**C1**), new (**C2**), or associated with values from different distributions (**C3**), and thus $\mathcal{A} \neq \mathcal{A}'$.*

The key notion in Problem 2 that is different from Problem 1 is that the linkage task is conducted on entity pairs sampled from a wider range of data sources than the labeled data used to train the model (ten or hundred orders of magnitude more in reality). Back to the example shown in Figure 1, while the trained model could make perfect prediction based on "Artist" only, it would fail to handle new records because the attribute "Artist" has missing or abbreviated values that contain less info. Moreover, the new data sources contain a rarely seen or unseen attribute ("Gender"). This issue can be addressed by aligning the union of ontology $\mathcal{A} \cup \mathcal{A}'$ with blank "dummy" attributes. Based on our definition, a solution

[1] 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.

to MEL should be able to **(G1)** make use of the massive unlabeled data from the new sources, and **(G2)** further improve the linkage performance by leveraging a few labeled record pairs from the new sources, if available (*i.e.*, an additional support set).

3.2 Terminology

Here we discuss the necessary terminology of our framework,.

Definition 3.1 (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 3.1, the seen and unseen set of data sources in Problem 2 are formulated as $\mathcal{S} = \mathcal{D}_S^*$ and $\mathcal{S}' = \mathcal{D}_T^*$. Besides, given a pair in the target domain, it could either contain one entity sampled from the seen data sources in $\mathcal{D}_S^*$ and the other one from the unseen, *i.e.*, $(r, r')_T \in \mathcal{D}_S^* \bigtimes \mathcal{D}_T^*$, or it has both entities sampled from the completely unseen data sources, *i.e.*, $(r, r')_T \in \mathcal{D}_T^* \bigtimes \mathcal{D}_T^*$. In both cases, achieving **G1** requires data in $\mathcal{D}_T$. To achieve **G2**, we introduce the support set.

Definition 3.2 (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 the target domain $\mathcal{D}_T^*$. It has at least one data source that is not contained in $\mathcal{D}_S^*$.

The support set corresponds to the real-world scenario that a few newly incoming entity pairs are well-labeled (*e.g.*, on-the-fly human annotation). Thus, entity pairs in $\mathcal{D}_S$, $\mathcal{D}_T$, as well as $\mathcal{S}_U$ are all required to achieve **G2** for MEL.

4 PROPOSED FRAMEWORK

We propose AdaMEL to address Problem 2, a deep framework that learns attribute importance as the transferable knowledge $\mathcal{K}$ (Section 4.1), and adapt it to multiple data-sources via domain adaptation . AdaMEL first extracts the contrastive relational features of entity pairs to derive the embeddings (Section 4.2). Then, by using the proposed attention embedding function f, AdaMEL projects features from $\mathcal{D}_S$ and $\mathcal{D}_T$ into the same attention space (Section 4.3), and jointly learns the feature importance for data sources in both $\mathcal{D}_S^*$ and $\mathcal{D}_T^*$. This process is conducted in an unsupervised or supervised domain adaptation manner (Section4.4), depending on the real-world scenario. The overview is depicted in Figure 2.

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 Section 3.1, the key difference between $\mathcal{D}_S$ and $\mathcal{D}_T$ lies in the

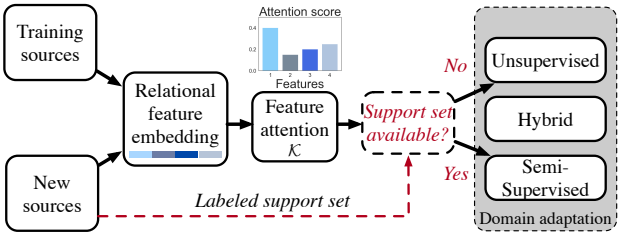


Figure 2: Overview. AdaMEL first embeds attributes for records from both the source and target domain to derive the feature representations, and uses the feature attention function to get the attention scores (importance) as the transferable knowledge $\mathcal{K}$. Then, depending on the availability of the labeled support set, AdaMEL uses $\mathcal{K}$ and performs either the unsupervised or semi-supervised manner of domain adaptation for MEL.

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) 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 leverages the attention mechanism to learn the importance of each textual attribute $A \in \mathcal{A}$ as the generic knowledge for transfer learning. However, instead of computing the attribute importance directly, AdaMEL 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 [37]. Taking the attribute $A =$"music version" as an example, a pair of music recordings sharing the same word (*i.e.*, "original" or "remix") is not as strong an identifier for matching as it would be for non-matching if one recording is "original" while the other is "remix". 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.

$$\begin{cases} sim(A) & = \{w\} \text{ for } w \in \{r[A] \cap r'[A]\} \\ uni(A) & = \{w\} \text{ for } w \in \{r[A] \cup r'[A] - r[A] \cap r'[A]\} \end{cases} \tag{2}$$

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 sums up the embeddings of the cropped word tokens [18, 26, 35] without using more sophisticated operations. The embeddings can be obtained using any pretraining

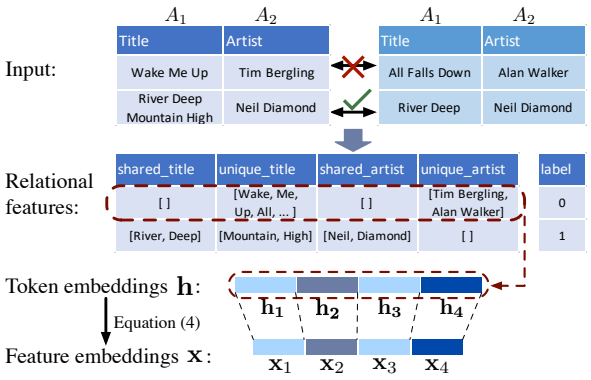


Figure 3: AdaMEL processes 1 attribute A as 2 relational features (*i.e.*, $sim(A)$ and $uni(A)$). In this example, $F = 4$ features are generated from $|\mathcal{A}| = 2$ attributes (*i.e.*, "Title" and "Artist"). The empty word tokens are embedded as the fixed normalized non-zero vector to form h (red dashed box). The feature embedding x is obtained through non-linear affine transformation of the token embedding h (Equation (4)). Each feature assumes to contribute independently to predict the linkage.

