Efficient Mixture of Experts for Low Resource Fast Training Large Language Model Data Preprocessing

Mengyi Yan, Yaoshu Wang, Min Xie, Kehan Pang, Jianbin Qin, Rui Mao, and Jianxin Li*

***Abstract*—Data preprocessing (DP), which transforms raw data into a clean version, is a cornerstone of the data mining pipeline, yet it often demands costly, domain-specific rules or extensive annotated data. In this paper, we present MELD (Mixture of Experts on Large Language Models for Data Preprocessing), a universal solver for low-resource DP. MELD leverages a Mixture-of-Experts (MoE) architecture to train and combine specialized experts, but training them on large augmented datasets can be computationally intensive. To address this, we propose MELD-DS, a data- and cost-efficient framework that intelligently selects high-quality training subsets. At its core, MELD-DS features a novel influence function approximation algorithm that requires only 3-10% of the computational overhead of traditional methods. This efficiency enables MELD-DS to achieve comparable or even superior performance using just 30% of the original training data and runtime of the full MELD framework. Consequently, our solution requires no extra pre-training and is deployable on consumer-level hardware, such as a single 3090 GPU. We validate our approach through extensive experiments on 19 datasets over 10 DP tasks and provide theoretical proofs for the superiority of the MoE design. Results confirm that MELD and MELD-DS significantly outperform state-of-the-art methods in both effectiveness and efficiency.**

***Index Terms*—Mixture of Expert, LLMs, Data Preprocessing, Low-resource, Data Selection**

I. 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], [2]. Over the past decades, the focus has predominantly been on a limited number of tasks such as error detection (ED) [3], [4], data cleaning (DC) [5], data imputation (DI) [6], entity matching (EM) [7], entity linking (EL) [8], relation extraction (RE) [9], and column type annotation (CTA) [9], [10]. A primary challenge in this field arises from the diverse data distributions and requirements across various tasks, 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. Another major challenge in DP tasks is the scarcity of manual annotations, as users are often reluctant to label extensive data due to high costs. Additionally, resource constraints limit the feasibility of using multiple large-memory GPUs solely for DP. Finally, the prohibitive training cost keeps LLMs from real-world deployment when only a handful of GPUs are available. Therefore, the motivation for low-resource DP involves the need for effective methods that operate with few-shot data and minimal computational resources.

The advent of large language models (LLMs) 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 [11]. 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 [12], *i.e.,* more parameters gives better generative abilities and universal performance in DP. Consequently, most existing universal LLM-based DP solutions [13], [14] 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 [14].

In low-resource scenarios where both model parameters and the amount of labeled data are constrained, a straightforward approach is to perform training data augmentation through methods such as self-annotation [15], [16], data mixture [17] and synthetic data generation [18], [19]. However, as recent studies have revealed, expanding the training data without careful selection, particularly through reliance on synthetic data, introduces a series of severe challenges. On one hand, redundant and duplicate data introduce unnecessary computational overhead [20]. On the other hand, the inherent noise, biases, and even contradictory information within synthetic data can adversely affect the model's fine-tuning process, leading to performance degradation [16], [21], [22], and in extreme cases, may even risk model collapse. Therefore, given the constraints of limited computational budgets and hardware resources, how to efficiently select a high-quality subset from large-scale augmented data to improve data efficiency and ultimately boost model performance has become a critical and pressing problem to be solved.

To conclude, in low-resource environment, we face the

(Corresponding Author: Jianxin Li)

Mengyi Yan is with School of Artificial Intelligence, Shandong University, Shandong, 250100 China. (email:yanmy@sdu.edu.cn)

Min Xie and Yaoshu Wang are with Shenzhen Institute of Computing Sciences, Shenzhen 518100, China. (email:xiemin@sics.ac.cn; yaoshuw@sics.ac.cn)

Rui Mao and Jianbin Qin are with Shenzhen University, Shenzhen 518060, China, and also with Shenzhen Institute of Computing Sciences, Shenzhen 518100, China (email: mao@szu.edu.cn; qinjianbin@szu.edu.cn).

Kehan Pang and Jianxin Li are with School of Computer Science and Engineering, Beijing 100191, China. (email:pangkehan@buaa.edu.cn; lijx@buaa.edu.cn)

following challenges for establishing universal DP framework:

- The capability for a single model to learn representations across domains is inherently upper limited, even with more parameters [23].
- 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.
- Data valuation and efficient subset selection for universal DP tasks remain under-explored: fine-tuning over few-shot labeled data tends to overfit, while adding large amounts of irrelevant, redundant, or synthetic data wastes computation, degrades performance, and can trigger model collapse—especially for LLMs.

Motivated by these challenges, in this paper we develop two framework for universal DP, namely MELD and MELD-DS, focusing on model- and data-perspective.

In model perspective, to enhance representation ability among multi tasks within limited parameter size, we revisit the Mixture of Experts (MoE) architecture [24], which not only achieve high efficiency but also enhance generalization ability by facilitating information sharing between tasks [25]. Different from existing state-of-the-art MoE-based models [26]–[28] that embed gates within their parameters, we propose MELD (Mixture of Experts on Large Language Models for Data Preprocessing), an open-source system designed as a universal solver for low-resource DP tasks. The core of MELD is its standalone router network, a design that permits the independent, domain-specific training of experts and their flexible, plug-and-play composition during inference. The training pipeline begins by serializing raw data from various sources and employing an enhanced RAG system to create a self-annotated augmented dataset via cross-domain retrieval. Subsequently, a set of experts is initialized and refined on this data using LoRA-based training. Finally, the standalone router is trained to dynamically allocate the top-m most relevant experts.

