Efficient Mixture of Experts based on Large Language Models for Low-Resource Data Preprocessing

Anonymous Author(s)

ABSTRACT

Data preprocessing (DP) that transforms erroneous and raw data to a clean version is a cornerstone of the data mining pipeline. Due to the diverse requirements of downstream tasks, data scientists and domain experts have to handcraft domain-specific rules or train ML models with annotated examples, which is costly/time-consuming. In this paper, we present MELD (Mixture of Experts on Large Language Models for Data Processing), a universal solver for low-resource DP. MELD adopts a Mixture-of-Experts (MoE) architecture that enables the amalgamation and enhancement of domain-specific experts trained on limited annotated examples. To fine-tune MELD, we develop a suite of expert-tuning and MoE-tuning techniques, including a retrieval augmented generation (RAG) system, meta-path search for data augmentation, expert refinement and router network training based on information bottleneck. To further verify the effectiveness of MELD, we theoretically prove that MoE in MELD is superior than a single expert and the router network is able to dispatch data to the right experts. Finally, we conducted extensive experiments on 19 datasets over 10 DP tasks to show that MELD outperforms the state-of-the-art methods in both effectiveness and efficiency. More importantly, MELD is able to be fine-tuned in a low-resource environment, *e.g.,* a local, single and low-priced 3090 GPU.

KEYWORDS

Mixture of Expert, LLMs, Data Preprocessing, Low-resource, Database

ACM Reference Format:

Anonymous Author(s). 2018. Efficient Mixture of Experts based on Large Language Models for Low-Resource Data Preprocessing. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 17 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Data Preprocessing (DP) tasks, including the discovery, extraction, transformation, cleaning, and integration of data from diverse sources, are crucial for a broad spectrum of organizations [1, 44]. Over the past decades, the focus has predominantly been on a limited number of tasks such as error detection (ED) [80, 96], data cleaning (DC) [79], data imputation (DI) [83], entity matching (EM) [73], entity linking (EL) [24], relation extraction (RE) [27], and column type annotation (CTA) [27, 32]. A primary challenge in this field arises from the diverse data distributions and requirements across various applications, each of which deals with unique issues such as errors, anomalies, matches, and necessitates the need of specific features or rules for detection, repair, and alignment.

The advent of large language models (LLMs), such as GPT-3 [25] and open-source LLaMa [112], has introduced a paradigm shift in addressing DP challenges. These models, typically adopting a decoder-only Transformer architecture, have demonstrated remarkable capabilities in DP tasks [87, 127, 128]. The effectiveness of LLMs in DP can be attributed to several inherent characteristics, including (1) natural language instructions of inputs and outputs, (2) few-shot learners, and (3) rich prior knowledge. It is noteworthy that LLMs obeys scaling laws[61], *i.e.,* more parameters gives better generative abilities and universal performance in DP. Consequently, most existing universal LLM-based DP solutions [9, 71, 87, 128] heavily rely on querying online GPT APIs. However, this approach encounter issues of stability and data privacy in certain scenarios because DP handles private data of enterprises in practice in most of the time and it is impossible for enterprises or governments to send their core datasets to GPT APIs. Another limitation is the difficulty in adapting these online models to highly specialized domains. In such cases, fine-tuning LLMs, such as GPT-3.5 or GPT-4, becomes a necessity, albeit a costly and sometimes infeasible one [71].

Considering the needs that online LLMs cannot fit, we focus on open-source LLMs with ≤7B parameters, that can be deployed locally in low-resource environment. However, by constraining the parameters of LLMs for universal DP, we face several challenges:

- The capability for a single model to learn representations across domains is inherently upper limited, even with more parameters.
- It is hard to leverage the world knowledge in LLMs, *i.e.,* knowledge learned from large corpus in the pre-training stage, for fine-tuning on few-shot data, leading to potential model over-fitting.
- Because task subspaces of DP tasks are discrete and far away from each other, traditional methods, *e.g.,* multi-task learning, are hard to work well for intrinsic task subspace identification.

To address these challenges, we revisit the Mixture of Experts (MoE) architecture, powered by the recent advancement of LLMs. Intuitively, an MoE [58] comprises a set of experts (*i.e.,* neural networks) and a trainable gating mechanism (*i.e.,* a router network). The gate assigns weights to the experts and the MoE model produces a weighted combination of experts' responses as the output. This weighting mechanism allows each expert to specialize in distinct segments of the input space, reducing training/inference costs.

Although MoE has been extensively studied over the past decades, the recent advent of LLMs has necessitated a revisit on MoE. Several pioneering studies [3, 60] have shown that models with sparsely activated MoE layers can significantly reduce the computational cost, where sparsely activated MoE layers are neural network layers that consist of multiple expert models and only a

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

subset is activated. MoE advocates that language models can be segmented into specialized sub-models or "experts", each of which focuses on different aspects of input. This approach enables efficient computation and resource allocation. Moreover, MoE facilitates the information/parameters sharing between tasks, to enhance generalizability by leveraging the inter-connection between tasks [6].

Different from existing MoE based models [3, 60], which embed a sparse gate network on the model parameters, we propose MELD (Mixture of Experts on Large Language Models for Data Preprocessing), an open-source LLM-based MoE system as a universal task solver for *low-resource DP* that DP task is addressed in resource-constrained environments with limited labeled data. MELD adopts a standalone router network, which allows independent and domain-specific expert training, and flexible plug-in design of experts during inference.

In training, MELD first employs a serializer to transform raw data from various sources into a standardized representation with task-specific prompts. Then an enhanced Retrieval-Augmented Generation (RAG) system is used to retrieve similar instances across domains, generating self-annotations for each instance as training data. MELD also incorporates heuristic methods to identify effective meta-paths for guided data augmentation with multiple experts (as illustrated in Figure 1). Alongside this, a set of experts is trained using parameter-efficient fine-tuning (PEFT) methods, addressing the scarcity of annotated data, where PEFT involves fine-tuning with a small number of model parameters. Finally, a standalone router network is trained to allocate the top-k most relevant experts for each input.

Contributions. Our major contributions are listed as follows:

- We present a uniform framework for DP, integrating multiple DP tasks and datasets into a standardized representation.
- We prove the error bounds for domain adaptations across DP tasks and the convergence of the MoE design across domains.
- We present an enhanced RAG system, along with a meta-path selection mechanism, which efficiently retrieves and generates effective examples across domains, facilitating the training of experts that exhibit both generalizability and robustness.
- We design effective MoE that could be fine-tuned in low resources, *e.g.,* a RTX 3090 GPU. Also we dynamically assign the top-k experts for inputs across domains, ensuring data security, domain generalizability, and feasibility for further fine-tuning.
- Extensive experiments were conducted on 19 datasets over 10 DP tasks. Benefiting from MoE, MELD demonstrates superior few-shot performance, particularly in cross-domain/task scenarios.

The rest of this paper is organized as follows. Section 2 introduces the preliminary and the problem definition. Section 3 prove the error bounds for domain adaptations across various DP tasks and the convergence of MoE. Section 4 presents MELD, by delving into the data preparation, efficient expert training and router network training. Section 5 shows the experimental results. After discussing the related works in Section 6, Section 7 concludes this paper.

2 PRELIMINARY AND PROBLEM DEFINITION

2.1 Preliminary

*Large Language Models (*LLMs*)* Representative LLMs, *e.g.,* GPT-3 [25] and LLaMa [112], are pre-trained on enormous corpora, and have been shown incredible performance on various generative tasks in few-shot or zero-shot scenarios. LLMs are well known for their emergent abilities (*i.e.,* the sudden appearance of unseen behavior) [33], with no or few labeled data as demonstration on unseen tasks. Moreover, open-source LLMs, *e.g.,*LLaMa [112], Mistral [60] can be fine-tuned locally with more tasks, to improve their specialized abilities, while close-source LLMs, *e.g.,* OpenAI's GPT series (in particular GPT-3, 3.5 and 4) can only be queried online with APIs.

However, LLMs may suffer from the hallucination problem when demonstration is beyond the knowledge/scope of LLMs, leading to factual errors or unrelated answers [35]. To alleviate this, strategies below are typically adopted to constrain the responses of LLMs:

(1) Instruction: a combo of prompts and options (*i.e.,* candidate outputs/answers) for guiding LLMs to accomplish a given task.
(2) In-Context Learning (ICL): a method of prompt engineering that provides LLMs with demonstrations in the instruction [26].
(3) Retrieval Augmented Generation (RAG): a method to improve the quality of responses by feeding LLMs with relevant context retrieved, without updating the parameters of LLMs [29].

*Mixture of Experts (*MoE*)* The Mixture of Experts (MoE) architecture [58] is the basis of many state-of-the-art deep learning models. For example, MoE-based layers are being used to perform efficient computation in high-capacity neural networks and to improve parameter sharing in multi-task learning (MTL) [67, 78].

The original MoE model can be formulated as $y(x) = \sum_{i=1}^{n} g(x)_i e_i(x)$, where $\mathbf{E} = \{e_1, \cdots, e_n\}$ represents n expert networks, and g represents a gate network that ensembles the results from all experts. Specifically, g produces a distribution over n experts based on input x, and the final output is a weighted sum of the outputs of all experts. When truncated to top-k experts, each input only needs to activate k experts in inference without much information loss.

While MoE was first developed as an ensemble method of multiple individual models, recent works, *e.g.,* Switch Transformer [41], Mixtral [60], successfully turn it into basic building blocks (*a.k.a.* a router layer, MoE layer) and stack them into transformer layer. These router layers allocate input examples to different experts in $\mathbf{E}$, and are jointly trained with these experts. During inference, only the parameters of top-k experts are activated for each example (*e.g.,* top-2 for Mixtral). However, such design requires to train experts all in once, lacking the flexibility for fine-tuning a single expert. It is also hard to guarantee the experts are specialized in different domains, as observed in The Pile dataset [43] for Mixtral [60].

In light of this, we focus on external router networks as [18].

*Multi-task Learning(*MTL*)* Multi-task learning (MTL) solves multiple tasks at the same time, by exploiting commonalities and differences across tasks. In MTL, deep learning-based architectures that perform soft (*i.e.,* partial) parameter sharing have been proven to be effective [94, 99]. Inspired by this, we can cast DP tasks into a MTL problem, and solve such problem by multi-gate MoE [78].

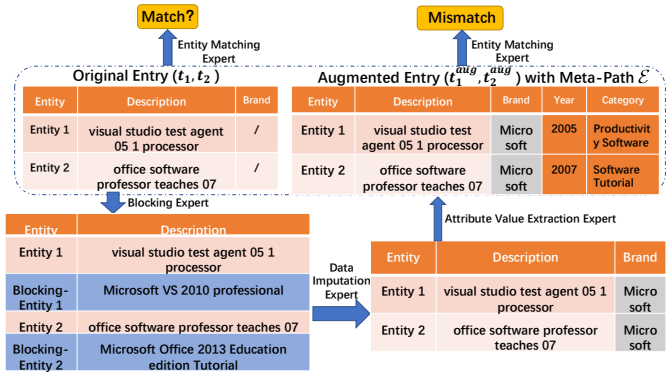


Figure 1: A toy example of multiple experts for enhanced EM **(entity matching). A meta-path "**BLK **(blocking)** → DI **(data imputation)** → AVE **(attribute value extraction)** → EM **" is found to help the** EM **expert to make the correct prediction.**

2.2 Problem Definition

In data management and data mining, DP is a critical step to deal with noises, missing values, inconsistencies and moreover, capture relations and associations between entries. Major DP procedures include data cleaning, data integration, data transformation and data reduction [49]. In this work, we mainly focus on tabular data, including both relational tables and web tables. Frequently used notations are summarized in Table 4 in Appendix.

Inspired by the success of *instruction-tuning* paradigm from the NLP literature[17], we adopt a universal DP task definition.

Assume that we are given a set $\mathcal{T}$ of DP tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$. Each $\mathcal{T}_i$ is provided with a set of training *queries* and associated *labels*, denoted as $\mathcal{X}_i = \{q_1, q_2, \ldots\}$ and $\mathcal{Y}_i = \{l_1, l_2, \ldots\}$, respectively.

Definition 2.1: (Data Preprocessing Query): A DP query for task $\mathcal{T}_i$ on table T is defined as a quadruple $q = (Ins^{\mathcal{T}_i}, D^{\mathcal{T}_i}, t, C^{\mathcal{T}_i})$, where (a) $Ins^{\mathcal{T}_i}$ is the natural-language instruction that specifies the task $\mathcal{T}_i$ (*e.g.*, entity matching, EM), (b) $D^{\mathcal{T}_i}$ is a set of $\mathcal{T}_i$-related demonstrations (*e.g.*, labeled examples of EM), (c) $t \in T$ is a *tuple* (*a.k.a. entry*) from table T, on which $\mathcal{T}_i$ is performed and (d) $C^{\mathcal{T}_i}$ is the expected output domain by performing the task following the instructions on the tuple t (*e.g.*, {match, mismatch} for EM). □

Given a training query q for task $\mathcal{T}_i$, its associated label l gives the ground truth from the expected output domain $C^{\mathcal{T}_i}$. To conduct a task $\mathcal{T}_i$, one should query an expert with q. To illustrate, we give a few representative tasks below, and a complete list with illustrating examples can be accessed in Appendix A.3:

Entity Matching (EM). Given a pair of tuples t_1, t_2 in T, EM is to infer whether they refer to the same real-world entity.