language model, such as BERT [20] or Fasttext [18]. 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:

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

By doing so, we denote the entity pairs (r, r') as F textual embedding features ($F = 2|\mathcal{A}|$) for matching. The complete process is depicted in Figure 3. Besides, AdaMEL leverages per-feature non-linear affine transformation to project the word embeddings to get the latent feature $\mathbf{x}$:

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_F] = [\sigma(\mathbf{V}_j \mathbf{h}_j + \mathbf{b}_j)] \text{ for } j = 1, \cdots, F \quad (4)$$

where $\mathbf{V}_j^{H \times D}$ is the learnable weight matrix, $\mathbf{b}_j^H$ is the learnable bias vector, and σ denotes the non-linear activation function (*e.g.*, Relu). With this representation, Equation (1) can be rewritten as: $y = M(\mathcal{K}, \mathbf{x}) \in \{0, 1\}$. Next we discuss how AdaMEL learns feature importance [2] as the transferable knowledge $\mathcal{K}$.

4.3 Feature Attention Embedding

Given a pair of entities denoted through F features, AdaMEL defines the energy score 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 $\mathbb{R}^{H'} \rightarrow \mathbb{R}$, 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. Formula in Equation (5) computes the attention score of feature j:

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

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

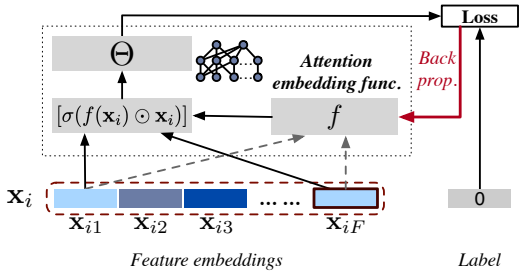


Figure 4: AdaMEL-base architecture that updates f via labeled data in $\mathcal{D}_S$. AdaMEL-base first computes the attention vector $f(\mathbf{x}_i)$ for the i−th entity pair (dashed line), and then compose it with the feature embeddings (solid line) as the input to the neural network Θ.

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 feed-forward neural network Θ to perform the binary classification task:

$$\hat{y} = \Theta(\sigma(f(\mathbf{x}) \odot \mathbf{x})) = \Theta([\sigma(g(\mathbf{x}_1) \cdot \mathbf{x}_1), \cdots, \sigma(g(\mathbf{x}_F) \cdot \mathbf{x}_F)]) \quad (7)$$

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:

$$L_{base} = -\frac{1}{N} \sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \quad (8)$$

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 $\mathcal{D}_S$ and illustrate the architecture in Figure 4. 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 $\mathcal{D}_T$ sampled from multiple data sources to alleviate this issue and make $\mathcal{K}$ data-source agnostic.

4.4 Domain Adaptation-based Variants

Based on AdaMEL-base, we propose three variants that leverage domain adaptation to handle different learning scenarios.

4.4.1 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 in Equation (9). At a specific iteration in the training, AdaMEL uses the up-to-date f to project data from both $\mathcal{D}_S$ and $\mathcal{D}_T$ into the same feature attention space. Then, AdaMEL updates $\mathbf{W}$ and $\mathbf{a}$ so that not only the cross-entropy loss introduced in Equation (8) is minimized, but also the KL divergence between feature attention distributions for $\mathcal{D}_S$ and $\mathcal{D}_T$. In this way, feature importance for entity records in $\mathcal{D}_S$ is jointly updated with records sampled from a wider range of data sources in $\mathcal{D}_T$, and thus being agnostic to previously unseen data sources that have significantly different value distribution (**C3**).

$$L_{\text{un}} = (1-\lambda)L_{\text{base}} + \lambda L_{\text{target}} \tag{9}$$

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 follows.

$$L_{\text{target}} = \text{KL}(f(\mathbf{x}), \bar{f}(\mathbf{x}')) = \sum_{i=1}^{|\mathcal{D}_S|}\sum_{j=1}^{F} \bar{f}(\mathbf{x}')_j \log(\frac{\bar{f}(\mathbf{x}')_j}{f(\mathbf{x}_i)_j}) \tag{10}$$

where $\bar{f}(\mathbf{x}')_j = \frac{1}{|\mathcal{D}_T|}\sum_{\mathbf{x}'_i \in \mathcal{D}_T} f(\mathbf{x}'_i)_j$, which 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 any labeled data in $\mathcal{D}_T$ and performs linkage in the zero-shot manner. This model also follows the design pattern in [12]. Figure 5 depicts the architecture and the algorithm is given in Algorithm 1. 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 transferable knowledge $\mathcal{K} = f(\mathbf{x}_i)$ for $\mathbf{x}_i \in \mathcal{D}_T$ (Line 12). Line 14- 15 denote the inference.

4.4.2 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 follows:

$$\mathbf{c}^+_{\mathcal{D}_S} = \frac{1}{|\mathcal{D}_S|}\sum_{(\mathbf{x}_i^+, y_i^+)\in\mathcal{D}_S} f(\mathbf{x}_i) \tag{11}$$

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 (12), 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

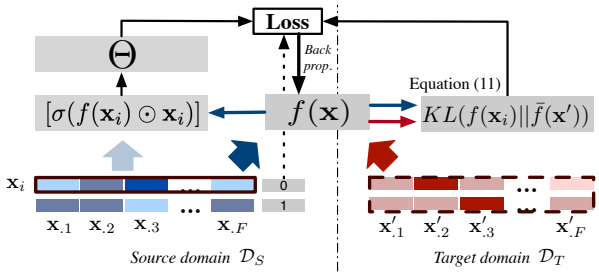


Figure 5: AdaMEL-zero architecture that attempts to align the i-th entity pair $f(\mathbf{x}_i)$ in $\mathcal{D}_S$ (solid box) with the averaged $f(\mathbf{x}')$ (dashed box) in $\mathcal{D}_T$. $\mathbf{x}_{.j}$ and $\mathbf{x}'_{.j}$ ($j = 1, \cdots, F$) denote the j-th feature in general from $\mathcal{D}_S$ and $\mathcal{D}_T$, respectively.