In data perspective, to address the high training cost and the uncertain quality from using augmented data in MELD, we propose a data-efficient framework, MELD-DS, one of the first works focusing on highly efficient data selection methods for LLM-based universal DP. MELD-DS introduces a small-parameter proxy model to quickly measure the diversity and quality of the existing data among various tasks; then by using an efficient batch-level influence function approximation algorithm, it effectively measures the complexity of the data for task-specific training. MELD-DS integrates the above metrics via a unified optimization process to selects the most useful subset of data for each task, thus efficiently updating the parameters of each expert model in MoE architecture from a data-centric perspective. MELD-DS significantly reduces training costs while maintaining the downstream task performance.

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 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 propose MELD-DS, a data- and cost-efficient framework for LLM-based multi-task DP. It unified multiple data evaluation metrics into an integrated optimization process, while cooperating cost-efficient batch-level influence function approximation for maximizing downstream task performance.
- We design efficient MoE that could be fine-tuned in low resources, *e.g.,* a RTX 3090 GPU. Also we dynamically assign the top-m 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. MELD-DS further achieves comparable performance with only 30% training data from MELD, significantly improves data efficiency in low-resource environment.

This journal version presents a comprehensive extension of prior work [29], offering significant algorithmic advancements: (1) We discuss and evaluate existing metrics, *e.g.,* diversity/quality/complexity in data selection for LLM-based DP, and reveal the potential conflict among these metrics, highlighting the need of integrating those approaches. (Section V-A, Section VI-C) (2) We formulate data selection problem as an unified optimization process, incorporating multiple selection metrics as whole. (Section V-B) (3) We propose a batch-level IF approximation algorithm for efficient complexity measurement with theoretical guarantee, which only requires 3-10% computational overhead of traditional IF methods.(Section V-C) (4) We design a new data-efficient DP framework MELD-DS, incorporating the above measurement with cost-efficient implementation.(Section V-D) (5) We systematically evaluate our method MELD-DS, which achieves an average of 11% performance lead compare to baseline with high computing efficiency, and ablation study confirm the key components of MELD-DS are essential. (Section VI-C,Section VI-G). Proof of all theorem and additional experiment result is provided in full version [30].

II. Preliminary and Problem Definition

A. Preliminary

Mixture of Experts (MoE) The Mixture of Experts (MoE) architecture [24] 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) [31], [32].

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-m 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 [33], Mixtral [26], 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-m 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 [34] for Mixtral [26].

In light of this, we focus on external router networks as [35].

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 [36], [37]. Inspired by this, we can cast DP tasks into a MTL problem, and solve such problem by multi-gate MoE [32].

B. 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 [38]. In this work, we mainly focus on tabular data, including both relational tables and web tables.

Inspired by the success of *instruction-tuning* paradigm from NLP [39], 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. A complete list of illustrating examples can be accessed in full version [30].

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. □

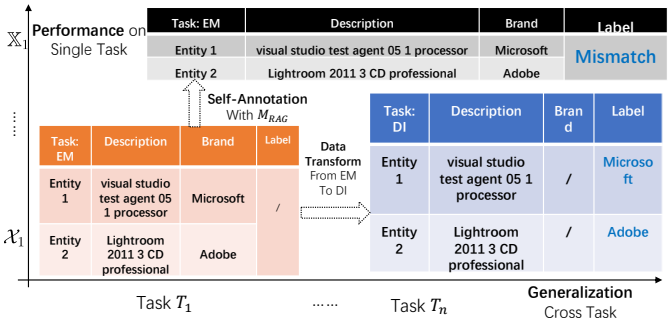


Fig. 1: Illustration of our data augmentation method.

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 1 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 trained and deployed in consumer-level small-memory GPUs with few-shot labeled data. □

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

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 MoE architecture, enable to answer (unseen) query of all $\mathcal{T}_i$.

III. Theoretical Analysis

Despite the empirical success of the MoE architecture, its theoretical understanding remains elusive. It is unclear why experts specialize on different inputs and how the router learns

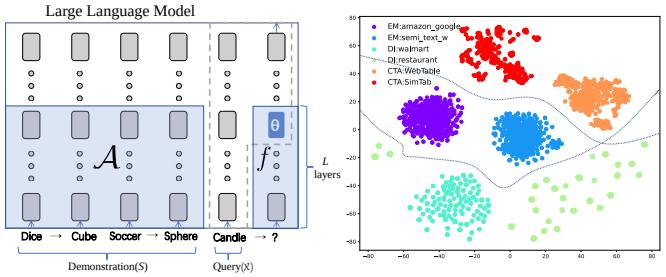


Fig. 2: Illustration of Intrinsic Task Subspace (ITS). The left part [40] shows how an LLM processes a task-specific query with demonstrations. The right is a t-SNE plot of task vectors for different DP tasks, where dotted lines indicate decision boundaries.

to dispatch data correctly. To this end, this section provides theoretical analysis to answer the following questions:

- Q1: Can various DP tasks over different domains be represented in a compact low-dimension space, *i.e.,* an *intrinsic task subspace*?
- Q2: Why can a single expert not fit well across multiple domains?
- Q3: How does the router learn to dispatch data to the right experts?