Error Detection (ED). Given a tuple t and an attribute a_i, ED is to detect whether there is an error in the a_i-attribute value of tuple t.

Data Imputation (DI). Given a tuple t and an attribute a_i such that the a_i-attribute value of t is missing, DI is to infer its correct value.

Column Type Annotation (CTA) Given a table T, CTA is to infer the type of each column h of T from a set of predefined semantic types.

Definition 2.2: (Expert): An expert e_i trained on DP task $\mathcal{T}_i$, is defined as a fine-tuned language model, which takes the query q as input, and return the task-specific output from the output domain. □

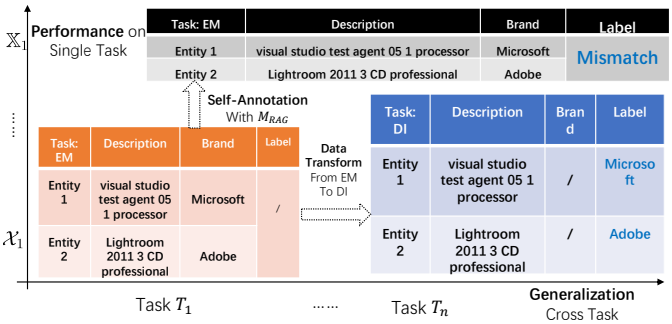


Figure 2: Illustration of our data augmentation method.

Note that each single expert e_i can response to queries of different tasks, since the experts in $\mathbf{E} = \{e_1, \cdots, e_n\}$ share the same architecture and most parameters with each other.

Definition 2.3: (Few-shot Learning): Each task $\mathcal{T}_i \in \mathcal{T}$ is provided with few-shot training queries and labels $\{\mathcal{X}_i, \mathcal{Y}_i\}$, and the remaining unlabeled queries are denoted as $\widetilde{\mathcal{X}}_i$. The training set $\mathbb{X}_i$ for $\mathcal{T}_i$ contains both labeled and unlabeled queries, *i.e.*, $\mathbb{X}_i = \mathcal{X}_i \cup \widetilde{\mathcal{X}}_i$. The overall training set $\mathbb{X} = \cup_{i=1}^{n} \mathbb{X}_i$ is the training set cross all tasks. □

For task $\mathcal{T}_i$ with training queries and labels $(\mathcal{X}_i, \mathcal{Y}_i)$, we denote $Eval(e_i, \mathcal{X}_i)$ as the performance evaluation, between $\mathcal{Y}_i$ and the output of expert e_i over $\mathcal{X}_i$. For binary classification DP tasks, the evaluation metrics is F-measure, for the other tasks is accuracy.

Note that given a training query $q \in \mathbb{X}_i$ for task $\mathcal{T}_i$, *e.g.*, EM, it may be possible to transform q (and its associated label) to a new query-label pair (q', l') for another task $\mathcal{T}_j$, *e.g.*, DI, via self-supervised learning, or masking strategies. Here the label l' can be a masked attribute from the original query q, or self-annotated, depending on tasks. The horizontal axis of Figure 2 give a toy example that transforms a query for EM to a new query-label pair for DI.

Definition 2.4: (Low-resource DP): DP tasks are solved by LLMs in a single small-memory GPU with few-shot labeled data. □

In practice, we refer to a single small-memory GPU as a consumer-level one with memory not exceeding 24GB, and few-shot labeled data as comprising up to 10% of the original labeled benchmark dataset.

Problem. The problem studied in this paper is stated as follows.

- *Input:* A set of tasks $\{\mathcal{T}_1, \cdots, \mathcal{T}_n\}$ with few-shot training data $\mathbb{X}$ in the low-resource DP setting.
- *Output:* An universal LLM-based system under the MoE architecture that is able to answer the (unseen) query of all $\mathcal{T}_i$.

3 THEORETICAL ANALYSIS

Despite the empirical success of the MoE architecture in MTL, the theoretical understanding of such architecture is still elusive. It is unclear why the experts can be specialized to make predictions for different inputs, and why the router can automatically learn to dispatch data. To this end, we provide some theoretical analysis in this section, answering the following questions:

- Q1: Can various DP tasks (*e.g.*, EM and DI) over different domains

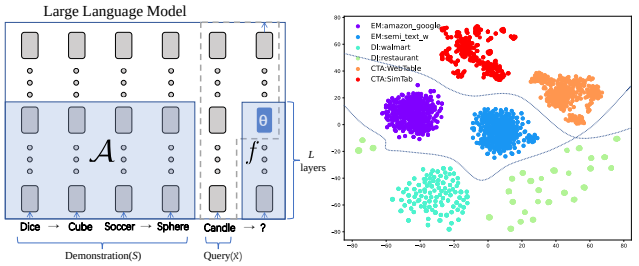


Figure 3: Illustration of Intrinsic Task Subspace (ITS) over task vectors. The left part[52] shows how LLM responses a task-specific query q with demonstrations. The right part is a 2d t-SNE plot of task vectors for different DP tasks over ITS. Dotted lines indicate decision boundary over different tasks.

(*e.g.,* scholar and e-commerce) be represented and learned over a compact low-dimension space, *i.e.,* a *intrinsic task subspace*?

- Q2: Why cannot a single expert fit well for multiple domains?
- Q3: How the router learn to dispatch data to the right experts?

To answer these questions, we provide the following three theorems. For the lack of space, the proofs are provided in Appendix A.5.

Theorem 1: ***(Intrinsic Task Subspace)*** *With unified representation of different tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$ and in-context learning (ICL) demonstrations D_i (i.e., $D^{\mathcal{T}_i}$), fine-tuning a LLM on task $\mathcal{T}_i$ is equivalent to learn a* ***task vector*** $\boldsymbol{\theta}_i(D_i)$*, and such vector is embed in a low-dimensional and compact intrinsic task subspace (ITS).* □

Theorem 1 indicates that with unified representation and proper ICL for each task $\mathcal{T}_i$, we can represent and learn multiple DP tasks $\mathcal{T}$ in a small ITS, denoted by **V**. In other words, fine-tuning a small set of parameters in a LLM can generalize it to multiple tasks. Figure 3 gives a intuitive visualization of ITS over task vectors.

Theorem 2: ***(Error Bound for Single and Mixture of Experts)*** *Consider fine-tuning a single expert h_N from the base LLM model h_0, to apply MTL over all DP tasks. Let $C \sim \cup_{i=1}^{|\mathcal{T}|} \mathcal{X}_i$ be the sampled distribution over all tasks $\mathcal{T}$ with N samples, $\mathbb{C} \sim \cup_{i=1}^{|\mathcal{T}|} \mathbb{X}_i$ be the actual distribution over $\mathcal{T}$, S be the source domain distribution from h_0, $\epsilon_{\mathbb{C}}(h_N)$ be the expected error bound of the single fine-tuned expert, and $\epsilon_C(h_N)$ be the empirical error. The expected error $\epsilon_{\mathbb{C}}(h_N)$ for single fine-tuned expert is upper bounded,* i.e.,

$$\epsilon_{\mathbb{C}}(h_N) \leq \epsilon_C(h_N) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}} + 2D(S,C)$$

where $D(S,C)$ is a distance function representing the gap between the source domain S and the target domain C, and δ is a constant.

Let $\mathcal{R}_N(H)$ be the Rademacher complexity of the hypothesis space H associated with expert models, $d_{\mathcal{N}}$ be the Natarajan dimension of the gating network $\mathcal{N}$ within its hypothesis space $\mathcal{B}$, $n = |\mathbf{E}|$ and k is the number of experts selected per query. For mixture of experts, the error bound is:

$$O(4C\mathcal{R}_N(H) + 2\sqrt{\frac{2kd_{\mathcal{N}}(1 + ln(\frac{n}{k}) + d_{\mathcal{N}}ln(2N) + ln(4/\delta)}{2N}})$$

which holds with a probability of at least $1 - \delta$. □

Theorem 2 shows that in few-shot learning, single expert cannot fit well for multiple target domains if (a) the model capacity is small, *i.e.,* $KL(h_N||h_0)$, which is negatively correlated with the model capacity [20], correspondingly large, (b) the sample number N in the target domain is small, and (c) the empirical error $\epsilon_C(h_N)$ is high, *i.e.,* there is a large bias between the sampled example distribution C and the actual distribution $\mathbb{C}$. And the error bound of the mixture of experts is directly proportional to the sparse factor $s = O(\sqrt{\frac{k}{N}(1 + \log(\frac{n}{k}))}$. This implies that, with a constant total number of experts n selecting fewer experts k leads to a more sparse network architecture, which consequently reduces the bound on the generalization error. Moreover, an increase in the number of training samples N can also minimize the error bound.

These conclusions have been validated through the experimental results and hyperparameter analysis in Section 5.2.

Theorem 3: ***(Router learns Clusters in ITS)*** *Given $N = \Gamma(dk \log k)$ samples drawn from a mixture of k spherical Gaussian in d-dimensions which are c-separated for some constant c, and an instantiation of the MoE architecture with $O(k \log k)$ experts, if we initialize the router weights g_i randomly, the router will learn to route examples according to the ITS task cluster distribution.* □

Theorem 3 guarantees the converge of the router network, the core of MoE architecture. However, such performance relies on a proper set of demonstration D_i, a suitable division of experts **E**, and a stratified sampling strategy for training the router network.

4 MIXTURE OF EXPERTS ARCHITECTURE BASED ON LARGE LANGUAGE MODELS

The overall architecture of MELD is presented in Figure 4, and it consists of the following four components.

- **The enhanced RAG component**. It takes few-shot labeled data as input, and enlarge/enrich labeled data in $\mathcal{X}_i$ as output $\mathbb{X}_i$ in a self-supervised manner for task $\mathcal{T}_i$. A fine-tuned sentence-bert model is used as the backbone of RAG system; such design can effectively encode data entries from different domains to a unified representation space. It also retrieves relevant demonstrations D for each data entry and initializes a set **E** of experts.
- **The meta-path search component**. It takes the enlarged training data $\mathbb{X}_i$ and the expert set **E** as input, and finds a meta-path $\mathcal{E}_i$ (*i.e.,* a sequence of experts in **E**) for task $\mathcal{T}_i$, to augment $\mathbb{X}_i$ to $\mathbb{X}_i^{aug}$, by revising and adding attributes for each query $q \in \mathbb{X}_i$.
- **The expert refinement**. It takes augmented training data $\mathbb{X}_i^{aug}$ and the expert set **E** as input, and fine-tunes the experts **E** to $\mathbf{E}^{aug}$, guided by the information bottleneck theory.
- **The router network** $\mathcal{N}$. It takes the (fixed) refined expert set $\mathbf{E}^{aug}$ as input, and designs with a sparse multi-gate network, to select top-k experts for answering a query $q \in \mathbb{X}$.

Below we elaborate each component one by one.

4.1 Enhanced RAG for Cross-domain Retrieval

Retrieval-Augmented Generation(RAG) is a method to retrieve relevant contextual data entries or chunks from a large corpus (*e.g.,* knowledge graph, book) and provide to the model as reference, to improve the quality of LLM responses. However, data entries from multiple DP tasks may have different structures, which are hard

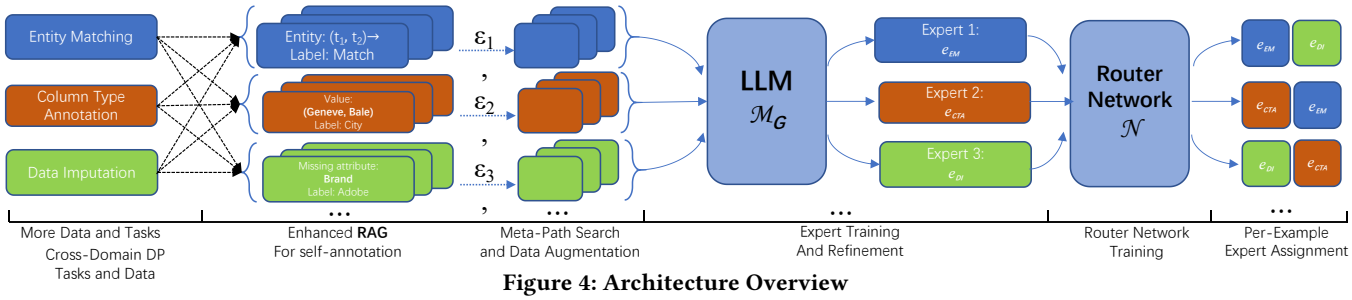


Figure 4: Architecture Overview

to compare and retrieve. In this section, we propose a simple yet effective method to serialize and align data from different domains.

Entry Alignment. For structure and semi-structured data, the structure similarity holds equal importance as semantic similarity, *e.g.,* if t_1 and t_2 share the same *brand* and *category* attribute, we can align t_1 and t_2 as similar entities. For tasks across tables, *e.g.,* CTA, columns with same semantic type or knowledge graph relations should also be aligned; for binary classification tasks, *e.g.,* EM, if t_1 and t_2 are labeled as *match*, they should be grouped as similar entries.