Algorithm 1 AdaMEL-zero

Input: $\mathcal{D}_S = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{D}_T = \{\mathbf{h}_i\}$, λ, 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: $L_{\text{un}} \leftarrow (1-\lambda)L_{\text{base}} + \lambda L_{\text{target}}$ ▷ Eq. (9)
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** $\hat{\mathbf{y}}$, $\mathbf{a}$, $\mathbf{W}$

Algorithm 2 AdaMEL-few

Input: $\mathcal{D}_S = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{S}_U = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{D}_T = \{\mathbf{h}_i\}$, ϕ, 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: $J \leftarrow J + \nabla L_{\text{base}}$ ▷ Eq. (8)
9: Form f with updated $\mathbf{a}, \mathbf{W}$ ▷ Eq. (6)
10: Compute $\mathcal{D}_S^+, \mathcal{D}_S^-, \bar{d}_{\mathcal{D}_S}^+, \bar{d}_{\mathcal{D}_S}^-$ ▷ Eq. (11)
11: $L_{\text{ssl}} \leftarrow L_{\text{base}} + \phi L_{\text{support}}$ ▷ Eq. (13)
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** $\hat{\mathbf{y}}$, $\mathbf{a}, \mathbf{W}$

entity pairs in $\mathcal{S}_U$ that are deviating from seen data sources.

$$L_{\text{support}} = \sum_{y_i=1} \frac{d(f(\mathbf{x}_i^+), \mathbf{c}_{\mathcal{D}_S}^+)}{\bar{d}_{\mathcal{D}_S}^+} \log \hat{y_i} + \sum_{y_i=0} \frac{d(f(\mathbf{x}_i^-), \mathbf{c}_{\mathcal{D}_S}^-)}{\bar{d}_{\mathcal{D}_S}^-} \log(1-\hat{y_i}) \tag{12}$$

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 follows:

$$L_{\text{ssl}} = L_{\text{base}} + \phi L_{\text{support}} \tag{13}$$

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. Particularly, 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 $\mathcal{S}_U$.

4.4.3 Hybrid Model. We further propose a hybrid model that incorporates both the labeled support set as well as the unlabeled data in the target domain in the training process. It can be seen as the composition of AdaMEL-zero and AdaMEL-few. The loss function is as follows.

$$L_{\text{hybrid}} = (1 - \lambda) L_{\text{base}} + \lambda L_{\text{target}} + \phi L_{\text{support}} \tag{14}$$

This variant uses the loss L_{target} defined in Equation (10) and L_{support} defined in Equation (12). We name this hybrid solution as **AdaMEL-hyb**. The algorithm is similar to Algo. 2, the main difference is to incorporate L_{un} *i.e.*, Equation (9) into the training process (line 7- 8) to learn the parameters simultaneously.

4.5 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 $O(FDH)$ learnable parameters. For (ii), as $\mathbf{W}^{H' \times H}$ and $\mathbf{a}^{H'}$ are shared across all features, there are totally $O(HH')$ 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 $O(FDH + HH' + FH'H_{\text{hidden}})$ parameters to learn. We discuss the setup values of H, H' and H_{hidden} in the configuration of Section 5, and empirically estimate the parameter number in Section 5.6.

5 EXPERIMENTS

In this section we describe the experiments to evaluate properties and the performance of AdaMEL. Specifically, we aim to answer the following research questions: **Q1.** Does AdaMEL effectively handle MEL with the data challenges (**C1**-**C3**) under the transfer learning paradigm? **Q2.** How well does AdaMEL adapt feature importance in the target domain and how does it affect the linkage results? **Q3.** Are generated feature attention values meaningful? **Q4.** How stable is AdaMEL in handling different data sources? **Q5.** How does the size of support set $\mathcal{S}_U$ impact the performance of AdaMEL? We conclude with the model justification (ablation study, limitation).

5.1 Experimental Setup

Data In accordance with (**Q1**-**Q5**), we use both the public benchmark dataset from the Data Integration to Knowledge Graphs (DI2KG) challenge [5] and two real-world datasets in different scales from an online-sales company. Both datasets are in the tabular form and the entities are associated with descriptive textual features. The data statistics and source info is given in Table 2.

- **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 following the hyperlinks in pages, so there might be mixed-type errors such as 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.

Table 2: 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

Table 3: Train, support and test statistics in the experiments.

Data	Entity_type	Train $\lvert\mathcal{D}_S\rvert$	Support $\lvert\mathcal{S}_U\rvert$	Test $\lvert\mathcal{D}_T\rvert$
Music-3K	Artist	374	100	541
	Album	490	100	509
	Track	314	100	542
Music-1M	Artist	298 566	100	541
	Album	697 739	100	509
Monitor	Monitor	17 766	100	1 432

Comparing with the public benchmark datasets [26], the above datasets are collected from larger ranges of real-world data sources with heterogeneous schemas. 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 Music-3K, artist type, 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 benchmark "dirty" and "heterogeneous" Walmart-Amazon dataset [11], 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 further inspects whether the pair of music works indicate the same physical copy (*i.e.*, "Album"), or 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 pairs being unmatched. Additionally, less than 50% entity pairs in this dataset have non-missing values for most attributes. Non-missing pairs of 5 attributes only exist in the target domain. These data challenges do not exist in the benchmark datasets [26], and we detail the analysis in Section A.2 of the supplementary material. All the above issues make the datasets more challenging and closer to the real-world knowledge integration scenario.

Baselines. The following baselines are used in this work.