We provide three theorems to answer these questions. For brevity, full proofs are provided in our full version [30].

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*).* □

In response to Q1, Theorem 1 [36], [40] shows that diverse DP tasks can be learned within a compact Intrinsic Task Subspace (ITS). This implies that a small set of parameters can generalize across multiple tasks, as visualized in Figure 2.

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 *for* MTL *over all DP tasks. Let* C *be the sampled distribution over all tasks* $\mathcal{T}$ *with* N *samples, and* S *be the source domain distribution from* h_0*. The expected error* $\epsilon_{\mathbb{C}}(h_N)$ *for this single expert is upper bounded by:*

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

where $D(S,C)$ *is a distance function for the gap between the source domain* S *and the target domain mixture* C*.*

For a mixture of $n = |\mathbf{E}|$ *experts where* k *are selected per query, let* $\mathcal{R}_N(H)$ *be the Rademacher complexity of the expert hypothesis space and* $d_{\mathcal{N}}$ *be the Natarajan dimension of the gating network. The* MoE *error bound is:*

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

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

Theorem 2 [41], [42]addresses Q2. It 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 [43], 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 indicates that an MoE model with higher sparsity (fewer active experts k) can achieve better generalization, a conclusion validated in our experiments (Section VI-B).

Theorem 3: ***(Router learns Clusters in*** ITS***)*** *Given* $N = \Gamma(dk\log k)$ *samples drawn from a mixture of* k *spherical Gaussians 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.* □

To answer Q3, Theorem 3 [35], [44] guarantees that the router can learn to identify the underlying task clusters within the ITS. This convergence, however, depends on having suitable demonstrations and a well-structured training strategy.

IV. Mixture of Experts Architecture based on Large Language Models

The overall architecture of MELD is presented in Figure 3, 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 $\mathbf{E}$ of experts.
- **The meta-path search component**. It takes the enlarged training data $\mathbb{X}_i$ and the expert set $\mathbf{E}$ as input, and finds a meta-path $\mathcal{E}_i$ (*i.e.,* a sequence of experts in $\mathbf{E}$) 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 $\mathbf{E}$ as input, and fine-tunes the experts $\mathbf{E}$ 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-m experts for answering a query $q \in \mathbb{X}$.

Below we elaborate each component one by one.

A. 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

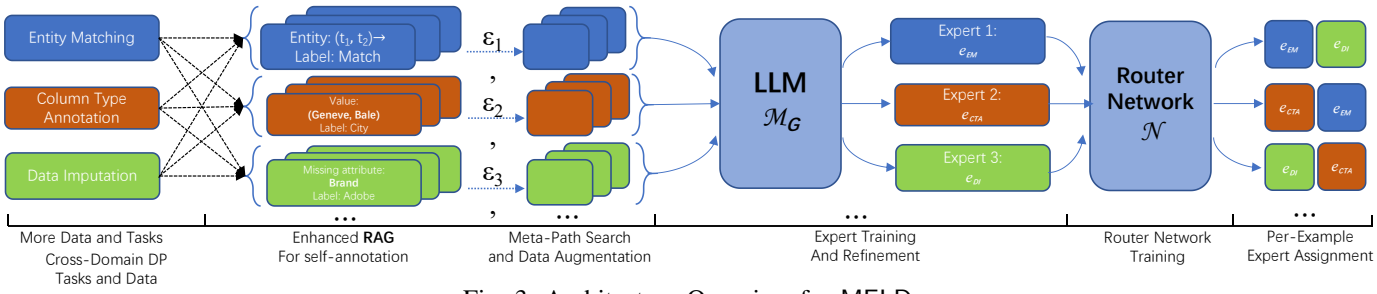


Fig. 3: Architecture Overview for MELD

structures, which are hard 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 [45] model $\mathcal{M}_{\mathsf{RAG}}$, and fine-tune the model with the contrastive learning loss [46]:

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

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 $\mathcal{M}_{\mathsf{RAG}}$ over $\mathbb{X}$, to add negative examples for $\mathcal{N}_q$.

Self-Annotation. When the training of $\mathcal{M}_{\mathsf{RAG}}$ is finished, we apply $\mathcal{M}_{\mathsf{RAG}}$ to self-annotate unlabeled queries in $\widetilde{\mathcal{X}_i}$. *e.g.,* for EM, given an unlabeled query $q_i \in \widetilde{\mathcal{X}_{EM}}$, $\mathcal{M}_{\mathsf{RAG}}$ 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 II-B to further enlarge $\mathbb{X}_i$ with labeled queries from other tasks. Figure 1 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 $\mathcal{M}_G$.

B. Heuristic Meta-path Search

There are a host of data augmentation methods [7], [15] 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. □

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.,* $\{e_{\mathsf{Blocking}}, e_{\mathsf{EM}}\}$ (resp. $\{e_{\mathsf{EL}}, e_{\mathsf{CTA}}\}$) is widely used in EM (resp. tabular interpretation learning [9]). To avoid error accumulation, we restrict the meta-path length to at most 3, and apply fixed augmentation strategy per task after meta-path search. We denote $\mathbb{X}_i^{\text{aug}}$ as augmented data for task $\mathcal{T}_i$, while $\mathbb{X}^{\text{aug}} = \cup \mathbb{X}_i^{\text{aug}}$.