Based on this, for each q in $\mathbb{X}$, we search a positive set $\mathcal{P}_q$ (resp. a negative set $\mathcal{N}_q$) containing all aligned (resp. unaligned) entries.

Fine-tuning RAG *Model.* Given $(q, \mathcal{P}_q, \mathcal{N}_q)$ as training data, we tokenize and pass them to a sentence-bert [95] model $\mathcal{M}_{\text{RAG}}$, and fine-tune the model with the contrastive learning loss [28]:

$$\min \sum_{p \in \mathcal{P}_q} -\log \frac{\exp\left(\langle \boldsymbol{emb}_q, \boldsymbol{emb}_p \rangle / \tau\right)}{\sum_{p' \in \mathcal{P}_q \cup \mathcal{N}_q} \exp\left(\langle \boldsymbol{emb}_q, \boldsymbol{emb}_{p'} \rangle / \tau\right)},$$

where $\boldsymbol{emb}$ is an embedding and τ is the temperature parameter.

Moreover, we serialize each query q to a dict format, which also contains meta-data for q, *e.g.,* table title, column header; if $\mathcal{N}_q = \emptyset$, we conduct hard negative sample search with the initial model $\mathcal{M}_{\text{RAG}}$ over $\mathbb{X}$, to add negative examples for $\mathcal{N}_q$.

Self-Annotation. When the training of $\mathcal{M}_{\text{RAG}}$ is finished, we apply $\mathcal{M}_{\text{RAG}}$ to self-annotate unlabeled queries in $\widetilde{\mathcal{X}_i}$. *e.g.,* for EM, given an unlabeled query $q_i \in \widetilde{\mathcal{X}_{EM}}$, $\mathcal{M}_{\text{RAG}}$ can search the most similar q_j over entire $\mathbb{X}$ and self-annotate the entries in q_i and q_j as *match*. This procedure follows a self-supervised learning paradigm, and effectively enlarge the labeled data $\mathcal{X}_i$ to $\mathbb{X}_i$ by adding self-annotated data. Besides, we can also apply the transformation technique in Section 2.2 to further enlarge $\mathbb{X}_i$ with labeled queries from other tasks. Figure 2 gives an example of both ways for enlarging labeled data.

Expert Initialization. For each task $\mathcal{T}_i$, $\mathbb{X}_i$ is used to initialize the training of each expert e_i, by fine-tuning a LLM, denoted by $\mathcal{M}_G$.

4.2 Heuristic Meta-path Search

There are a host of data augmentation methods [30, 73] for DP. However, such methods are either statistical or they use pre-defined global operators for augmentation. Alternatively, we consider a fixed set of experts $\mathbf{E} = \{e_1, \cdots, e_n\}$, and find a meta-path (*i.e.,* a sequence of experts in $\mathbf{E}$) for task $\mathcal{T}_i$. Such meta-path can help to augment data in $\mathbb{X}_i$ reasonably. Below we define such expert-based meta-path and data augmentation over the meta-path.

Definition 4.1: (Meta-path over Experts): A meta-path $\mathcal{E}_i$ for task $\mathcal{T}_i$ is a sequence of experts $e_{j_1}, \cdots, e_{j_n}$ from the experts set $\mathbf{E}$; it describes the order of experts to be applied for task $\mathcal{T}_i$. □

Definition 4.2: (Data Augmentation Over Meta-path): Given $\mathbb{X}_i$ for task $\mathcal{T}_i$, we denote $\mathbb{X}_i^{j_1}$ as the augmented set of $\mathbb{X}_i$ by querying expert e_{j_1}. Similarly, $\mathbb{X}_i^{\mathcal{E}_i}$ is the augmented set of $\mathbb{X}_i$ by a meta-path $\mathcal{E}_i = \{e_{j_1}, \cdots, e_{j_n}\}$, *i.e.,* by querying the experts in $\mathcal{E}_i$ in order. □

We show an example in Figure 1, in which $\mathbb{X}_{EM}$ is augmented by the meta-path $\mathcal{E}_{EM} = \{e_{\text{blocking}}, e_{\text{DI}}, e_{\text{AVE}}, e_{\text{EM}}\}$ in order.

Heuristic Meta-path search. Given labeled data $\mathcal{X}_i$ for task $\mathcal{T}_i$, we want to find a sequence of experts $\mathcal{E}_i = \{e_{j_1}, \cdots, e_{j_n}\}$ such that the performance of the augmented data, *i.e., Eval*$(e_i, \mathcal{X}_i^{\mathcal{E}})$, is the best.

Here we apply a greedy search algorithm for finding a meta-path $\mathcal{E}_i$ for task $\mathcal{T}_i$, which reduces the search space by incorporating user-defined sub-optimal paths, *e.g.,* $\{e_{\text{Blocking}}, e_{\text{EM}}\}$ (resp. $\{e_{\text{EL}}, e_{\text{CTA}}\}$) is widely used in EM (resp. tabular interpretation learning [27]).

Theorem 4: *Given a fixed set of expert* $\mathbf{E}$*, querying a meta-path* $\mathcal{E}_i$ *for task* $\mathcal{T}_i$ *is Church-Rosser,* i.e., *it converges at the same result no matter in what order the experts are queried.* □

After finding a meta-path $\mathcal{E}_i$ for task $\mathcal{T}_i$, we can query $\mathcal{E}_i$ to augment training data $\mathbb{X}_i$ to $\mathbb{X}_i^{aug}$ with self-supervised annotation.

4.3 Expert Refinement

Note that for each initialized expert e_i for task $\mathcal{T}_i$, there is a high risk that e_i may overfit to the biased distribution of $\mathcal{X}_i$, since $\mathbb{X}_i^{aug}$ are augmented from $\mathcal{X}_i$ and may share a similar biased distribution. As discussed in [8], such distribution may lead to a higher empirical error ϵ_C (h_N) in Theorem 2. To alleviate such concern, we introduce the Min-Max optimization target guided by the information bottleneck theory, to improve the generalizability of each expert e_i.

Information Bottleneck. Information bottleneck [110, 111] was used to balance the complexity of representation and the power of predicting, based on the notion of minimal sufficient statistics for extracting information about target Y from input X into representation Z. It imposes regularization at representation Z by minimizing the mutual information between input X and the learned representation Z,*i.e.,* $\min I(X;Z)$, while maximizing the mutual information between target output Y and X, *i.e.,* $\max I(Y;Z)$ [62].

In expert training, the information bottleneck theory provides useful insights: consider training expert e_i with training data $(\mathbb{X}_i, \mathcal{Y}_i)$; it is equivalent to find the most relevant task vector $\boldsymbol{\theta}_i$ as representation. On the one hand, the distribution of $\mathbb{X}_i$ should be diverse,

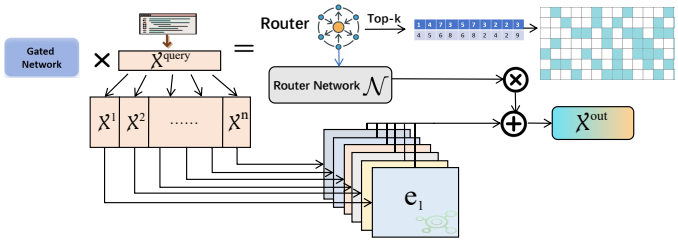


Figure 5: Model architecture of MELD

a.k.a. minimize $I(\mathbb{X}_i; \boldsymbol{\theta}_i)$. Otherwise e_i may overfit to a biased distribution of sampled training data $\mathbb{X}_i$, and cannot learn the high-level and intrinsic features. On the other hand, the distribution of $\mathbb{X}_i$ should fall in the same cluster with $\boldsymbol{\theta}_i$ in ITS, as shown in Theorem 3, *a.k.a.* maximize $I(\mathcal{Y}_i; \boldsymbol{\theta}_i)$. Otherwise e_i may suffer from underfitting issue with low performance, due to the lack of relevant information.

Training Process. We denote $\theta_{\mathcal{M}_{\text{RAG}}}$ as the parameters for fine-tuned RAG model in Section 4.1, and $\theta_{\mathcal{M}_G}$ as the parameter of the base LLM-model of each expert, and $\text{RAG}(\mathcal{X}_i))$ as the operations we use to augment $\mathcal{X}_i$, including both self-annotation and meta-path augmentation. The optimization function of training LLM-based e_i is:

$$\arg \min_{\theta_{\mathcal{M}_{\text{RAG}}}} \max_{\theta_{\mathcal{M}_G}} I(\mathcal{M}_G(\mathcal{X}_i); \mathcal{M}_G(\text{RAG}(\mathcal{X}_i))) \quad (1)$$

Intuitively, (a) $\max \theta_{\mathcal{M}_G}$ is explicitly conducted, by parameter-efficient fine-tuning $\mathcal{M}_G$ and maximizing the mutual information between the output of $\mathcal{M}_G$ and label $\mathcal{Y}_i$; and (b) $\min \theta_{\mathcal{M}_{\text{RAG}}}$ is implicitly enforced, by controlling the sample parameter for $\mathcal{M}_{\text{RAG}}$ and meta-path $\mathcal{E}_i$ and adding $\Delta\mathcal{X}_i = \text{RAG}(\mathcal{X}_i)$ as supplement training data for $\mathcal{M}_G$, while minimizing the mutual information between the labeled training data $\mathcal{X}_i$ and external training data $\Delta\mathcal{X}_i$.

In practice, we adopt a iterative optimization strategy to fulfill the target function. Specifically, $\mathcal{M}_G$ is initialized with expert e_i (Section 4.1). Then we iteratively control $\text{RAG}(\mathcal{X}_i)$ to add diverse training data $\Delta\mathcal{X}_i$ by extracting cross-domain examples and demonstrations, as well as implementing data augmentation with meta-path $\mathcal{E}_i$. After adding $\Delta\mathcal{X}_i$ to $\mathcal{X}_i$, we further fine-tune $\mathcal{M}_G$ with new data until convergence. Such iterations continue σ times.

After refinement, e_i is refined to e_i^{aug}, which is more robust to various DP tasks and cross-domain queries, while retaining high performance on its own $\mathcal{T}_i$. Denote the set of refined experts by $\mathbf{E}^{aug}$. We apply low-rank adaptation [54] (*a.k.a.* LoRA) to fine-tune $\mathcal{M}_G$ for training and refining each expert $e_i \in \mathbf{E}^{aug}$.

4.4 Router Network

In this section, we train a light-weighted sparse-gated router network $\mathcal{N}$ to select top-k experts in $\mathbf{E}^{aug}$ for each input query.

The information bottleneck theory also provides insight in optimizing $\mathcal{N}$. Given query q_i, on the one hand, the selected top-k experts should be diverse to provide different yet valuable views of q_i; this is equivalent to minimize the mutual information between the selected experts. On the other hand, the selected experts should be relevant to q_i; this is equivalent to maximize the mutual information between the output of selected experts and corresponding labels.

Router Network. Given a labeled query $q_u \in \mathbb{X}_u^{aug}$, let $\mathcal{N}(q_u)$ be the top-k experts selected by the sparse gated network $\mathcal{N}$ for q_u and task $\mathcal{T}_u$, and (q_u^i, l_u^i) be the transformed query-label pair from task $\mathcal{T}_u$ to $\mathcal{T}_i$ with self-annotation. The optimization function is:

$$\max \sum_{e_i \in \mathcal{N}(q_u)} I(e_i(q_u^i); l_u^i); \min \sum_{e_i, e_j \in \mathcal{N}(q_u)}^{i \neq j} I(e_i(q_u^i); e_j(q_u^j)) \quad (2)$$

In practice, Eq.2 can be approximated with contrastive training loss [90, 105]. Therefore, we apply a transformer network that shares the encoding layers with $\mathcal{M}_{\text{RAG}}$, for $\mathcal{N}$ and further fine-tune it with contrastive loss. The positive and negative examples are extracted from labeled data across all tasks. Figure 5 gives an illustration of $\mathcal{N}$.