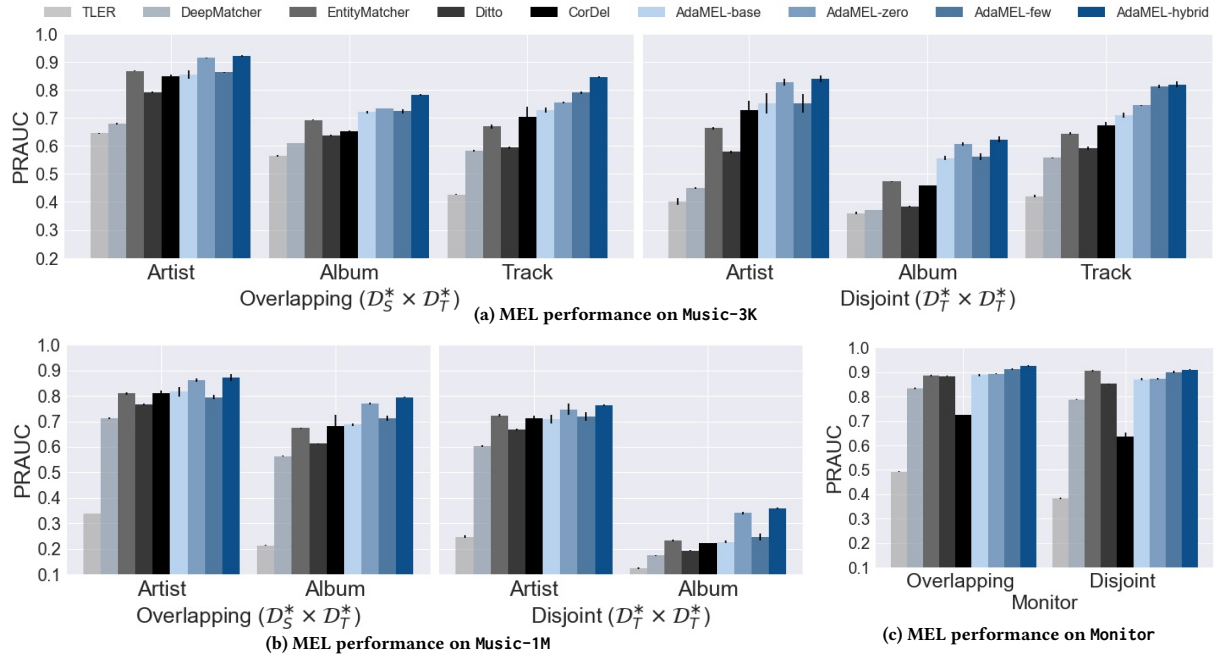
- **TLER** [34] is a non-deep transfer learning framework that defines a standard feature space and reuses the seen data to train models for the new domain.
- **DeepMatcher** [26] is a deep learning framework that consists of 3 modules: attribute embedding, attribute similarity representation, and classification. The public implementation uses Fasttext to embed attribute words and uses attentative RNN to summarize attributes. We report results using the best-performing variant, DeepMatcher-hybrid.
- **EntityMatcher** [11] is a hierarchical deep framework for heterogeneous schema matching. It jointly matches entities at the level of token, attribute, and entity. The token-level matching strategy allows EntityMatcher to perform cross-attribute alignment. Fasttext is used to embed word tokens.
- **Ditto** [23] is an EL system that leverages fine-tuned, pretrained Transformer-based language models (*i.e.*, BERT, DistilBERT, or RoBERTa) with optimization including domain knowledge injection, text summarization, and data augmentation with difficult samples.
- **CorDel** [37] adopts an alternative deep architecture to the widely-used "twin architecture". It compares and contrasts word tokens to filter out minor deviations between attribute values before embedding. CorDel also uses Fasttext, and it shows higher performance with reduced runtime compared to DeepMatcher. Out of the variants, CorDel-Attention is reported to perform the best on dirty EL datasets.

We consider these baselines since they are reported to achieve state-of-the-art EL performance and outperform methods such as Seq2SeqMatcher [27] and DeepMatcher+ [19]. Most of them are particularly proposed to handle heterogeneous entity linkage.

Configuration. In our experiments, we follow the original paper and fine-tune the baselines for optimal performance. The statistics of training, support and testing data is given in Table 3. Specifically, Musci-1M shares the same testing set as Musci-3K. Monitor adopts all positive and randomly selected 1000 negative pairs to form the testing set (see Section A.1 of the appendix for more details). For DeepMatcher, we use its hybrid variant (bi-directional RNN with attention) to summarize attributes with 2-layer highway neural network ((hidden dim= 300)). 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 that learns the word importance 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 3 baselines use the pretrained FastText [18] 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}.

To evaluate the effectiveness of our proposed framework, we configure AdaMEL with consistent setup as the baselines. Specifically, we use the 300-dim Fasttext to embed word tokens for fairness because 3 of the 4 baselines also use it, even though AdaMEL supports any word embedding techniques such as Bert embedding [20]. We set the cropping size= 20 as CorDel. The hyparameters of AdaMEL are given as follows: the dimension of the projected embeddings per feature is $H = 64$, the dimension of the hidden layer in f is $H' = 256$, and the dimension of the hidden layer in Θ is $H_{\text{hidden}} = 256$. The activation σ is set to be Relu. We set $\lambda = 0.98$ and $\phi = 1.0$ in Equation (9), (13) and (14) for AdaMEL variants unless otherwise addressed. To train the AdaMEL, we use Adam optimizer [21] for 100 epoches with learning rate = 10^{-4} and batch size = 16. 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.

Evaluation Metric. We evaluate the model performance using PRAUC as it measures the precision-recall relation globally and handles data imbalance. We use the python Sklearn library to compute PRAUC based on the open-source implementation of all baselines.



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

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

(c) MEL performance on `Monitor`

Figure 6: MEL performance (PRAUC) comparison between AdaMEL variants and baselines. AdaMEL variants outperform baseline heterogeneous entity matching methods in almost all cases. Particularly, AdaMEL-hyb performs the best on all entity types and datasets.