C. 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 $\mathbb{X}_{\text{aug}}^i$ are augmented from $\mathcal{X}_i$ and may share a similar biased distribution. As discussed in [47], such distribution may lead to a higher empirical error $\epsilon_C(h_N)$ in Theorem 2. To alleviate such concern, we introduce the Min-Max optimization target guided by the information bottleneck theory [48], to improve the generalizability of each expert e_i.

We denote $\theta_{\mathcal{M}_{\mathsf{RAG}}}$ as the parameters for fine-tuned RAG model in Section IV-A, and $\theta_{\mathcal{M}_G}$ as the parameter of the base LLM-model of each expert, and $\mathbb{X}_i$ 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:

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

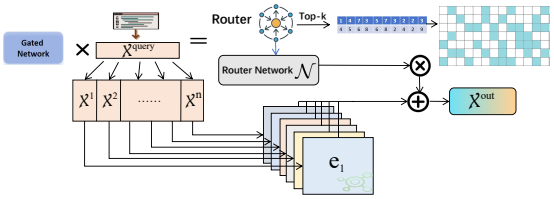


Fig. 4: Model architecture of MELD.

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) $\min \theta_{\mathcal{M}_{\mathsf{RAG}}}$ is implicitly enforced, by controlling the sample parameter for $\mathcal{M}_{\mathsf{RAG}}$ and meta-path $\mathcal{E}_i$ to add $\Delta\mathcal{X}_i$ 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 IV-A). Then we iteratively control the augmentation process 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 $\mathbf{E}^{\text{aug}}$. We apply low-rank adaptation [49] (*a.k.a.* LoRA) to fine-tune $\mathcal{M}_G$ for training and refining each expert $e_i \in \mathbf{E}^{\text{aug}}$.

Router Network. The information bottleneck theory also provides insight in optimizing $\mathcal{N}$. Given query q_i, on the one hand, the selected top-m 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.

Given a labeled query $q_u \in \mathbb{X}_u^{\text{aug}}$, let $\mathcal{N}(q_u)$ be the top-m experts selected by the sparse gated network $\mathcal{N}$ for q_u and task $\mathcal{T}_u$, (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:

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

In practice, Eq.2 can be approximated with contrastive training loss. Thus, we apply a transformer network that shares the encoding layers with $\mathcal{M}_{\mathsf{RAG}}$, 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 4 gives an illustration of $\mathcal{N}$.

V. Data-Efficient DP Framework MELD-DS

After self-annotation in Section IV-A and LLM-based data augmentation in Section IV-B, we accumulate a large augmented dataset $\mathbb{X}^{\text{aug}}$ (e.g., 14,856 samples for the CTA task in WebTable dataset). However, directly using $\mathbb{X}_i^{\text{aug}}$ to train expert models for task $\mathcal{T}_i$ is impractical due to two main issues: (i) synthetic data may contain false positives, biases, and contradictions [21], [22], which harm supervised fine-tuning (SFT); (ii) duplicates and redundancies [20] introduce unnecessary computational overhead, critical in LLM training.

To address these issues, we propose our data-efficient DP solution MELD-DS, dedicated to select a high-quality subset $S' \in \mathbb{X}^{\text{aug}}$ for the cost-efficient training stage. This method dynamically integrates multiple data value measures into an unified optimization process. Overall framework for MELD-DS is illustrated in Figure 5.

A. Data Selection Framework

To enhance the capability and generalization of LLMs for multi-task learning, it is essential to retrieve the most relevant and impact data from a large pool of synthetic data for the supervised fine-tuning (SFT) stage.

Formally, given the augmented data pool $\mathbb{X}^{\text{aug}}$, and a human-labeled ground-truth set $\mathcal{X}$, where $|\mathcal{X}| \ll |\mathbb{X}^{\text{aug}}|$, the task is to select a subset $S' \subset \mathbb{X}^{\text{aug}}$ to fine-tune the large model $\mathcal{M}$ toward $\mathcal{M}'$, minimizing the distribution discrepancy between S' and the actual distribution $\mathbb{X}$. (The discrepancy follows Theorem 2, and in this section we omit subscript i for simplicity)

Given the low-resource setting, the selection pipeline is constrained by a budget $|S'| \leq k$ and computation limits, relying on a proxy model $\mathcal{M}_{\text{proxy}}$, which has a significantly smaller parameter size than the large model $\mathcal{M}$.

In MELD-DS, the selection process is driven by three key widely-adopted metrics [50]: diversity, quality, and complexity. Each of these metrics has specific inputs and outputs:

Diversity: the input is the concatenated embedding vectors for the augmented data $\mathbb{X}^{\text{aug}}$, while the output is the diversity score $\mathcal{F}_{\text{div}}(S')$, which quantifies the spread and distinctiveness of the selected data subset S' over all data $\mathbb{X}^{\text{aug}}$. A higher diversity score indicates that the subset is more representative of the entire dataset, reducing redundancy.

Quality: the input is perplexity scores computed for each instance in $\mathbb{X}^{\text{aug}}$, both with and without demonstrations, from a small proxy model $\mathcal{M}_{\text{proxy}}$. The output is quality score $\mathcal{F}_{\text{qual}}(S')$, which evaluates the quality of the data instances in S', based on the sum of their perplexity ratio with and without demonstration. A higher ratio suggests the selected data can be considered as hard sample for further efficient training.