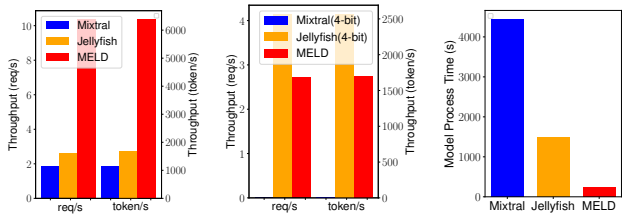
Table 1: Overall Performance

Task	Dataset	MELD Few-shot	Non-LLM Baseline Few-shot	LLM Baseline Few-Shot	Mixtral Few-shot
EM & (BLK)	Amazon-Google	**83.41(74.12)**	61.88(50.47)	65.98(/)	51.28(/)
	Walmart-Amazon	**91.42(78.80)**	79.09(58.21)	42.03(/)	39.78(/)
	WDC-All	**91.97(31.50)**	34.35(1.70)	49.80(/)	48.97(/)
	Ant-Buy	**91.12(86.20)**	84.89(40.66)	71.40(/)	60.42(/)
	Semi-Text-Watch	**78.28(59.23)**	23.60(2.66)	54.27(/)	40.55(/)
	Semi-Text-Computer	**86.46(30.85)**	33.90(8.09)	76.80(/)	73.15(/)
DC	Hospital	**95.01**	67.10	49.30	53.20
	Rayyan	**82.15**	28.50	9.39	6.68
	Beer	**97.30**	90.31	51.30	56.27
ED	Hospital	**98.51**	95.23	89.41	69.14
	Rayyan	**90.37**	80.21	69.67	31.96
	Beer	99.10	**100.00**	81.64	70.23
CTA	SemTab19	**89.35**	69.70	87.77	**89.35**
	WebTables	**96.30**	90.93	94.77	80.16
RE	WikiGS-RE	**89.30**	73.50	60.38	65.88
EL	WikiGS-EL	**87.05**	60.55	82.20	73.25
SM	CMS	**60.27**	50.00	59.29	31.01
	Synthea	**56.00**	38.50	40.00	23.53
DI	Walmart	**87.50**	65.70	57.69	79.82
	Amazon	**75.12**	60.35	60.05	62.62
	Restaurant	**93.10**	37.50	68.97	72.41
AVE	OA-mine	74.62	67.00	65.70	**77.36**

5 EXPERIMENTAL STUDY

Our experiments focus on answering the following questions:

- How does MELD perform compare with other non-LLM methods and local-LLM methods, especially in few-shot scenarios?
- How does MELD benefit from the MoE architecture design, especially in cross-dataset and cross-task scenarios?
- The effectiveness and efficiency comparison between the light-weighted standalone router network architecture, *e.g.*,MELD, and the built-in MoE layer based model, *e.g.*, Mixtral 8×7B?
- How does the number of experts, as well as the meta-path selection, affect the overall performance of MELD?



(a) 4 × 3090 (b) 1 × 3090 (c) Model process time

Figure 6: Efficiency among different LLMs-based models (4-bit quantization for Jellyfish and Mixtral **on** 1 × 3090**)**

5.1 Setup

Statistics. As shown in Table 5 in Appendix, we selected 19 datasets over 10 typical DP tasks to show the performance of MELD. In all tasks except schema matching, we use few-shot labeled data (usually ≤ 10%), as shown in column #Instance (few-shot).

Methods. We categorized the baselines as follows. (1) Non-LLM methods [87]. (a) ED: Raha[80], (b) DI: IPM[83], (c) Blocking: DeepBlocker[108], (d) EM: Ditto[73] and PromptEM[114], (e) DC: Baran[79] and Garf[93], (f) CTA: RECA[32], (g) RE/EL: TURL[27], (h) SM: CONSchema[118] and SMAT[129], and (k) AVE: MAVE[122]. Other methods, *e.g.,* HoloClean [96], DODUO [31] have been shown to be outperformed by the listed competitors [32, 80], and hence not compared. (2) LLM-based methods. JellyFish[127] uses a 13B LLM model (1.8× than MELD) to solve multiple DP tasks. For table interpretation tasks (*e.g.,* CTA, RE, EL), we compared TableLLaMa[132] which applies a 7B foundation model. For AVE task, we used ExtractGPT[5], compared to its local LLM model with up to 70B parameters (10× than MELD). (3) MoE models. We compared the state-of-the-art MoE foundation model Mixtral-8×7B[60] (*i.e.,* Mixtral), which embeds the MoE layer $\mathcal{N}$ in model parameters, and jointly train $\mathcal{N}$ with a set $\mathbf{E}$ of 8 experts, each of which is a 7B LLM.

Default Parameters. For Blocking and ED, we only applied our RAG model $\mathcal{M}_{\text{RAG}}$ due to the large search space. For other tasks, we also uses the LLM-based MoE system. The default number of k is set to 3, the number of iterations σ for expert refinement in Section 4.3 is set to 3, the demonstration number $|D_i|$ is set to 8. τ in RAG is 0.02. The detailed implementation is listed in Appendix A.4.

Metrics To evaluate DP tasks, we measured accuracy for DI, AVE; top-1 accuracy for EL; top-1 recall for blocking; F1 score for EM, ED, DC, SM, and micro-F1 score for CTA, RE tasks in a 100-scale.

Environment We select bge-large-en[34] as the backbone for the RAG models $\mathcal{M}_{\text{RAG}}$, and Mistral-7B[59] as the backbone of expert model by default. We conducted the experiments on a single machine powered by 256GB RAM and 32 processors with Intel(R) Xeon(R) Gold 5320 CPU @2.20GHz and 4 Nvidia GeForce RTX 3090 GPUs. We use one GPU for MELD and all for other baselines. Each experiment was run 3 times and the average is reported.

5.2 Effectiveness Evaluation

We compared the performance of MELD with various non-LLM and LLM baselines in Table 1. In few-shot scenarios, MELD consistently outperforms all non-LLM baselines, which means that MELD has better data utilization. In particular, 10%-20% labeled training data suffices to train a robust expert e_i for task $\mathcal{T}_i$, while the shared parameter from other experts can prevent e_i from being overfitting.

Compared to LLM baselines, which are trained over MTL paradigm, MELD beats them with significant fewer parameters. This indicates that the MoE architecture is good at handling MTL, and multiple sparse experts can outperform one dense one. Besides, we argue that several LLM baselines, including Jellyfish and TableLLaMa, require high-cost pre-training over enormous task-specific corpus with thousands of GPU hours (*e.g.,* millions of Wikipedia webtables[132]), while MELD only needs low-cost fine-tuning for training each expert from a base model with less than 20 GPU hours.

Compared to Mixtral, which also applies a build-in MoE layer, we can see that Mixtral outperforms MELD in a few tasks (*i.e.,* AVE, CTA). However, Mixtral fails to apply a good routing strategy, and Mixtral does not balance the load well for the task family $\mathcal{T}$ to its 8 experts, leading to its better performance in open-domain/complex tasks with long context and information retrieval, *e.g.,* DI, AVE, and poor performance in close-domain/simple tasks, *e.g.,* EM, DC.

5.3 Efficiency Evaluation

We compared the efficiency of MELD, Jellyfish and Mixtral in Figure 6, comparing the inference throughput speed and model process time. This comparison is conducted on two settings: 4×3090 GPUs and 1×3090 GPU with vLLM [66]. Due to the VRAM requirement of Mixtral, we only report its performance on the former.

Firstly, we report the throughput over 4 GPUs with vLLM [66]. Due to the small size of experts in MELD, a single 3090 GPU can hold a maximum of 16 experts for MELD, while the load-balance system of MELD and vLLM can gather similar queries within the same GPU. Therefore, MELD achieves data parallelism over 4 GPUs, and gain non-trivial 3.7× throughput improvement with 13B Jellyfish and 5.6× with 56B Mixtral, which have to apply tensor parallelism, and suffer from the communication overhead over multiple GPUs.

Secondly, we report the throughput over a single GPU, a prevalent consumer scenario. MELD perform well with full precision model, while Jellyfish has to apply a 4-bit quantization [42] to make inference on a single GPU, and Mixtral cannot deploy on a single GPU even with 4-bit quantization, due to OOM issues. Although MELD is around 1.3× slower than 4-bit Jellyfish, the quantization is time-consuming and it leads to a significant performance drop.

We also report the model process time of each methods, *i.e.,* the time of merging trained LoRAs into the base model and preparing it for inference with vLLM. MELD applies a dynamic LoRA switch technique, which avoids merging multiple LoRA into a single model, and only needs to load and concatenate on multiple LoRAs, reducing the i/o cost. While Jellyfish and Mixtral have to apply a time-consuming merging and quantization operation. As a result, MELD is 10× and 30× faster than Jellyfish and Mixtral in model process.

5.4 Cross-Dataset and Cross-Task Comparison

We evaluated the cross-dataset (*i.e.,* C-D) and cross-task (*i.e.,* C-T) performance of MELD, where C-D means we mask expert e_i and training data $\mathcal{X}_i$ for task $\mathcal{T}_i$, while C-T means we mask all experts and training data that are same as $\mathcal{T}_i$ (*e.g.,* mask all EM experts for evaluation on the Amazon-Google dataset). The result is presented in Table 2. To prevent the domain overlap, we select 6 datasets with different domains, and limit the overall experts of MELD into 6.

Table 2: Cross-Dataset(C-D) and Cross-Task(C-T)

Task	Dataset	MELD C-D	MELD C-T	LLM Baseline C-D	LLM Baseline C-T	Mixtral C-D	Mixtral C-T
EM	Amazon-Google	**69.05**	67.95	18.58	18.58	43.23	43.23
	Semi-Text-Watch	**65.07**	51.13	20.52	20.51	37.12	37.12
CTA	SemTab19	**76.84**	61.21	15.79	7.96	64.83	61.64
	WebTables	86.76	**88.95**	38.92	14.29	79.72	67.64
DI	Walmart	54.80	54.80	43.26	17.86	**79.82**	**78.85**
	Restaurant	**75.86**	**75.86**	68.96	6.95	72.43	58.62

Table 3: Performance for Ablation Study

Task	Dataset	MELD w/o MoE	MELD w/o RAG	MELD w/o Meta-Path	MELD with Mixtral
EM	Amazon-Google	76.70	69.21	62.52	77.85
	Walmart-Amazon	87.66	81.44	79.55	91.03
	WDC-All	90.38	83.16	91.73	91.32
	Ant-Buy	87.58	85.75	90.12	85.26
	Semi-Text-Watch	70.78	55.07	39.89	75.42
	Semi-Text-Computer	79.49	42.02	63.74	81.98

Compared with LLM baselines, MELD suffers less performance drop in C-D and C-T scenarios, which is contributed by the information bottleneck guided expert training, as well as the RAG system across datasets and tasks. Nonetheless, Mixtral also performs well in open-domain tasks, which means the MoE architecture is suitable in MTL. Besides, the shared parameters of experts in MELD and Mixtral effectively prevent them from being overfitting to few-shot data and specific task, or suffering hallucination problems.

5.5 Ablation Study

We selected EM for ablation study, varying the following in Table 3:

- MELD w/o MoE, a single expert fine-tuned per task;
- MELD w/o RAG, where each expert is fine-tuned without cross-domain data augmentation and RAG; and
- MELD-w/o Meta-Path, where each expert is fine-tuned without meta-path based data augmentation

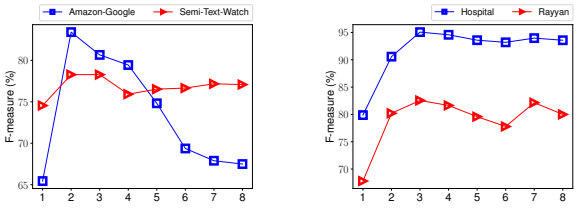
With only a domain-specific expert e_i, MELD w/o MoE, is not the good solution, since different experts can provide additional information to boost the performance. MELD w/o RAG suffers from performance drop over all scenarios, justifying the effectiveness of $\mathcal{M}_{\text{RAG}}$. For semi-structured or low-quality data, *e.g.,* semi-text-w and amazon-google, the meta-path can augment structural information and significantly improve the performance. As remarked earlier, if we replace the router network $\mathcal{N}$ with Mixtral, it also suffers performance drop, due to unbalance load between experts in Mixtral.

5.6 Hyper-parameter Analysis

We tested the impact of k and the distribution of task vectors Θ and attention weights across experts in $\mathbf{E}$ and tasks in $\mathcal{T}$. Following [52], we present the t-SNE plot figure in Figure 3, to visualize the embeddings generated by $\mathbf{E}$ and router network $\mathcal{N}$. This proves that $\mathcal{N}$ dispatch queries based on the latent distribution of Θ.

Figure 7 provides the sensitive analysis of number k. Initially, the performance rises with the number of experts. However, when $k \geq 4$, the overall performance shows a slight drop, while the parameter size still increases. This justifies that involving more experts are not always good, since their inherit parameters may conflict.

We provide the attention weights across experts in Appendix Figure 8. The utilization rate of experts diverged significantly across tasks and datasets. We also provide the comparison of MELD and online GPT-4 in Appendix Table 6, following the best performance in [5, 127]. Similar to Mixtral, GPT-4 shows better ability in complex/open-domain tasks.



(a) EM (b) DC

Figure 7: Performance for different number of experts k

6 RELATED WORKS

6.1 Data Processing Solutions

We briefly review the related works on DP solutions below.

Non-LLM *solutions.* For ED and DC, traditional methods mainly depends on hand-crafted rules [11, 14, 19, 39, 40, 45, 46], pattern discovery [12], statistical modeling [16, 56, 68]. While recent works apply ML model, they focus on few-shot learning with a series of ML pipelines [51, 76, 79, 80, 96, 121] or pre-trained language models (PLMs) [30, 93]. For entity resolution (*i.e.,* Blocking, EM), traditional solutions mostly consider attribute equivalence, hashes or similarities [4, 38, 47, 48, 91, 115], while recently ML methods for blocking has also been invested [108], following the application of ML and PLMs for entity matching [30, 36, 70, 72, 73, 84, 114, 134]. For tabular understanding [130] (*i.e.,* SM, CTA, RE, EL), most of recent works focus on table representation learning [27, 31, 32, 57, 129], usually cooperated with fine-tuned PLMs. For data extraction (*i.e.,* DI, AVE), while traditional rule-based solutions [37, 96, 103] remain one of the prevalent approaches, a variety of ML models are applied, including LSTM-CRF [133], GAN [104, 125], autoregressive models [53], OT [86], autoencoders [88, 124], transformer-based ML methods [2, 37, 109, 119, 120] and PLMs [83].