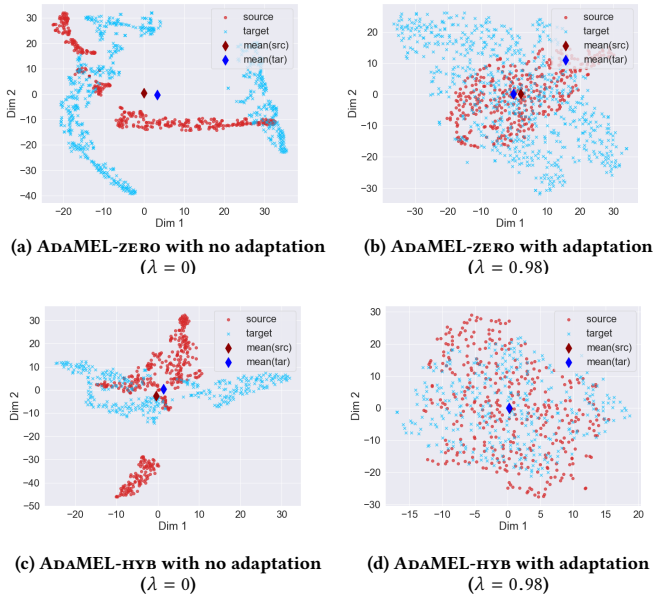
5.2 Transfer Learning for MEL

Our first experiment is to verify the effectiveness of AdaMEL variants on the task of MEL (**Q1**). 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.*, $(r, r')_T \in \mathcal{D}_S^* \bigtimes \mathcal{D}_T^*$), and **(S2)** data sources in the target domain are disjointed from the source domain (*i.e.*, $(r, r')_T \in \mathcal{D}_T^* \bigtimes \mathcal{D}_T^*$).

Setup. For the `Music` data, we use three data sources (*i.e.*, $\mathcal{D}_S^* = \{$*website 1*, *website 2*, *website 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 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 also randomly picked different sources to form $\mathcal{D}_S$ and $\mathcal{D}_T$ to eliminate the randomness, and found similar patterns in the results.

Results. We report the results in Figure 6 with complete numerical results in Table 9 and 8 of Section A.3 in the supplementary material. Our first observation is that all AdaMEL variants tend to outperform the baseline methods and our base model without adaptation, AdaMEL-base. The heterogeneous entity matching baselines do not perform well on these datasets under the supervision of labeled data only. This is likely because of the long and noisy word sequences in the data and the difference in attribute value distribution across data sources that is unseen during model training. AdaMEL highlights the impact of important features, and only represent the sequences by summing the token embeddings. This confirms our conjecture that learning the attribute-level attention as the transferable knowledge is more effective in handling the MEL task than refining the word-level sequence representation. Also, we observe that out of all variants, AdaMEL-hyb achieves the best performance in all cases with 0.64% ~ 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. On the contrary, since the records are more consistent for the "Artist" and "Album" type, incorporating more records in $\mathcal{D}_T$ leads to higher improvement in MEL performance. Note Figure 6b shows that AdaMEL-few performs slightly worse than AdaMEL-base because the labeled samples from $\mathcal{S}_U$ only overfits to the trained model on the source domain, that deviates the actual feature importance for the massive unlabeled samples in $\mathcal{D}_T$. To summarize, the improvement of AdaMEL-zero, -few and -hyb 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**). This is as expected as the disjoint scenario represents an extreme case where data sources in $\mathcal{D}_T$ are less likely or even entirely not used in training the model if the support set is unavailable. 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



(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 ($\lambda = 0.98$)

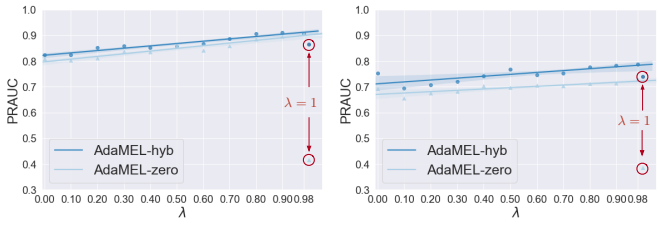
Figure 7: Source and target domain feature attention vectors are better aligned with high value of λ for both AdaMEL-few and AdaMEL-hyb (visualized with TSNE, dim=2).

distinguish the actual media of the music work (*i.e.*, the physical or digital copy). As the models are tested on the same well-labeled set, training on `Music-1M` could be vulnerable to cases such as mixed-type matching. Nevertheless, we observe that AdaMEL still achieves promising results in both hard cases of transfer learning for MEL, *i.e.*, unseen data sources in the target domain and training on weakly labeled data, which further demonstrates the advantage of AdaMEL. Table 8 gives the result on `Monitor`. Similarly, 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. On average, AdaMEL-hyb outperforms the baseline by 9.39% improvement in the overlapping scenario and 11.55% improvement in the disjoint scenario. These results also validates our findings above.

5.3 Effectiveness of Adaptation

Adaptation is the key component to our proposed method. 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 training (larger λ leads to more adaptation).

Setup. We run both variants of AdaMEL on the `Music-3K` dataset and report the performance on MEL. As discussed in Section 4.4.1, 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 from both domains should align well, and further benefit the linkage task. To validate this conjecture, we visualize the learned feature attention vectors using AdaMEL-zero and AdaMEL-hyb with different values of λ by projecting them into 2-dimensional space using TSNE [25]. We also study the linkage



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

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

Figure 8: 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.

performance of AdaMEL-zero and AdaMEL-hyb with different λ values on the "artist" and "album" type of the `Music-3K` dataset.

Results. In Figure 7, 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 7b), AdaMEL-hyb (Figure 7d) 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.

To evaluate the impact of adaptation to the linkage results, in Figure 8 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.

5.4 Attention Analysis

Setup. To testify whether AdaMEL learns meaningful feature attention values (**Q3**), 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 and omit the other two types for brevity. AdaMEL-hyb is configured with the best performance ($\lambda = 0.98, \phi = 1.0$) in the previous experiments.

Result. For the `Monitor` dataset in Table 4, 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).

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

Table 4: AdaMEL learned importance of top-5 features for Monitor and Music-3K, artist type.