Complexity: The input is an initial subset S'_{init}, obtained from the previous optimization stage base on $\mathcal{F}_{\text{div}}(S')$ and $\mathcal{F}_{\text{qual}}(S')$; along with the batch division $\mathbf{B}$ from the entire dataset $\mathbb{X}^{\text{aug}}$. Using S'_{init}, the proxy model $\mathcal{M}_{\text{proxy}}$ is quickly warmed-up to $\mathcal{M}'_{\text{proxy}}$. The output is the complexity score $\mathcal{F}_{\text{compl}}(S')$, which estimates the impact of each instance on the model during training. This score is derived from the influence function (IF) of each data point, calculated using a batch-level IF approximation.

However, only consider single metric for selection may lead to unstable and suboptimal performance among different

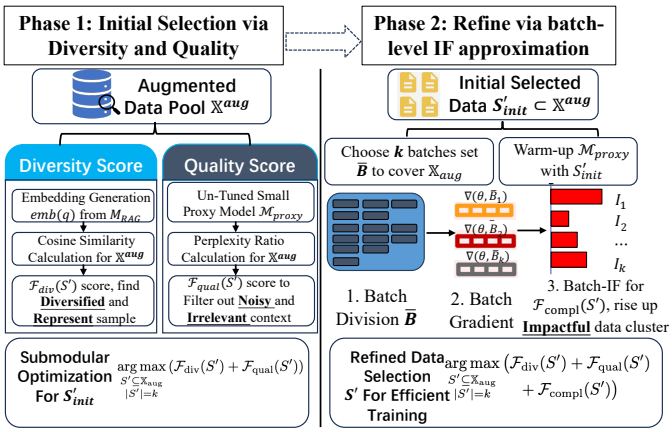


Fig. 5: Pipeline for data selection module in MELD-DS

tasks: training-free diversity-based [20] and quality-based [51] methods fail to align selected data with downstream task, while complexity-based solutions [52], [53] incur substantial computational overhead. Exiting work [54] also reveals the potential conflict among these metrics, *i.e.,* simply mixing these strategies also fails. (Discussed in Section VI-C). These observations motivate us to integrate multiple data selection methods into a unified optimization framework.

Submodular Optimization for Data Selection Our framework is inspired from submodular optimization method, which is widely-adopted for selecting representative and diverse data in NLP tasks [55], [56]. A set function $\mathcal{F} : 2^V \rightarrow \mathbb{R}$ on a finite set V is defined as submodular if it holds following properties:

Definition 5.1:(Submodular Function) For any set $A \subset B \subset V$ and an element $a \in V \setminus B$, the function $\mathcal{F}$ is submodular if it is monotonically non-decreasing and exhibits diminishing returns:

$$\mathcal{F}(A \cup \{a\}) - \mathcal{F}(A) \geq 0, \mathcal{F}(A \cup \{a\}) - \mathcal{F}(A) \geq \mathcal{F}(B \cup \{a\}) - \mathcal{F}(B)$$

□

The diminishing returns property makes submodular functions well-suited for data selection [57]. The objective is to find a subset $S' \subseteq \mathbb{X}^{\text{aug}}$ of a fixed size k that maximizes a submodular function $\mathcal{F}(S')$. Although this optimization problem is NP-hard [58], it can be efficiently approximated using a greedy algorithm [59] with a proven performance guarantee.

Theorem 4: *If S is the solution from a greedy search and S' is the optimal solution, then for a submodular function $\mathcal{F}(\cdot)$, the following inequality holds: $\mathcal{F}(S) \geq \left(1 - \frac{1}{e}\right) \mathcal{F}(S')$.* □

In Section V-B, we describe how MELD-DS initializes S'_{init} using a greedy search method, guided by submodular optimization techniques to balance data diversity and quality. In Section V-C, we add our batch-level influence function calculation method to refine the selection process, ensuring that the complexity of the selected data aligns with downstream task. Finally, in Section V-D, we present the complete MELD-DS framework, which dynamically adapts to various downstream tasks and proxy models.

B. Balancing Data Quality and Diversity

Diversity-based data quality measure. We first construct a representation vector for each instance $\mathbf{q} \in \mathbb{X}^{\text{aug}}$. Recall the embedding model $\mathcal{M}_{\text{RAG}}$ in Section IV-A, we define $\boldsymbol{emb}(\mathbf{q}) = \text{Concat}[\boldsymbol{emb}(\mathbf{q}), \boldsymbol{emb}(q_{\text{RAG}}), \boldsymbol{emb}(l)]$. We then formulate our objective $\mathcal{F}$ as a combination of diversity and quality scores. To measure diversity, given any symmetry and non-negative score function $w_{i,j}$ between $\boldsymbol{emb}(\mathbf{q}_i), \boldsymbol{emb}(\mathbf{q}_j)$, it is defined as:

$$\mathcal{F}_{\text{div}}(S') = \lambda_1 \sum_{\mathbf{q}_i \in \mathbb{X}^{\text{aug}}} \sum_{\mathbf{q}_j \in S'} w_{i,j} - \lambda_2 \sum_{\mathbf{q}_i \in S'} \sum_{\mathbf{q}_j \in S'} w_{i,j}, |\lambda_1| \geq 2|\lambda_2| \tag{3}$$

The first term rewards representativeness, while the second penalizes redundancy within S'. For simplicity, we choose $w_{i,j} = \cos(\cdot), \lambda_1 = 2, \lambda_2 = 1$.