Instead of focusing on one or a few similar DP tasks as above, we aim to design a universal DP task solver across all domains.

LLM *solutions.* Recently, a host of pioneering works focus on transforming DP tasks into generation tasks, leveraging online or local LLMs. Online models, *e.g.,* ChatGPT, GPT-3, [64, 71, 87, 92, 128], typically employ various prompt engineering methods on frozen LLMs or fine-tune ChatGPT for a variety of table-related tasks. However, such implementation on online model is unstable and costly. Worse still, data privacy cannot be guaranteed. There are also works on fine-tuning and deploying local LLMs [5, 127, 132] on various DP tasks, which however, aim to develop one base model for various DP tasks. They cannot perform well in MTL, and require to pre-train LLMs which is costly, while our method only requires low-cost fine-tuning, and incorporate MoE for high-performance MTL.

6.2 Mixture of Experts

MoE has been investigated in natural language processing [13, 22, 41, 63, 67, 77, 85, 136–138] and it has been proven to be an effective method of increasing the model's capacity in parameter size, where

certain modules of the model are activated, while computation is kept the same or close to its dense counterparts. Their is a host of work focusing on improving the routing strategy of MoE [50, 69, 97, 136], to sparsely select a single or k experts [10, 18, 138]. MoE has also be well invested in multi-task learning (MTL) [21, 75, 78], including multilingual machine translation [50, 65], natural language generation [117, 126] and recommendation system [131].

Unlike these studies, we apply MoE by scaling both the volume of data, and the number/types of DP tasks, aiming to mitigate the instability issue inherent in the training the MoE architecture.

Recently, a variety of work focuses on applying a unified MoE base model to MTL, *e.g.,* Mixtral [60], DeepSeek-MoE [3] and switch transformer [41]. [100, 123] focus on how instruction fine-tuning with scaled tasks can counteract the generalization challenges tied to MoE models combined with small models. Differ from this, we scrutinize the efficacy of instruction fine-tuning of each expert, and present an extreme parameter efficiency with small experts at a large scale up to 7B parameter base model. We use a MoE-like structure to address the parametric knowledge retention issue in LLMs, rather than significantly expanding the model parameters.

6.3 Multi-LoRA Architecture

There are effective multi-LoRA architectures that could be used to handle DP tasks, *e.g.,* [21, 55, 75, 89]. More details are in Appendix A.1. Different from them, our work could adopt any LLMs as the backbone and any inference frameworks.

7 CONCLUSIONS

We proposed an efficient Mixture of Experts on Large Language Models for Data Preprocessing (MELD) that is a universal solver for the low-resource DP tasks. To adapt to low-resource environment, we develop several expert-tuning and MoE-tuning techniques, including the RAG system, meta-path search strategy, expert refinement and router network training. We also theoretically prove that MoE in MELD is superior than a single expert and the proposed router network is able to assign data to the right experts. Finally we conduct thorough experiments to show MELD outperforms state-of-the-art methods in aspects of efficiency and effectiveness, especially in the low-resource environment.

REFERENCES

[1] Suad A Alasadi and Wesam S Bhaya. 2017. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences* 12, 16 (2017), 4102–4107.

[2] Parikshit Bansal, Prathamesh Deshpande, and Sunita Sarawagi. 2021. Missing Value Imputation on Multidimensional Time Series. *PVLDB* 14, 11 (2021), 2533–2545.

[3] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. DeepSeek LLM: Scaling Open-Source Language Models with Longtermism. *arXiv preprint arXiv:2401.02954* (2024).

[4] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. 2006. Adaptive Blocking: Learning to Scale Up Record Linkage. In *ICDM*. 87–96.

[5] Alexander Brinkmann, Roee Shraga, and Christian Bizer. 2023. Product Attribute Value Extraction using Large Language Models. *arXiv preprint arXiv:2310.12537* (2023).

[6] Rich Caruana. 1997. Multitask learning. *Machine learning* 28 (1997), 41–75.

[7] Lequn Chen, Zihao Ye, Yongji Wu, Danyang Zhuo, Luis Ceze, and Arvind Krishnamurthy. 2023. Punica: Multi-tenant lora serving. *arXiv preprint arXiv:2310.18547* (2023).

[8] Shuxiao Chen, Edgar Dobriban, and Jane H Lee. 2020. A group-theoretic framework for data augmentation. *The Journal of Machine Learning Research* 21, 1 (2020), 9885–9955.

[9] Zui Chen, Lei Cao, Sam Madden, Tim Kraska, Zeyuan Shang, Ju Fan, Nan Tang, Zihui Gu, Chunwei Liu, and Michael Cafarella. 2023. SEED: Domain-Specific Data Curation With Large Language Models. *arXiv e-prints* (2023), arXiv–2310.

[10] Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. 2022. Towards understanding mixture of experts in deep learning. *arXiv preprint arXiv:2208.02813* (2022).

[11] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 458–469.

[12] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 1247–1261.

[13] Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. 2022. Unified scaling laws for routed language models. In *International Conference on Machine Learning*. PMLR, 4057–4086.

[14] Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. 2007. Improving Data Quality: Consistency and Accuracy. In *VLDB*. 315–326.

[15] Koby Crammer, Michael Kearns, and Jennifer Wortman. 2008. Learning from Multiple Sources. *Journal of Machine Learning Research* 9, 8 (2008).

[16] Sushovan De, Yuheng Hu, Venkata Vamsikrishna Meduri, Yi Chen, and Subbarao Kambhampati. 2015. BayesWipe: A Scalable Probabilistic Framework for Cleaning BigData. *CoRR* abs/1506.08908 (2015).

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[18] Nishanth Dikkala, Nikhil Ghosh, Raghu Meka, Rina Panigrahy, Nikhil Vyas, and Xin Wang. 2023. On the benefits of learning to route in mixture-of-experts models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 9376–9396.

[19] Xiaoou Ding, Hongzhi Wang, Jiaxuan Su, Muxian Wang, Jianzhong Li, and Hong Gao. 2020. Leveraging currency for repairing inconsistent and incomplete data. *TKDE* (2020).

[20] M.N. Do. 2003. Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models. *IEEE Signal Processing Letters* 10, 4 (2003), 115–118. https://doi.org/10.1109/LSP.2003.809034

[21] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, et al. 2023. LORAMOE: REVOLUTIONIZING MIXTURE OF EX-PERTS FOR MAINTAINING WORLD KNOWLEDGE IN LANGUAGE MODEL ALIGNMENT. *arXiv preprint arXiv:2312.09979* (2023).

[22] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*. PMLR, 5547–5569.

[23] Paul ErdHos and Alfréd Rényi. 1961. On a classical problem of probability theory. *Magyar Tud. Akad. Mat. Kutató Int. Közl* 6, 1 (1961), 215–220.

[24] Bhagavatula et al. 2015. Tabel: Entity linking in web tables. In *ISWC*.

[25] Brown et al. 2020. Language models are few-shot learners. *NIPS* (2020).

[26] Dong et al. 2022. A survey for in-context learning. *arXiv preprint* (2022).

[27] Deng et al. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* (2022).

[28] Karpukhin et al. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint* (2020).

[29] Lewis et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NIPS* (2020).

[30] Miao et al. 2021. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In *SIGMOD*.

[31] Suhara et al. 2022. Annotating columns with pre-trained language models. In *SIGMOD*.

[32] Sun et al. 2023. RECA: Related Tables Enhanced Column Semantic Type Annotation Framework. *VLDB* (2023).

[33] Wei et al. 2022. Emergent abilities of large language models. *arXiv preprint* (2022).

[34] Xiao et al. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding.

[35] Zhao et al. 2023. A survey of large language models. *arXiv preprint* (2023).

[36] Wenfei Fan, Wenzhi Fu, Ruochun Jin, Muyang Liu, Ping Lu, and Chao Tian. 2023. Making It Tractable to Catch Duplicates and Conflicts in Graphs. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–28.

[37] Wenfei Fan, Ziyan Han, Weilong Ren, Ding Wang, Yaoshu Wang, Min Xie, and Mengyi Yan. 2023. Splitting Tuples of Mismatched Entities. *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–29.

[38] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about Record Matching Rules. *PVLDB* 2, 1 (2009), 407–418.

[39] Wenfei Fan, Jianzhong Li, Shuai Ma, Nan Tang, and Wenyuan Yu. 2012. Towards certain fixes with editing rules and master data. *VLDBJ* 21, 2 (2012), 213–238.

[40] Wenfei Fan, Ping Lu, and Chao Tian. 2020. Unifying Logic Rules and Machine Learning for Entity Enhancing. *Sci. China Inf. Sci.* 63, 7 (2020).

[41] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research* 23, 1 (2022), 5232–5270.

[42] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323* (2022).

[43] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027* (2020).

[44] Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera. 2016. Big data preprocessing: methods and prospects. *Big Data Analytics* 1, 1 (2016), 1–22.

[45] Floris Geerts, Giansalvatore Mecca, Paolo Papotti, and Donatello Santoro. 2013. The LLUNATIC data-cleaning framework. *PVLDB* 6, 9 (2013), 625–636.

[46] Amir Gilad, Daniel Deutch, and Sudeepa Roy. 2020. On multiple semantics for declarative database repairs. In *SIGMOD*. 817–831.

[47] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F. Naughton, Narasimhan Rampalli, Jude W. Shavlik, and Xiaojin Zhu. 2014. Corleone: Hands-off crowdsourcing for entity matching. In *SIGMOD*. ACM.

[48] Songtao Guo, Xin Luna Dong, Divesh Srivastava, and Remi Zajac. 2010. Record Linkage with Uniqueness Constraints and Erroneous Values. *PVLDB* 3, 1 (2010), 417–428.

[49] Jiawei Han, Jian Pei, and Hanghang Tong. 2022. *Data mining: concepts and techniques*. Morgan kaufmann.

[50] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. 2021. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems* 34 (2021), 29335–29347.

[51] Alireza Heidari, Joshua McGrath, Ihab F. Ilyas, and Theodoros Rekatsinas. 2019. HoloDetect: Few-Shot Learning for Error Detection. In *SIGMOD*. 829–846.

[52] Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916* (2023).

[53] Benjamin Hilprecht and Carsten Binnig. 2021. ReStore - Neural Data Completion for Relational Databases. In *SIGMOD*. 710–722.

[54] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[55] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269* (2023).

[56] Zhipeng Huang and Yeye He. 2018. Auto-detect: Data-driven error detection in tables. In *Proceedings of the 2018 International Conference on Management of Data*. 1377–1392.

[57] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584* (2021).

[58] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[59] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel,

Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[60] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of Experts. *arXiv preprint arXiv:2401.04088* (2024).

[61] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[62] Kenji Kawaguchi, Zhun Deng, Xu Ji, and Jiaoyang Huang. 2023. How Does Information Bottleneck Help Deep Learning? *arXiv preprint arXiv:2305.18887* (2023).

[63] Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2022. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055* (2022).

[64] Keti Korini and Christian Bizer. 2023. Column Type Annotation using ChatGPT. *arXiv preprint arXiv:2306.00745* (2023).

[65] Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint arXiv:2110.03742* (2021).

[66] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[67] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668* (2020).

[68] Alexander K. Lew, Monica Agrawal, David A. Sontag, and Vikash K. Mansinghka. 2020. PClean: Bayesian Data Cleaning at Scale with Domain-Specific Probabilistic Programming. *CoRR* abs/2007.11838 (2020).

[69] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*. PMLR, 6265–6274.

[70] Bing Li, Wei Wang, Yifang Sun, Linhan Zhang, Muhammad Asif Ali, and Yi Wang. 2020. GraphER: Token-Centric Entity Resolution with Graph Convolutional Neural Networks.. In *AAAI*. 8172–8179.

[71] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263* (2023).

[72] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *PVLDB* 14, 1 (2020), 50–60.

[73] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584* (2020).

[74] Fan Liu, Tianshu Zhang, Wenwen Dai, Wenwen Cai, Xiaocong Zhou, and Delong Chen. 2024. Few-shot Adaptation of Multi-modal Foundation Models: A Survey. *arXiv preprint arXiv:2401.01736* (2024).

[75] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *arXiv preprint arXiv:2310.18339* (2023).

[76] Zifan Liu, Zhechun Zhou, and Theodoros Rekatsinas. 2020. Picket: Self-supervised Data Diagnostics for ML Pipelines. *CoRR* abs/2006.04730 (2020).

[77] Yuxuan Lou, Fuzhao Xue, Zangwei Zheng, and Yang You. 2021. Cross-token modeling with conditional computation. *arXiv preprint arXiv:2109.02008* (2021).

[78] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[79] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1948–1961.

[80] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data*. 865–882.

[81] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

[82] Andreas Maurer. 2004. A note on the PAC Bayesian theorem. *arXiv preprint cs/0411099* (2004).

[83] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. 2021. Capturing semantics for imputation with pre-trained language models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 61–72.

[84] 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 *SIGMOD*. 19–34.

[85] Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. 2022. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems* 35 (2022), 9564–9576.

[86] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. 2020. Missing data imputation using optimal transport. In *International Conference on Machine Learning*. PMLR, 7130–7140.

[87] Avanika Narayan, Ines Chami, Laurel Orr, Simran Arora, and Christopher Ré. 2022. Can foundation models wrangle your data? *arXiv preprint arXiv:2205.09911* (2022).