Monitor		Music-3K, artist	
Feature	Score	Feature	Score
Page_title_shared	0.1635	Main_performer_shared	0.0739
Page_title_unique	0.0595	Name_unique	0.0697
Source_shared	0.0535	Name_shared	0.0628
Manufacturer_unique	0.0473	Source_unique	0.0597
Manufacturer_shared	0.0416	Name_Native_Language_shared	0.0583

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 5. 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). Also, they shows the effectiveness of the feature attention module of AdaMEL in learning reasonable feature importance.

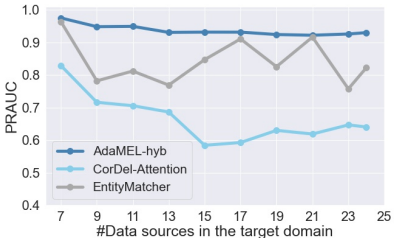
Table 5: Performance (PRAUC) comparison using the selected important features vs. the other features and all features. Numbers in the parenthesis denote the counts of features.

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

5.5 Data Sources Analysis

In this experiment we simulate the real-world knowledge integration, where new data sources often arrive one by one incrementally (such as in batches from neighboring data sources), and testify the stability of AdaMEL in handling the various data sources under this scenario (**Q4**) .

Setup. We use the public Monitor dataset and compare AdaMEL-hyb with the optimal configuration ($\lambda = 0.98, \phi = 1.0$ as shown in Section 5.2 and Section 5.3) with the best-performing baseline approach, EntityMatcher, and the fastest baseline approach, CorDel-Attention. In this experiment, we use 1500 entity pairs from the same 5 data sources as mentioned in Section 5.2 to train the models (*i.e.*, $\mathcal{D}_S^*$ = {*ebay.com*, *catalog.com*, *best-deal-items.com*, *cleverboxes.com*, *pcpartpicker.com*}). To test the performance on MEL, we first randomly select 200 entity pairs from each of 7 data sources (the same 5 data sources as $\mathcal{D}_S^*$ and 2 unseen ones, *i.e.*, $\mathcal{D}_T^* = \mathcal{D}_S^* \cup$ {*yikus.com*, *getprice.com*}) and form totally 1400 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 $\mathcal{D}_T^* = \mathcal{D}_T^* \cup \Delta\mathcal{D}_T^*$. Each of the newly added pairs $\{(r, r')\}$ contains at least one record from $\Delta\mathcal{D}_T$ to ensure new data sources are introduced to the target domain. As AdaMEL-hyb requires a small set of labeled entity pairs from $\mathcal{D}_T^*$, we randomly select 100 labeled samples from all data sources ($\mathcal{D}_S^* \cup \mathcal{D}_T^*$). This small set simulates the on-the-fly manual labeling in the real-world, and we fix it throughout each run of the experiment to ensure the impact of $\mathcal{S}_U$ is consistent. We also record the average runtime over all runs as an empirical study of the model efficiency.



Method	Runtime (s)
Hybrid	319.20 ± 7.20
CorDel	906.19 ± 46.35
E-Matcher	2500.43 ± 17.56

Figure 9: AdaMEL-hyb performs more stably (0.9750 ∼ 0.9219 in PRAUC) as #data sources increases in $\mathcal{D}_T$ with less runtime.

Results. We report the performance of AdaMEL-hyb and the two baselines on MEL in Figure 9, as well as their empirical runtime. As shown in the figure, 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. Moreover, based on the table in Figure 9, AdaMEL-hyb takes much less time to train than CorDel-Attention and EntityMatcher. The empirical runtime comparison corresponds to our analysis in Section 4.5 as AdaMEL-hyb does not require sophisticated operations on word-level embeddings and thus having relatively less parameters to train. In practice, the number of parameters to train for AdaMEL-hyb is ∼ 2 219 520, which is much less than the number given by EntityMatcher: ∼ 123 119 104. These findings demonstrate the capability of AdaMEL in consistently handing MEL with a variety of incoming data sources, while being more robust. In addition, they strengthen our claim that finding important features as the transferable knowledge in MEL could benefit the model performance with reduced computational complexity.

5.6 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.

Result. The experimental result is shown in Figure 10. Our first observation is that at the initial stage of the experiment, the performance of both AdaMEL-few and AdaMEL-hyb improves as

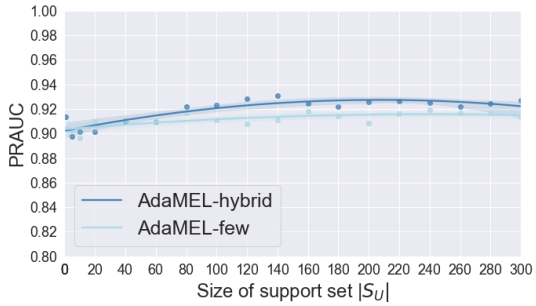


Figure 10: Sensitivity analysis of the size of support set $|\mathcal{S}_U|$ fitted with order-2 polynomial regression on AdaMEL-few and AdaMEL-hyb. As more labeled samples are included in $\mathcal{S}_U$, the model performance (PRAUC) increases initially and then flattens out.

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 10 indicates that a small support set with $|\mathcal{S}_U| = 100 \sim 200$ labeled samples from $\mathcal{D}_T$ is beneficial to learn feature 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.

5.7 Model Justification

In this section we run experiments to justify the design choices of AdaMEL and its limitation.

5.7.1 Ablation Study. We perform the ablation study of AdaMEL that uses the shared and unique contrastive features, as well as using both of them as the default setting. Table 6 shows that including the shared and unique attribute values capture different perspectives of the data and thus enriches the feature space. Including both achieves the highest performance with $0.41\% - 6.72\%$ improvement over using one feature.