Perplexity-based data quality measure. For quality, which is neglected by embedding-based diversity, we incorporate perplexity (PPL) ratio calculated by a small proxy model $\mathcal{M}_{\text{proxy}}$ to represent instruction-following difficulty [51], [60]. We compute PPL for an entry $\mathbf{q}$ with its label $\mathbf{l}$ with and without its demonstration context $\mathbf{q}_{\text{RAG}}$.

$$\text{PPL}_{\text{prior}}(\mathbf{q}) = \exp(-\frac{1}{|\mathbf{l}|} \sum_{i=1}^{|\mathbf{l}|} \log p_\theta(l_k | \mathbf{l}_{<k}, \mathbf{q})) \tag{4}$$

$$\text{PPL}_{\text{cond}}(\mathbf{q}) = \exp(-\frac{1}{|\mathbf{l}|} \sum_{i=1}^{|\mathbf{l}|} \log p_\theta(l_k | \mathbf{l}_{<k}, \mathbf{q}, \mathbf{q}_{\text{RAG}})) \tag{5}$$

The quality score $\mathcal{F}_{\text{qual}}$ is the sum of the PPL ratios over the selected set:

$$\mathcal{F}_{\text{qual}}(S') = \sum_{\mathbf{q}_j \in S'} f\left(\frac{\text{PPL}_{\text{cond}}(\mathbf{q}_j)}{\text{PPL}_{\text{prior}}(\mathbf{q}_j)}\right), f(x) = \begin{cases} x, & \text{if } 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

This function filters out instances that are either contradictory (ratio > 1) or too simple (ratio near 0). Considering both aspects, our final optimization objective is:

$$S = \underset{S' \in \mathbb{X}^{\text{aug}}, |S'|=k}{\arg\max} (\mathcal{F}_{\text{div}}(S') + \mathcal{F}_{\text{qual}}(S')) \tag{7}$$

Theorem 5: *$\mathcal{F}(S')$ is a submodular function.* □

By first computing embeddings and PPL scores for all data and then applying the greedy search to Eq. 7, we obtain an initial high-quality and diverse subset, denoted as S'_{init}. However, this approach relies on static metrics from untuned models, failing to capture the dynamic impact of data during SFT. To address this, Section V-C will introduce a computationally-efficient data influence approximation module.

C. Efficient Influence Calculation via Batch-Level Approximation

Influence functions (IF) offer a state-of-the-art approach to data selection by approximating the impact of an instance q on a model's parameters [61]. Given a model $\mathcal{M}'$ trained on a subset of $\mathbb{X}^{\text{aug}}$, the IF score for an instance z with respect to a validation set $\mathcal{X}$ is:

$$I_\theta(\mathcal{X}, z) = -\nabla L(\theta, \mathcal{X})(H + \lambda I)^{-1} \nabla L(\theta, z), \tag{8}$$

where $\nabla L(\cdot)$ is the gradient of the loss, H is the Hessian matrix, and λI is a regularization term.

However, applying IF to LLM presents two major computational bottlenecks:

(a) **Per-sample gradients**: Computing $\nabla L(\theta, z)$ for every instance is prohibitively slow due to repeated backpropagation and I/O overhead.
(b) **iHVP calculation**: Approximating the inverse Hessian $(H + \lambda I)^{-1}$(iHVP for short) is challenging for models with massive parameter counts.

For instance, DataInf [53] requires over 800GB of storage and 1,500 seconds to compute IF scores for just 6,000 samples on a 7B model, making it impractical for consumer hardware.

To address bottleneck (a), we exploit the data redundancy in $\mathbb{X}^{\text{aug}}$. Our key observation is that instances within the same semantic cluster exhibit similar influence and gradient distributions. We therefore propose a batch-level IF approximation method.

Instead of computing per-sample gradients, we calculate a single gradient for each data batch and use it to approximate the influence of all its members. The clusters (batches) $\bar{\mathbf{B}} = \{\bar{B}_1, \cdots, \bar{B}_k\}$ are formed using k-centered clustering [62], with the set S'_{init} from the previous step as initial centers, while the validation set is $\mathcal{X}_i$ for task $\mathcal{T}_i$. This batch-wise approach directly mitigates the per-sample cost. Naturally, we use the IF score $I(B_k)$ to represent the IF score of all samples x_i in the same cluster B_k. The theoretical error bound for the following batch-based and proxy-model based approximation is established in the following theorems:

Theorem 6: *The average error bound between the batch-IF score $I(B_j)$ and per-sample IF scores within that batch is $O(\frac{\sigma \cdot ||H^{-1}||}{\sqrt{|B|}})$, where σ is the average standard deviation of gradients within batches, $||H^{-1}||$ is the spectral norm of the inverse Hessian, and $|B|$ is the batch size. With a damping factor λ, the bound is no larger than $O(\frac{\sigma}{\lambda\sqrt{|B|}})$.* □

Theorem 7:*[Asymptotic Optimality of Proxy-Based Influence Sampling] Denote the optimal data selection policy as Q_{opt}, which minimizes asymptotic variance, samples data with a probability proportional to the true influence norm, $\|\psi(x)\|_2$. We can construct a feasible policy, $\tilde{Q}$, using an estimated norm, $\|\tilde{\psi}(x)\|_2$, from a proxy model, $\mathcal{M}_{proxy}$. If the proxy's influence estimate is consistent (i.e., $\sup_x \|\tilde{\psi}(x) - \psi(x)\|_2 \to 0$), then our feasible policy $\tilde{Q}$ achieves the **same minimum asymptotic variance** as the oracle policy Q_{opt}.* □