[88] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* 107 (2020), 107501.

[89] Nvidia. 2023. Nvidia TensorRT-LLM. https://github.com/NVIDIA/TensorRT-LLM

[90] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[91] George Papadakis, Dimitrios Skoutas, Emmanouil Thanos, and Themis Palpanas. 2020. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–42.

[92] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for Entity Matching. *arXiv preprint arXiv:2305.03423* (2023).

[93] Jinfeng Peng, Derong Shen, Nan Tang, Tieying Liu, Yue Kou, Tiezheng Nie, Hang Cui, and Ge Yu. 2022. Self-supervised and Interpretable Data Cleaning with Sequence Generative Adversarial Networks. *Proceedings of the VLDB Endowment* 16, 3 (2022), 433–446.

[94] Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Jing Yi, Weize Chen, Zhiyuan Liu, Juanzi Li, Lei Hou, et al. 2021. Exploring Universal Intrinsic Task Subspace via Prompt Tuning. *arXiv preprint arXiv:2110.07867* (2021).

[95] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.

[96] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *arXiv preprint arXiv:1702.00820* (2017).

[97] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems* 34 (2021), 17555–17566.

[98] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752 [cs.CV]

[99] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[100] Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. 2023. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705* (2023).

[101] Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. 2023. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285* (2023).

[102] Anthony Sicilia, Katherine Atwell, Malihe Alikhani, and Seong Jae Hwang. 2022. PAC-bayesian domain adaptation bounds for multiclass learners. In *Uncertainty in Artificial Intelligence*. PMLR, 1824–1834.

[103] Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, and Jianmin Wang. 2018. Enriching data imputation under similarity rule constraints. *IEEE transactions on knowledge and data engineering* 32, 2 (2018), 275–287.

[104] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260.

[105] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems* 34 (2021), 15920–15933.

[106] LLaMA-MoE Team. 2023. LLaMA-MoE: Building Mixture-of-Experts from LLaMA with Continual Pre-training. https://github.com/pjlab-sys4nlp/llama-moe

[107] Mergekit Team. 2023. Mergekit. https://github.com/cg123/mergekit

[108] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *Proceedings of the VLDB Endowment* 14, 11 (2021), 2459–2472.

[109] Simon Tihon, Muhammad Usama Javaid, Damien Fourure, Nicolas Posocco,

and Thomas Peel. 2021. DAEMA: Denoising autoencoder with mask attention. In *International Conference on Artificial Neural Networks*. Springer, 229–240.
[110] Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057* (2000).
[111] Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*. IEEE, 1–5.
[112] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[113] Sriram Varadarajan, Paul Miller, and Huiyu Zhou. 2013. Spatial mixture of Gaussians for dynamic background modelling. In *2013 10th IEEE international conference on advanced video and signal based surveillance*. IEEE, 63–68.
[114] Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. PromptEM: prompt-tuning for low-resource generalized entity matching. *arXiv preprint arXiv:2207.04802* (2022).
[115] Steven Euijong Whang and Hector Garcia-Molina. 2013. Joint entity resolution on multiple datasets. *VLDB J.* 22, 6 (2013), 773–795.
[116] Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. 2024. LLaMA Pro: Progressive LLaMA with Block Expansion. *arXiv preprint arXiv:2401.02415* (2024).
[117] Haoyuan Wu, Haisheng Zheng, and Bei Yu. 2024. Parameter-Efficient Sparsity Crafting from Dense to Mixture-of-Experts for Instruction Tuning on General Tasks. *arXiv preprint arXiv:2401.02731* (2024).
[118] Kevin Wu, Jing Zhang, and Joyce C Ho. 2023. CONSchema: Schema matching with semantics and constraints. In *European Conference on Advances in Databases and Information Systems*. Springer, 231–241.
[119] Richard Wu, Aoqian Zhang, Ihab Ilyas, and Theodoros Rekatsinas. 2020. Attention-based learning for missing data imputation in HoloClean. *Proceedings of Machine Learning and Systems* 2 (2020), 307–325.
[120] Richard Wu, Aoqian Zhang, Ihab F. Ilyas, and Theodoros Rekatsinas. 2020. Attention-based Learning for Missing Data Imputation in HoloClean. In *MLSys 2020*.
[121] Mohamed Yakout, Laure Berti-Équille, and Ahmed K. Elmagarmid. 2013. Don't Be SCAREd: Use SCalable Automatic REpairing with Maximal Likelihood and Bounded Changes. In *SIGMOD*. ACM.
[122] Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. MAVE: A product dataset for multi-source attribute value extraction. In *Proceedings of the fifteenth ACM international conference on web search and data mining*. 1256–1265.
[123] Qinyuan Ye, Juan Zha, and Xiang Ren. 2022. Eliciting and Understanding Cross-Task Skills with Task-Level Mixture-of-Experts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2567–2592.
[124] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *International conference on machine learning*. PMLR, 5689–5698.
[125] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *ICML*. PMLR, 5675–5684.
[126] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. 2023. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444* (2023).
[127] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023. Jellyfish: A Large Language Model for Data Preprocessing. *arXiv preprint arXiv:2312.01678* (2023).
[128] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023. Large Language Models as Data Preprocessors. *arXiv preprint arXiv:2308.16361* (2023).
[129] Jing Zhang, Bonggun Shin, Jinho D Choi, and Joyce C Ho. 2021. SMAT: An attention-based deep learning solution to the automation of schema matching. In *Advances in Databases and Information Systems: 25th European Conference, ADBIS 2021, Tartu, Estonia, August 24–26, 2021, Proceedings 25*. Springer, 260–274.
[130] Shuo Zhang and Krisztian Balog. 2019. Web table extraction, retrieval and augmentation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1409–1410.
[131] Shijie Zhang, Xin Yan, Xuejiao Yang, Binfeng Jia, and Shuangyang Wang. 2023. Out of the Box Thinking: Improving Customer Lifetime Value Modelling via Expert Routing and Game Whale Detection. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3206–3215.
[132] Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2023. TableLlama: Towards Open Large Generalist Models for Tables. *arXiv preprint arXiv:2311.09206* (2023).
[133] Yi Zhang and Zachary G Ives. 2020. Finding related tables in data lakes for interactive data science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1951–1966.
[134] 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.
[135] Jinze Zhao, Peihao Wang, and Zhangyang Wang. 2024. Generalization Error Analysis for Sparse Mixture-of-Experts: A Preliminary Study. *arXiv preprint arXiv:2403.17404* (2024).
[136] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems* 35 (2022), 7103–7114.
[137] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906* (2022).
[138] Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming sparsely activated transformer with stochastic experts. *arXiv preprint arXiv:2110.04260* (2021).

A APPENDIX

A.1 Additional Related Work

A.1.1 Multi-LoRA Architecture. Existing LLM inference system, *e.g.,* TensorRT-LLM[89], only support effective inference with a fixed LLM, which falls short on serving multiple LoRAs at once. Recently, some researchers have also proposed the application of multiple LoRAs to improve the performance of one same base models, or to be more advantageous over different aspects. LoraHub[55] trains several LoRAs, and choose different LoRA combinations during the inference phase, however LoraHub can only be applied to PLMs (*e.g.,* Flan-T5) and cannot easily extend to standard LLM architecture, *e.g.,* LLaMa, Mixtral; MOELoRA[75] applies MoE structure with fine-tuned language models, and boosting their performance for MTL in medical domain, however MOELoRA also applies a build-in MoE layer, and cannot support dynamic LoRA switch and generation, nor further fine-tuning of each experts. LoRAMoE[21] also design a plugin version of MoE, and designs a localized balancing constraints to coordinate parts of experts for task utilization over MTL with a typical build-in MoE layer. We argue that the above works modify the architecture of LLM, thus cannot easily combine with existing LLM efficient inference framework, and falls short in inference speed (typically 3×-12× slower than vLLM[7, 66]).

To serve the requirement of streaming pipeline inference with multiple LoRAs into a single machine without full size model merge, the dynamic LoRA switch technique is proposed. Since the detail is discussed in Section A.4, we only list a few representative works here. S-LoRA[101] is a system that can serve thousands of LoRA adapters from a single machine. Similarly, Punica[7] contains a new CUDA kernel design that allows batching of GPU operations for different LoRA models, achieves 12x higher throughput in serving multiple LoRA models compared to state-of-the-art LLM serving systems while only adding 2ms latency per token. In this work, we use the CUDA kernel design of Punica into vLLM[66] inference system for MELD.

A.2 Additional Experiment Result

In Table 4, we provide the general notations with corresponding descriptions.

In Table 5, we report the dataset and the few-shot labeled sample number we apply in our experiment.

In Table 6, we report performance of all local LLM methods compare to online LLM GPT-4 as an additional baseline.

In Figure 8, we report the attention weights of different tasks over different experts, generated by the router network $\mathcal{N}$ with selecting top-3 experts.

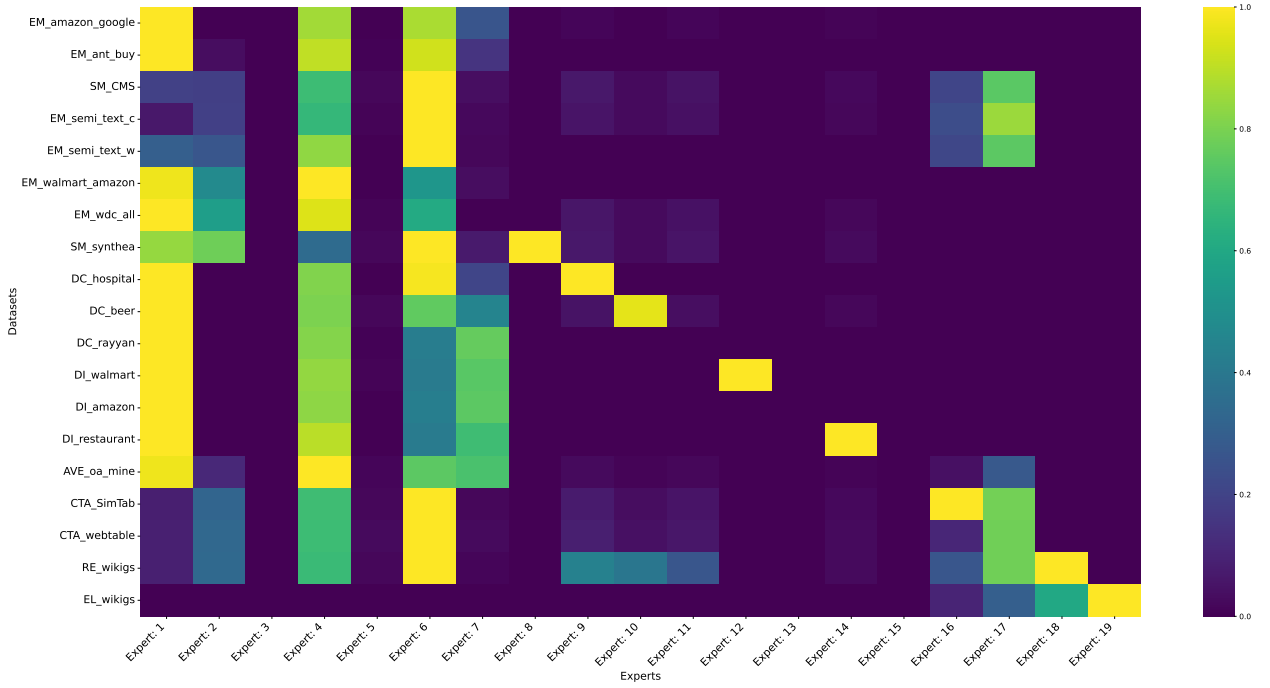


Figure 8: The heat map of the assignment by Router Network $\mathcal{N}$ for each task $\mathcal{T}_i \in \mathcal{T}$ and each expert $e_i \in \mathbf{E}^{aug}$. The vertical axis indicates different tasks(*i.e.,* dataset), while the horizontal axis represents its attention weights across all experts. For each block $B_{i,j}$, a brighter color means more attention weights(*i.e.,* probability to select as top-k experts) for i-th task over j-th expert, and vise versa. The i-th expert is initialized from i-th task(from top to down).

Table 4: General notations with corresponding descriptions.