5.7.2 Performance on Single Domain. Here we compare AdaMEL-zero and -hyb with DeepMatcher on the benchmark datasets to justify their performance on well-labeled data from the same seen domain without the 3 challenges (**C1** - **C3**). From Table 7, we observe that AdaMEL-zero does not perform as well as DeepMatcher on these benchmark datasets of one single domain. This shows the limitation of AdaMEL in handling data with no missing values or schema difference. The reason is likely due to the simplicity of AdaMEL architecture, as it aims to learn the data-source-level feature importance instead of improving the token-level embeddings as DeepMatcher or its variants. In the real-world knowledge integration process where data distributions are highly heterogeneous, transferring these token-level contextualized embeddings brings extra computation and does not always generalize well, as shown in Section 5.2. Nevertheless, even though AdaMEL is designed to handle data challenges in practice (**C1-C3**), we observe that AdaMEL-hyb performs comparably as DeepMatcher with reduced model complexity, which shows its effectiveness of adaptation.

Table 6: Ablation study: AdaMEL contrastive features on Music-3K, artist and album type. AdaMEL-zero and -few perform similarly.

Dataset	Method	Shared	Unique	Shared & Unique
Music-3K, artist	AdaMEL-base	0.7868 ± 0.0045	0.7170 ± 0.0132	**0.8545 ± 0.0143**
	AdaMEL-hyb	0.8539 ± 0.0026	0.8069 ± 0.0112	**0.9211 ± 0.0040**
Music-3K, album	AdaMEL-base	0.7163 ± 0.0048	0.5520 ± 0.0044	**0.7204 ± 0.0033**
	AdaMEL-hyb	0.7504 ± 0.0059	0.5879 ± 0.0028	**0.7833 ± 0.0031**

Table 7: Entity linkage performance (F1) of DeepMatcher, AdaMEL-zero and -hyb on the benchmark datasets, single domain scenario. AdaMEL-hyb performs comparably as DeepMatcher.

Type	Datasets	Domain	DeepMatcher	AdaMEL-zero	AdaMEL-hyb
Structured	Amazon-Google	Software	69.3	60.2	65.1
	Beer	Product	78.8	78.6	82.8
	DBLP-ACM	Citation	98.4	98.7	98.9
	DBLP-Google	Citation	94.7	93.1	93.5
	Fodors-Zagats	Restaurant	100	90.0	99.8
	iTunes-Amazon	Music	91.2	91.2	98.7
	Walmart-Amazon	Electronics	71.9	57.8	66.7
Dirty	DBLP-ACM	Citation	98.1	95.7	97.7
	DBLP-Google	Citation	93.8	89.7	91.5
	iTunes-Amazon	Music	79.4	79.3	80.7
	Walmart-Amazon	Electronics	53.8	48.2	52.2

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 also 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 on average 8.21% improvement than the baselines based on supervised learning in PRAUC score for the multi-source entity linkage task. In addition, our experiments demonstrate the effectiveness of AdaMEL in adaptation, provide the analysis on the learned feature attention, and study the impact of different data sources as well as the size of the support set. Future directions include combining our work with advanced NLP techniques for sequence representation in attribute summarization to further improve the model performance in MEL, and extending our framework to handle data sources in different languages.

REFERENCES

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

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

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

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

[5] di2kg. 2020. 2nd International Workshop on Challenges and Experiences from Data Integration to Knowledge Graphs. http://di2kg.inf.uniroma3.it/2020/.

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

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

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

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

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. PMLR, 1126–1135.

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

[12] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*. PMLR, 1180–1189.

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

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

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

[16] Di Jin, Mark Heimann, Ryan A Rossi, and Danai Koutra. 2019. node2bits: Compact Time-and Attribute-aware Node Representations for User Stitching. In *ECML-PKDD*. Springer, 483–506.

[17] Muhammad Ebraheem Saravanan Thirumuruganathan Shafiq Joty and Mourad Ouzzani Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proceedings of the VLDB Endowment* 11, 11 (2018).

[18] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of ACL: Volume 2, Short Papers*. 427–431.

[19] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource Deep Entity Resolution with Transfer and Active Learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5851–5861.

[20] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

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

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

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

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on EMNLP*. 1412–1421.

[25] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

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

[27] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. 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.

[28] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. 2009. Zero-shot learning with semantic output codes. *NIPS* (2009), 1410–1418.

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

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

[31] Gabriele Schweikert, Gunnar Rätsch, Christian Widmer, and Bernhard Schölkopf. 2008. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. *Advances in neural information processing systems* 21 (2008), 1433–1440.

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

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

[34] Saravanan Thirumuruganathan, Shameem A Puthiya Parambath, Mourad Ouzzani, Nan Tang, and Shafiq Joty. 2018. Reuse and adaptation for entity resolution through transfer learning. *arXiv preprint arXiv:1809.11084* (2018).

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

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[37] Zhengyang Wang, Bunyamin Sisman, Hao Wei, Xin Luna Dong, and Shuiwang Ji. 2020. CorDEL: A Contrastive Deep Learning Approach for Entity Linkage. *arXiv preprint arXiv:2009.07203* (2020).

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

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

[40] Chen Zhao and Yeye He. 2019. Auto-EM: End-to-end Fuzzy Entity-Matching using Pre-trained Deep Models and Transfer Learning. In *WWW*. 2413–2424.

A SUPPLEMENTARY MATERIALS

A.1 Public Data Processing

In this section we detail the processing of the public dataset, `Monitor` that is used in the experiment. We follow the *'monitor_label.csv'* file (1 073 positive pairs and 110 082 negative pairs) from the DI2KG to create the labeled entity pairs that fall into the 24 data sources we are interested in. The resultant dataset, `Monitor` has 734 positive and 66 061 negative pairs. The source domain $\mathcal{D}_S^*$ contains 5 data sources (*i.e.*, $\mathcal{D}_S^* = \{$*ebay.com*, *catalog.com*, *best-deal-items.com*, *cleverboxes.com*, *ca.pcpartpicker.com*$\}$), which contains 302 positive pairs. Thus, the test data includes all the remaining positive 432 pairs and randomly-selected 1, 000 negative pairs.

We provide the code and the splitted public `Monitor` data in the following in the repo https://github.com/DerekDiJin/AdaMEL-supplementary.