To tackle bottleneck (b), based on the above theorems, we introduce two further optimizations. First, we compute influence using a much smaller proxy model, $\mathcal{M}'_{\text{proxy}}$ (e.g., a 0.5B model with LoRA) warmed-up by S'_{init}, rather than the large target model $\mathcal{M}$, reducing the parameter space for gradient calculations by over 20×. Second, for large-scale computation, we leverage multi-GPU parallelism inspired by HyperInf [63]. For the iHVP step, we partition layer-wise gradients across different GPUs (model parallelism). For gradient calculation, we use a dynamic load-balancing strategy to distribute batches $\mathbf{B}$ across GPUs (data parallelism). These combined optimizations yield significant speedups, achieving up to 9× and 90× acceleration over baselines DataInf [53] and LESS [52], respectively.

D. overall framework for MELD-DS

After calculating the IF score for each instance, the iterative data selection pipeline is clear, with the optimization goal extended from Eq. 7 to:

$$S = \underset{S' \in \mathbb{X}^{\text{aug}}, |S'|=k}{\arg\max} (\mathcal{F}_{\text{div}} + \mathcal{F}_{\text{qual}} + \mathcal{F}_{\text{compl}}), \mathcal{F}_{\text{compl}}(S') = \sum_{\mathbf{q}_j \in S'} I(\mathbf{q}_j) \tag{9}$$

while $I(\cdot)$ is normalized to $(0, 1]$ to hold submodular for the updated $\mathcal{F}$, and Theorem 5 also holds.

Cost-Efficient Training Strategy. For the training process of MELD-DS, we replace the iterative training strategy in Eq. 1 with an offline data-mixture paradigm to improve efficiency. Let the augmented data for task $\mathcal{T}_i$ be denoted as $\mathbb{X}_i^{\text{aug}}$, and let $\overline{\mathbb{X}_i^{\text{aug}}} = \mathbb{X}_{\text{aug}} \setminus \mathbb{X}_i^{\text{aug}}$ denote the augmented data excluding task $\mathcal{T}_i$. The original training set $\mathbb{X}_i^{\text{train}}$ is selected from $\mathbb{X}_i^{\text{aug}}$ to optimize Eq.9. In contrast, an auxiliary set $\Delta\mathbb{X}_i^{\text{train}}$ is first filtered via the importance resampling method DSIR [20], aligning $\overline{\mathbb{X}_i^{\text{aug}}}$ with the target domain $\mathcal{X}_i$. This filtered subset is then used to select $\Delta\mathbb{X}_i^{\text{train}}$, which is optimized with respect to Eq.7 rather than Eq.9. This selection strategy enables reuse of the pre-computed $\mathcal{F}_{\text{div}}$ and $\mathcal{F}_{\text{qual}}$ for $\overline{\mathbb{X}_i^{\text{aug}}}$ without incurring additional influence-function calculations for each task $\mathcal{T}_i$. Finally, we directly mix $\mathbb{X}_i^{\text{train}}$ with $\Delta\mathbb{X}_i^{\text{train}}$ to train each expert e_i^{aug}, thereby eliminating the need for iterative parameter updates of e_i.

VI. Experimental Study

Our experiments focus on 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-DS effectively reduce the expert training time by selecting high-quality data? What is the tradeoff between diversity, complexity and quality?
- 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?
- How does the number of experts, as well as the meta-path selection, affect the overall performance of MELD?

A. Setup

Statistics. 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 $\leq 10\%$), as shown in column #Instance (few-shot). The selection of few-shot examples are kept the same among all methods.

DP Baselines For MELD, we categorized the baselines as follows. (1) Non-LLM methods . (a) ED: Raha [3], (b) DI:

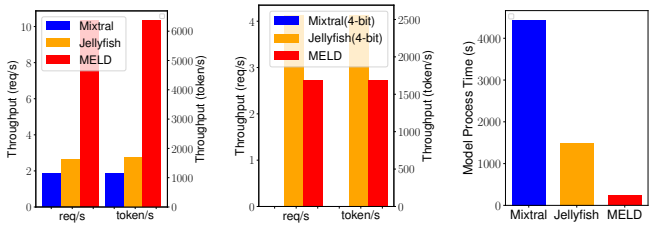
TABLE I: Overall Performance for MELD and baseline methods in DP tasks, the first results are marked in **bold**.

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	**89.35/76.17**	69.70/44.20	73.71/56.65	**89.35/76.17**
	WebTables	**96.41/78.76**	90.93/68.00	66.29/46.47	80.16/60.67
RE	WikiGS-RE	**77.19/60.89**	68.04/46.86	69.61/47.15	75.24/59.57
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**

IPM [6], (c) Blocking: DeepBlocker [64], (d) EM: Ditto [7] and PromptEM [16], (e) DC: Baran [5] and Garf [65], (f) CTA: RECA [10], (g) RE/EL: TURL [9], (h) SM: CONSchema [66] and SMAT [67], and (k) AVE: MAVE [68]. (2) LLM-based methods. JellyFish [11] 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 [69] which applies a 7B foundation model. For AVE task, we used ExtractGPT [70], 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 [26] (*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.