Symbol	Description
$\mathcal{T}$	Data Preprocessing Task Family
$\mathcal{T}_1, \cdots, \mathcal{T}_n$	Different DP tasks in $\mathcal{T}$
$e_1, \cdots, e_n$	Experts over task $\mathcal{T}_i$
$\mathcal{X}_i, \mathcal{Y}_i$	Few-shot training data and label for task $\mathcal{T}_i$
$\tilde{\mathcal{X}}_i$	Unlabeled data for task $\mathcal{T}_i$
$\mathbb{X}_i$	All training data for task $\mathcal{T}_i$
t, T	A tuple/entry t in a relation table T
q_i	Natural-language query for LLM in task $\mathcal{T}_i$
$D_i, D^{\mathcal{T}_i}$	Demonstrations for task $\mathcal{T}_i$
$\mathcal{E}_i$	Meta-path over experts for task $\mathcal{T}_i$
$\mathcal{M}_{\text{RAG}}$,RAG	RAG model for retrieval and self-annotation
$\mathcal{M}_G$	LLM model for generation
$\boldsymbol{\theta}(D_i)$	Task vectors learned from D_i for task $\mathcal{T}_i$
$\mathbb{X}_i^{aug}$	Augmented training data via meta-path $\mathcal{E}_i$
$e_i^{aug}, \mathbf{E}^{aug}$	Refined expert from e_i for task $\mathcal{T}_i$
$\theta_{\mathcal{M}_{\text{RAG}}}, \theta_{\mathcal{M}_G}$	The parameter for $\mathcal{M}_{\text{RAG}}, \mathcal{M}_G$
$\mathcal{N}$	Router network for MoE

A.3 Complete Definition of DP tasks and Prompt Template

Entity Matching (EM) Given a pair of tuples t_1, t_2, our task is to infer whether they refer to the same entity. Formulated as:

$$(Ins^{\text{EM}}, D^{\text{EM}}, (t_1, t_2)), C^{\text{EM}}), C^{\text{EM}} = \{\text{match}, \text{mismatch}\}$$

EM is a binary classification task.

Data Cleaning (DC) Given a tuple t and an attribute a_i , the data cleaning over a relational table is a process that identifies and repairs such cell with the correct values, with a few annotated tuples D^{DC}. Formulated as:

$$(Ins^{\text{DC}}, D^{\text{DC}}, (t, a_i), C^{\text{DC}})$$

DC is an open-domain generation task, which means the output domain for C^{DC} has no limits.

Error Detection (ED) Given a tuple t and an attribute a_i, our task is to detect whether there is an error in the cell value of this attribute a_i for tuple t. Formulated as:

$$(Ins^{\text{ED}}, D^{\text{ED}}, (t, a_i), C^{\text{ED}}), C^{\text{ED}} = \{\text{error}, \text{correct}\}$$

ED is a binary classification task.

Column Type Annotation (CTA) Given a web table T and a set of pre-defined semantic types $\mathcal{L}$, our task is to annotate a column

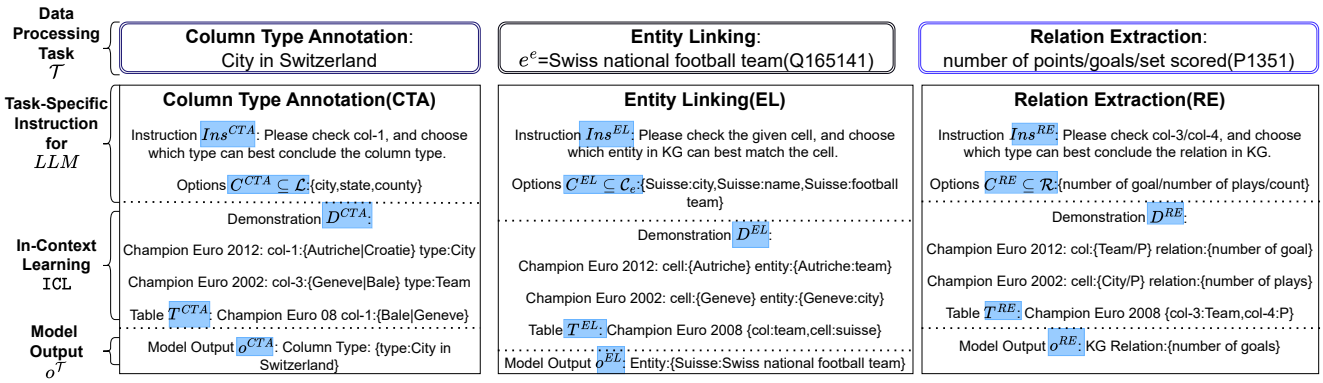


Figure 9: Illustration of Definition 2.1 in Column Type Annotation, Relation Extraction and Entity Linking tasks.

Table 5: Task, Datasets, and few-shot labeled sample number.

Task	Dataset	#Instance (few-shot)	#Instance (All)
Entity Matching (EM) & Blocking	Amazon-Google[73]	100	6874
	Walmart-Amazon[73]	100	6144
	WDC-All[73]	100	7229
	Ant-Buy[73]	100	5743
	Semi-Text-Watch[114]	80	5540
	Semi-Text-Computer[114]	80	12538
Error Detection(ED) & Data Cleaning(DC)	Hospital[79]	20	1000
	Rayyan[79]	20	1000
	Beer[79]	20	2410
Column Type Annotation(CTA)	SemTab19[32]	1920	7603
	WebTables[32]	15420	61023
Relation Extraction(RE)	WikiGS-RE[27]	6502	65026
Entity Linking(EL)	WikiGS-EL[27]	5441	54410
Schema Matching(SM)	CMS[129]	20505	20505
	Synthea[129]	23709	23709
Data Imputation(DI)	Walmart[83]	242	2421
	Amazon[83]	2001	20013
	Restaurant[83]	86	864
Attribute Value Extraction(AVE)	OA-mine[5]	286	1452

$h \in T$ with a semantic type $l \in \mathcal{L}$, such that all entities in column h hold the same type l. Formulated as:

$$(Ins^{\text{CTA}}, D^{\text{CTA}}, (T, h), C^{\text{CTA}}), C^{\text{CTA}} = \mathcal{L}$$

CTA is a close-domain ranking task, which means the output is within a pre-defined domain $\mathcal{L}$.

Relation Extraction (RE) Given a web table T and a set of pre-defined knowledge graph (KG) relations $\mathcal{R}$, our task is to annotate a column $h \in T$ with a KG relation type $r \in \mathcal{R}$, such that all entities in column h hold the same relation r. Formulated as:

$$(Ins^{\text{RE}}, D^{\text{RE}}, (T, h), C^{\text{RE}}), C^{\text{RE}} = \mathcal{R}$$

RE is a close-domain ranking task.

Table 6: Performance compared with GPT-4

Task	Dataset	MELD Few-shot	GPT-4	LLM Baseline Few-Shot	Mixtral Few-shot
EM	Amazon-Google	**83.41**	74.21	65.98	51.28
	Walmart-Amazon	**91.42**	90.27	42.03	39.78
	Ant-Buy	91.12	**92.77**	71.40	60.42
SM	CMS	**60.27**	59.29	59.29	31.01
	Synthea	56.00	**66.67**	40.00	23.53
DI	Restaurant	93.10	**97.75**	68.97	72.41
AVE	OA-mine	74.62	**80.20**	65.70	77.36

Entity Linking (EL) Given a web table T and a knowledge graph (KG) denoted as $\mathcal{G}$, entity linking (EL) is to link each cell $e \in T$ to its corresponding entity $e^{\mathcal{G}} \in \mathcal{G}$,Formulated as:

$$(Ins^{\text{EL}}, D^{\text{EL}}, (T, e), C^{\text{EL}}), C^{\text{EL}} \subseteq \mathcal{G}$$

EL is a close-domain ranking task.

Data Imputation (DI) Given a relational table T and a set of cells in T which values are missing, data imputation (DI) is to impute the missing values of cell e. Formulated as:

$$(Ins^{\text{DI}}, D^{\text{DI}}, (T, e), C^{\text{DI}})$$

DI is an open-domain generation task.

Attribute Value Extraction (AVE) Given a relational table T with schema R, attribute value extraction is to extract the value of additional attributes a for each tuple $t \in T$, while $a \notin R$. Formulated as:

$$(Ins^{\text{AVE}}, D^{\text{AVE}}, (T, a), C^{\text{AVE}})$$

AVE is an open-domain generation task.

Schema Matching (SM) Given a pair of attributes (a_1, a_2) with its corresponding description (d_1, d_2), schema matching is to judge whether h_1 and h_2 refer to the same attribute or not. Formulated as:

$$(Ins^{\text{SM}}, D^{\text{SM}}, ((a_1, d_1), (a_2, d_2)), C^{\text{SM}}), C^{\text{SM}} = \{\text{match}, \text{mismatch}\}$$

SM is a binary classification task.

Blocking (BLK) Given two relational tables T_1, T_2, for each tuple $t_1 \in T_1$, blocking means to find a tuple $t_2 \in T_2$, such that t_1, t_2 refer to the same entity, and vice versa.

Blocking is a close-domain ranking task. Due to the huge search space ($|T_1| \times |T_2|$), we do not apply LLM-based method to solve blocking problem for MELD.

For the prompt template, we give an example in Figure 9.

A.4 Mixture of Experts Implementation

Given query q_u, a well-trained router network $\mathcal{N}$ will assign k out of n fixed experts for processing it. According to theorem 1, fine-tuning on a small subset of parameters can perform well, so we apply low-rank adaptation[54](*a.k.a.* LoRA) to fine-tune $\mathcal{M}_G$ for training and refining each expert $e_i \in \mathbf{E}^{aug}$. So the storage of $\mathbf{E}^{aug}$ over n experts, are not n copies of LLM model, only n LoRA weights.

To mix $\mathcal{N}(q_u)$ experts into one single model for inference, there exists a host of MoE implementation work[106, 107, 116, 126]. However most of them requires further training, merge full LLM models instead of LoRA weights, or only supports non-LLM models, *e.g.,* Roberta/T5. To provide a simple yet effective test case, illustrating the generalization of the proposed MELD framework, we currently select the merging method implemented by Peft[81] officially, which merge and concat the LoRA weights to generate a new LoRA weight in same parameter size, a prevailing method in diffusion models[98].

For inference, to serve the requirement of streaming pipeline, which need to generate per-example experts in LLM reference, we implement a multi-LoRA query system based on S-LoRA[101] and vLLM[66]. Such system can support serving one base LLM model and up to 200 LoRA weights(*a.k.a.* experts) on one single GPU at once, dynamically generate and switch to new experts for incoming queries without significant computation efficiency loss during MoE inference.

A.5 Additional Proof

A.5.1 Proof of Theorem 1: Following in the finding of [52], fine-tuning LLM with in-context learning(ICL) and demonstrations, is equivalent to jointly train a learning algorithm $\mathcal{A}$ and rule application f.

In detail, for task $\mathcal{T}_i$, the learning algorithm $\mathcal{A}$ can map demonstration D_i into a intrinsic-level query-agnostic task vector $\boldsymbol{\theta}_i(D_i)$. Rule application f will take query $\mathcal{X}_i$ and task vector $\boldsymbol{\theta}_i(D_i)$ as input, and map $\mathcal{X}_i$ to the output $\mathcal{Y}_i$ as $f(\boldsymbol{\theta}_i(D_i), \mathcal{X}_i)$. An illustration is provided in the left part of Fig 3, where $\mathcal{A}, f$ represent different layers of a LLM model.

By applying ICL, the complexity of learning the task family $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$ is transferred to learn the corresponding task vectors set $\Theta = \{\boldsymbol{\theta}_1(D_1), \boldsymbol{\theta}_2(D_2), \cdots, \boldsymbol{\theta}_n(D_n)\}$. Then following the conclusion in [94], a low-dimensional intrinsic task subspace(ITS) $\mathbf{V}$ can be found to approximately cover Θ. *e.g.,* for Bert, a 250-dimensional subspace can recover 97% and 83% of the full prompt tuning performance for 100 seen tasks and 20 unseen tasks. For LLM, such conclusion also holds, and we provide a visualized t-SNE figure for task vectors of DP tasks, illustrated in the right part of Fig 3.

A.5.2 Proof of Theorem 2: The expected error $\epsilon_{\mathbb{C}}(h_N)$ of the adapted fine-tuned model in the target domain is defined, with the empirical error of the fine-tuned model in the target domain denoted as $\epsilon_C(h_N)$. The distance function representing the gap between the source domain S and the target domain C is defined as $D(S,C) = E[d(S,C)] + \widetilde{\lambda_{S,C}}$, where $\widetilde{\lambda_{S,C}}$ donates the adaptability of the upstream and downstream tasks, and $E[d(S,C)]$ representing the gap between the source and target domains, and $KL(h_N \,||\, h_0)$ indicating the difference between the original model and the fine-tuned model. Here, N represents the number of data samples. Therefore, the expected error bound $\epsilon_{\mathbb{C}}(h_N)$ of the fine-tuned model in the target domain, for adaptation, depends on its empirical error $\epsilon_C(h_N)$, the domain difference $E[d(S,C)]$ between the source and target domains, model capacity $KL(h_N \,||\, h_0)$, sample size N, and the adaptability of the upstream and downstream tasks $\widetilde{\lambda}_{S,C}$. Following the conclusion of [74], we give the proof of Theorem 2.

Proof. Maurer et al. [82] focused on the relationship between mean error and expected error in the target domain in 2004:

$$\Delta_{h_N}(C,\mathbb{C}) \leq \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \tag{3}$$

where $\Delta_{h_N}(C,\mathbb{C}) = |\epsilon_C(h_N) - \epsilon_{\mathbb{C}}(h_N)|$, $\epsilon_C(h_N)$ denotes the mean error in the target domain, $\epsilon_{\mathbb{C}}(h_N)$ represents the expected error in the target domain, C stands for the data in the target domain dataset, $\mathbb{C}$ represents all the possible data that may exist in the target domain, while h_0 represents the prior model, h_N represents the adapted model, and N represents the number of samples used from the target domain dataset.