A.2 Analysis of Public Data

To illustrate the data challenges (**C1**-**C3**), we provide detailed analysis of the `Monitor` data. We first study the difference of the source and target domain in terms of the attributes, *i.e.*, missing attribute values (**C1**) and new attributes (**C2**). As the attributes associated with entity pairs, we plot the percentage of pairs without missing values per attribute, *i.e.*, $(r[A], r'[A])$ where $r[A] \neq, r'[A] \neq$ for $A \in \mathcal{A}$ in both the source and target domain. This metric also indicates the difference of data source attributes because for unseen incoming attributes, at least one entities in a pair should have the missing value. The result is depicted in Figure 11. Ideally, the percentage bars included in this plot should all be close to 1, and this holds for both data in the source and target domain. In fact, we observe this pattern in the benchmark datasets [26] (such as Beer, DBLP-ACM, etc.), which indicates few missing values and no significantly different attributes. For the `Monitor` dataset, however, the pattern is different. We first observe that only 2 attributes (*i.e.*, "page_title" and "source") are close-to-1, while for all the remaining 11 attributes, less than 50% entity pairs have complete attribute values. Such data sparsity reflects the challenge (**C1**). In addition, we observe that the percentages of pairs without missing values are significantly different for the source and target domain. Particularly, we find that there are 5 out of 13 attributes only have non-missing entity pairs only in the target domain, which can be seen as new attributes (**C2**).

To illustrate the different attribute value distribution (**C3**), we showcase the attribute value distribution of one attribute, "prod_type" as the representative. We plot the frequency distribution of 10 most frequently appearing word tokens. As shown in Figure 12, the distributions of attribute "prod_type" are quite different between the source and target domain, which indicates the challenge we attempt to address.

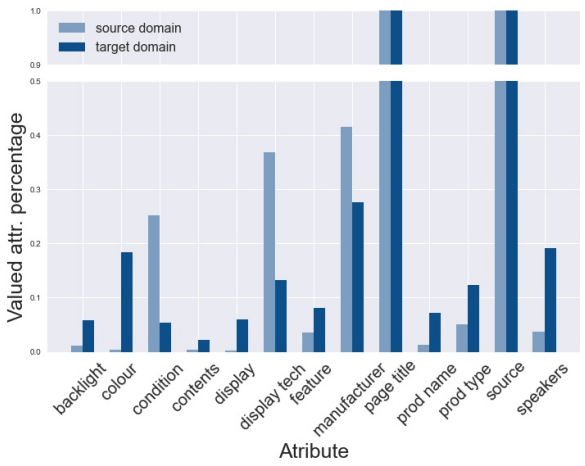
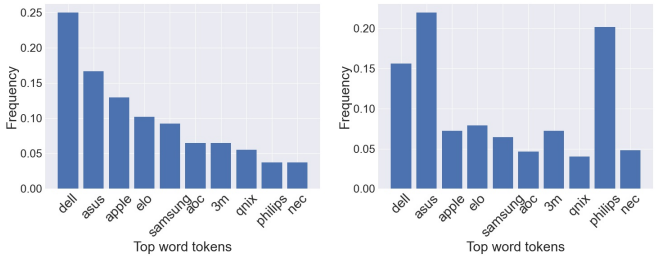


Figure 11: `Monitor`: the challenges of missing attribute values (C1) and new attributes (C2) between $\mathcal{D}_S$ and $\mathcal{D}_T$ is shown with the percentages of entity pairs without missing values per attribute (*i.e.*, nonempty for both entities). For most attributes, the majority of entity pairs have at least 1 entity with missing values. 5 out of 13 attributes have non-missing entity pairs only in the target domain (2 non-missing attributes are "page title" and "source"). For the remaining 6 attributes, the percentage is also significantly different between the source and target domain.



(a) Frequency of top 10 word tokens in the source domain

(b) Frequency of top 10 word tokens in the target domain

Figure 12: `Monitor`: the challenges of different attribute value distribution (C3) shown with the representative attribute "prod_type". The frequency distribution of top 10 word tokens under this attribute is significantly different between the source and target domain.

A.3 Complete Experimental Results

In this section we provide the complete numerical result shown in Section 5.2. Table 8 records the results on the `Monitor` dataset. The results on the `Music` dataset are shown in Table 9.

Table 8: AdaMEL performance (PRAUC) on `Monitor`. All variants outperform the baseline, AdaMEL-hyb performs the best (marked in bold) with at least 0.51% improvement over the second-best (∗).

	Method	Overlapping	Disjoint
`Monitor`	TLER	0.4932 ± 0.0028	0.3837 ± 0.0033
	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**

A.4 AdaMEL-hyb Algorithm

Here we provide the detailed algorithm of AdaMEL-hyb, shown in Algorithm 3.

Algorithm 3 AdaMEL-hyb

Input: $\mathcal{D}_S = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{S}_U = \{(\mathbf{h}_i, y_i)\}$, $\mathcal{D}_T = \{\mathbf{h}_i\}$, ϕ, B
Output: Predicted $\hat{y}_i$ for $\mathbf{h}_i \in \mathcal{D}_T$, updated $\mathbf{a}, \mathbf{W}$

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

Table 9: 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% ∼ 5.50% improvement over the second-best variant (marked with ∗) in PRAUC. AdaMEL-hyb outperforms the best-performing baselines (including AdaMEL-base) with 8.21% improvement on average.

	Method	Overlapping ($\mathcal{D}_S^* \bigtimes \mathcal{D}_T^*$)			Disjoint ($\mathcal{D}_T^* \bigtimes \mathcal{D}_T^*$)		
		Artist	Album	Track	Artist	Album	Track
`Music-3K`	TLER	0.6454 ± 0.0021	0.5655 ± 0.0032	0.4263 ± 0.0011	0.4014 ± 0.0121	0.3605 ± 0.0033	0.4203 ± 0.0042
	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`	TLER	0.3384 ± 0.0013	0.2128 ± 0.0019		0.2465 ± 0.0052	0.1237 ± 0.0031	
	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**	