Anthony Sicilia et al. [102] derived an empirical calculation of the domain gap from source and target domain data:

$$\Delta_{h_N}(S,C) \leq D(S,C). \tag{4}$$

And Crammer et al. [15] proposed the triangle inequality for errors:

$$\Delta_{h_N}(S,\mathbb{C}) \leq \Delta_{h_N}(S,C) + \Delta_{h_N}(C,\mathbb{C}). \tag{5}$$

Based on the above lemmas, Anthony Sicilia et al. proposed the PAC-Bayesian domain adaptation bound theory for multi-class classifiers:

$$\epsilon_{\mathbb{C}}(h_N) \leq \epsilon_S(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \tag{6}$$

Summarising the proof in [74], we choose to replace $\epsilon_S(h_N)$ in (6) with $\epsilon_S(h_N) - \epsilon_C(h_N) + \epsilon_C(h_N)$, which is organized into the form of (7):

$$\epsilon_{\mathbb{C}}(h_N) \leq \epsilon_S(h_N) - \epsilon_C(h_N) + \epsilon_C(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \tag{7}$$

as $\epsilon_S(h_N) - \epsilon_C(h_N) \leq |\epsilon_S(h_N) - \epsilon_C(h_N)|$ is constant, we can derive that:

$$\epsilon_{\mathbb{C}}(h_N) \leq |\epsilon_S(h_N) - \epsilon_C(h_N)| + \epsilon_C(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \quad (8)$$

according to the definition of $\Delta_{h_N}(S,C) = |\epsilon_S(h_N) - \epsilon_C(h_N)|$, it can be obtained:

$$\epsilon_{\mathbb{C}}(h_N) \leq \Delta_{h_N}(S,C) + \epsilon_C(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \quad (9)$$

substituting (4) into (9), the collation gives:

$$\epsilon_{\mathbb{C}}(h_N) \leq \epsilon_C(h_N) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}} + 2D(S,C). \quad (10)$$

And for the error upper bound of the mixture of experts, we directly follow the proof delineated in [135], leading to the following conclusion that the error bound holds with a probability of at least $1-\delta$ when selecting from N training samples:

$$O(4C\mathcal{R}_N(H) + 2\sqrt{\frac{2kd_N(1+ln(\frac{n}{k}) + d_N ln(2N) + ln(4/\delta)}{2N}})$$

where $\mathcal{R}_N(H)$ denotes the Rademacher complexity of the hypothesis space H associated with expert models, which is dependent solely on the data distribution and independent of the model's parameter count, $d_{\mathcal{N}}$ represents the Natarajan dimension of the gating network $\mathcal{N}$ within its hypothesis space $\mathcal{B}$, exclusively related to the gating network $\mathcal{N}$, $n = |\mathbf{E}|$ is the number of expert models in the expert set $\mathbf{E}$, and k is the number of experts selected per query. The loss function $l : Y \times R \rightarrow [0,1]$ for both the training expert model set $\mathbf{E}$ and the gating network $\mathcal{N}$ should be C-Lipschitz continuous. □

A.5.3 Proof of Theorem 3: The Mixture of Gaussians (MoG) distribution is widely recognized for its efficacy in practical applications [113], and it closely approximates the distribution of task vectors $\mathcal{T}_\Theta$. In alignment with the findings of [18], we posit that the experts e_i, which are associated with task vectors $\boldsymbol{\theta}_i$, follow a MoG distribution. This assumption ensures that the router network is capable of discerning the underlying clusters within the system. Building upon the results presented in [18], we proceed to provide a proof for Theorem 3 by summarising the proof in [18].

Proof. We define $\{G_c = \mathcal{N}(\mu_c, \sigma_c)\}_{c=1}^k$ as k spherical Gaussians in a mixture with weights $w_c = \frac{1}{k}$ w.l.o.g. Let $\{\mathcal{T}_c\}_{c=1}^k$ be cluster-specific vectors. For a sample x from $\frac{1}{k}\sum_{i=1}^k G_i$, the ground truth is $f(x) = \mathcal{T}_{c(x)}$, where $c(x)$ is the generating component. In k-means clustering, we need to find k clustering centers to minimize the sum of squared distances to their closest center. In the MoE architecture with weight sharing, the network function is defined as $y(x) = \sum_{i=1}^n g(x)_i e_i(x)$. Here $g(x)_i = 1$ when $e_i(x)$ is closest to x. We note that if we set the ground truth output to be the input x in our setting, then the best solution would be $e_i(x) = \mu_i$, which is what would optimize the k-means objective as well, thus framing k-means as a supervised learning. The loss functions are the same, so gradient descent steps are identical.

We extend the model by removing weight sharing to resemble a typical MoE model. The network is $y(x) = \sum_{i=1}^n g(x)_i e_i(x)$ with $g(x)_i = softmax(w_i^T x)$, where w_i is the router. The model has k Gaussian components G_i from a mixture with uniform weights $w_i = \frac{1}{k}$. Centers μ_i are randomly chosen with $\|\mu_i\|_2 \leq \alpha\sqrt{d}$, where $\alpha > 0$. The MoE architecture is similar, with trainable $g(x)_i$ and e_i. Then Theorem 3 can be restated based on these definitions.

Theorem 3 With $m = \Theta(dk\log k)$ samples from a k-component spherical Gaussian mixture in d dimensions, each component c-separated by a constant c, and an* MoE *architecture instantiated with $O(k\log k)$ experts, initializing router weights w_i to random training examples ensures that the router learns to correctly assign samples to their respective clusters.*

Let $X = \{x_i\}_{i=1}^m$ be the train samples and let $W_0 = \{w_j\}_{j=1}^{O(k\log k)}$ be $O(k\log k)$ randomly selected samples from X. We initialize router weights as W_0. A coupon collector[23]'s argument ensures with a probability not less than $1 - \frac{1}{pol y(k)}$. We group the experts sets $\mathbf{E}$ into k subsets $\{E[1:a_1], E[a_1+1:a_2], \cdots, E[a_{k-1}+1:a_k]\}$, with $E_i = E[a_i+1:a_{i+1}]$ comprising experts initialized with samples from the i-th Gaussian component G_i. These experts grouped in subset E_i are referred to as the specialists of cluster i.

Additionally, for the Gaussians are c-separated, for any distinct components $i \neq j$:

$$\Pr_{x_i\sim G_i, x_j\sim G_j}[\|x_i - x_j\| \leq C\sqrt{d}] \leq \frac{1}{d^{10}} \quad (11)$$

for a small enough constant C. Let E denote the set of all the experts. Note that there is a one-to-one association between an expert e_j and a router weight vector w_j. Partition the experts into k sets where E_k is the set of experts whose corresponding router weights were initialized using samples from the k-th component G_k. We call the set of experts E_k as cluster k's specialists.

When an input $x_i \sim G_k$ needs to be routed, the total probability weight the router assigns to experts not in E_k is $\leq e^{-O(d)}$. Hence, assuming $d \geq k$, the output is going to have contributions largely from the experts in E_k. This implies that the contribution of the gradient from the loss on input x_i is minimal on the experts not in E_k. Among the experts in E_k, there will exist at least one expert which each gradient will push in the direction of w_k. This can happen with multiple experts in E_k at the same time as well. Moreover, using the same argument as above, the experts in E_k do not get significantly affected by the gradients coming from examples x_j which are sampled from Gaussians other than G_k. By picking a small enough learning rate for the router, this leads to the following configuration after a logarithmic number of gradient descent steps. For every k, there will exist an expert in $j \in E_k$ such that $\|e_j - w_k\|_2 \leq o(1)$. In addition, the train loss will also be close to 0. Once this configuration has been reached, the subsequent steps of gradient descent only serve to boost the norms of the router weights without changing their direction by much which leads to a further cementing of the cluster based specialization of the experts. □

A.5.4 Proof of Theorem 4:

Proof. Suppose that $\mathcal{D}$ is the original data of task $\mathcal{T}$. Due to the

limited and fixed number of parameters for each ML-based expert, the number of inferences an expert can make is finite according to the information and coding theory. In other words, each expert $e_i \in \mathbf{E}$ can be modeled as a finite set of rules Φ_i, and thus the sequence of a meta-path querying $\mathcal{E}$ done by the experts is essentially the process of rule-based chasing C done by the rule sets:

$$\mathcal{E} = \{e_{j_1}, \cdots, e_{j_n}\} \iff C = \{\mathcal{D}_{c_0}, \mathcal{D}_{c_1}, \cdots, \mathcal{D}_{c_n}\} \quad (12)$$

and each inference in the meta-path $\mathcal{E}$ of task $\mathcal{T}$ made by an expert e_{j_i} can be considered as applying a specific rule ϕ from the rule set Φ_{j_i}:

$$\mathcal{D}_{c_{i-1}} \rightarrow_{\phi} \mathcal{D}_{c_i} \quad (13)$$

where $\mathcal{D}_{c_i}$ represents the view of the data after the i-th deduction of the expert, and $\mathcal{D}_{c_0} = \mathcal{D}$. That is to say, the process of applying an expert for each step of meta-path is actually the process of applying rules to chase with the data. Assuming that there will be no loops in the data view during the reasoning process, or that the chasing process converges when an identical data view appears again, we will proceed to prove: 1. Any chasing path is finite. 2. All chasing paths terminate at the same result.

1. Any chasing path is finite

Assuming the dataset consists of p rows and q columns, w.l.o.g., where each row represents an item and column represents an attribute of an item, and each attribute has at most a_i possible values, the actions involved in the expert's reasoning process include: 1. Filling in a specific value for an attribute in a cell with in the range of a_i. 2. Modifying the attribute value in a cell to another value with in the range of a_i. 3.Aligning two cells. Entity linkings can be considered as alignment of ID attribute cell. 4. Linking two columns with a definite type of relation. Note that although there may exist multiple associations between any pair of columns, the type of new relations are constrained by the rules in $\Phi = \bigcup_{i=1}^{|\mathbf{E}|} \Phi_i$ and $\mathcal{R}$, and hence at most $\min(|\Phi|, |\mathcal{R}|)$ type of relations can be added to $\mathcal{D}$, i.e., $\mathcal{D}_{c_0}$. Then, we have that $|C| \leq \min(|\Phi|, |\mathcal{R}|)^{\frac{q(q-1)}{2}} \times (\prod_{i=1}^{q} a_i)^p \times 2^{\frac{pq(pq-1)}{2}}$ and hence chasing path is finite.

2. All chasing paths terminate at the same result

We show this by contradiction. Assume that there exist two terminal chasing path $C_1 = \{\mathcal{D}_{c_0}, \mathcal{D}_{c_1}, ..., \mathcal{D}_{c_n}\}$, and $C_2 = \{\mathcal{D}'_{c_0}, \mathcal{D}'_{c_1}, ..., \mathcal{D}'_{c_m}\}$ of $\mathcal{D}$ by experts with different results. Because C_1 and C_2 have different results, we know that $\mathcal{D}_{c_0}$ is consistent and at least one of C_1 and C_2 is valid. Assume w.l.o.g. that C_1 is valid and the chase graph $\mathcal{D}_{c_n}$ is consistent. By analyzing the difference between C_1 and C_2, we show that C_1 is not terminal, a contradiction.

More specifically, since C_1 and C_2 have different results, there exist at least one cell or link and a chase step $\mathcal{D}'_{c_i} \rightarrow_{\phi'} \mathcal{D}'_{c_{i+1}}$ in C_2 such that $\mathcal{D}'_{c_{i+1}}$ extends $\mathcal{D}'_{c_i}$ w.r.t. the instantiation x; and x does not hold in $\mathcal{D}_{c_n}$ of C_1. Here the instantiation x replaces cell or link by ϕ. However, one can verify that $\mathcal{D}'_{c_i} \rightarrow_{\phi'} \mathcal{D}'_{c_{i+1}}$ is a valid chase step, which contradicts the assumption that sequence C_1 is terminal. Note that $\mathcal{D}'_{c_{i+1}}$ is the chase graph obtained from $\mathcal{D}'_{c_i}$ and x.

To show that $\mathcal{D}'_{c_i} \rightarrow_{\phi'} \mathcal{D}'_{c_{i+1}}$ is a chase step, we prove the following property by induction on the length of C_2: all attribute or links in $\mathcal{D}'_{c_i}$ ($i \in [0, m]$) are also in $\mathcal{D}_{c_n}$.

Basic case: At first, we consider the case when i = 0. Since C_2 starts with $\mathcal{D}'_{c_0} = \mathcal{D}_{c_0}$, property follows.

Inductive step: Assume that the property holds for $\mathcal{D}'_{c_i}$ ($i \leq j$). We next show that the property also holds for $\mathcal{D}'_{c_{j+1}}$. Suppose that step $\mathcal{D}'_{c_i} \rightarrow_{\phi'} \mathcal{D}'_{c_{i+1}}$, where ϕ' revised a cell or link of a item not in $\mathcal{D}'_{c_i}$. By the inductive hypothesis, we know that x is also a item in $\mathcal{D}_{c_n}$ with the same view in $\mathcal{D}'_{c_{i+1}}$. Then the attributes or links in $\mathcal{D}'_{c_{i+1}}$ must also be in $\mathcal{D}_{c_n}$. Since otherwise from the fact that x in $\mathcal{D}_{c_n}$ we can apply $\mathcal{M}$ to further extend $\mathcal{D}_{c_n}$, which contradicts the assumption that C_1 is terminal.

□