Data Selection Baselines We compare MELD-DS against representative baselines covering three primary data selection strategies. For **complexity-based** methods, we include DataInf [53], which uses an efficient closed-form expression to approximate per-sample influence function, and LESS [52], which relies on gradient projection. For **quality-based** selection, we evaluate against SuperFilter [51], a perplexity-based method, and QuRating [71], which leverages GPT-3.5 generated data for training quality judgment model. For **diversity-based** selection, we use DSIR [20], which employs importance resampling. To ensure a fair comparison, all methods are used to select the same data proportion c from the identical data pool $\mathbb{X}_{\text{aug}}$ and are used to fine-tune same backbone LLM $\mathcal{M}$.

Default Parameters. For Blocking, ED and EL, we only apply our RAG model $\mathcal{M}_{\text{RAG}}$ due to the large search space. For other tasks, we uses the LLM-based MoE system. Default number of k is set to 3, the number of iterations σ for expert refinement is set to 3, the demonstration number $|D_i|$ is set to 8. τ in RAG is 0.02. For MELD-DS, we set target selection ratio c to 30%, and batch-IF batch size $|B|$ to 8. The proxy model $\mathcal{M}_{\text{proxy}}$ is set to Qwen-2.5-0.5B [28].



(a) 4×3090 (b) 1×3090 (c) Process time

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

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/Macro-F1 score for CTA, RE tasks in 100-scale. By default, we only report Micro-F1 score for CTA/RE tasks in ablation studies.

Environment We select bge-large-en [72] as the backbone for the RAG models $\mathcal{M}_{\text{RAG}}$, and Mistral-7B [73] 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 with 24G VRAM. We use additional 2 A800 GPUs with 80G VRAM for other baselines with similar CPU configuration if necessary(*e.g.,*Mixtral/LESS/DataInf). Each experiment was run 3 times and the average is reported.

B. Effectiveness Evaluation

We compared the performance of MELD with various non-LLM and LLM baselines in Table I. 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.

In low-resource settings where labeled data is extremely limited, LLM baselines are prone to issues such as overfitting and hallucination problem, due to insufficient relevant demonstration data [74]; While non-LLM baselines often utilize rule-based approaches or rely on structural information, and are inherently robust in few-shot scenarios. MELD compensate such information incompleteness with $\mathcal{M}_{\text{RAG}}$ and self-distilled data augmentation with meta-path.

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.

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

TABLE II: Performance comparison of MELD-DS and baseline data selection methods. MELD indicates result trained over full augmented data $\mathbb{X}^{\text{aug}}$. The **first**, second, and third best selection results are marked correspondingly excluding MELD. $|S'|/|\mathbb{X}^{\text{aug}}|$ denotes the selection proportion c, as the selected data $|S'|$ over all augmented data $|\mathbb{X}^{\text{aug}}|$.

Task	Dataset	MELD (Full Set)	MELD-DS (Ours)	DataInf (ICLR'24)	QuRating (ICML'24)	LESS (ICML'24)	Superfilter (ACL'24)	DSIR (NIPS'23)	$\lvert S'\rvert/\lvert\mathbb{X}^{\text{aug}}\rvert$
CTA	SemTab19	89.35/76.17	**87.25/73.91**	76.60/66.45	86.59/72.37	78.05/64.78	65.83/53.34	71.87/55.83	25.40%
	WebTable	96.41/78.76	**95.42/78.24**	80.89/60.67	91.02/68.75	90.03/71.13	87.05/62.78	93.29/75.12	26.35%
RE	WikiTable	77.19/60.89	78.02/58.91	71.89/54.28	78.43/57.17	**79.30/59.79**	69.41/46.21	76.86/56.49	29.20%
ER	Semi-Text-w	78.28	75.97	20.51	68.08	n/a	24.16	**77.80**	34.05%
	Wdc	91.97	90.18	55.07	90.61	90.87	68.43	**91.37**	37.33%
	Abt-Buy	91.12	**89.57**	56.47	84.18	82.28	77.68	86.52	26.99%
	Amazon-Google	83.41	**79.24**	47.59	70.74	1.69	0.85	76.92	30.00%
	Walmart-Amazon	91.42	**86.78**	53.56	75.83	n/a	n/a	80.26	42.06%
DC	Hospital	95.01	**96.54**	96.45	95.26	92.50	95.74	94.17	13.56%
	Rayyan	82.15	**84.24**	83.86	79.90	81.94	14.37	84.09	32.37%
	Beer	97.30	99.30	99.04	98.40	99.44	98.57	**99.95**	34.64%
DI	Walmart	87.50	**89.42**	80.76	85.57	75.00	71.15	85.57	36.86%
AVE	OA-Mine	74.62	**77.92**	74.86	74.62	70.09	73.27	70.95	33.81%

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.

C. Data Selection Effectiveness Evaluation

TABLE III: Runtime and resource requirement for different data selection methods, for selecting 4k data over 14k data. Runtime overall remarks the overall runtime for each method, including selection and expert training time.

Selection Method	Initialize Time	Selection Time	VRAM Peak	Gradient Storage	Runtime Overall
MELD (1-GPU)	/	/	/	/	7522s
MELD (2-GPU)	/	/	/	/	3761s
MELD-DS(1-GPU)	115s	352s	14G	3.4G	2105s
MELD-DS(2-GPU)	63s	217s	7G	3.4G	1113s
DataInf	1200s	1500s	44G	22G	3533s
QuRating	/	540s	7.2G	/	1373s
LESS	1200s	25200s	68G	700M	27503s
SuperFiltering	/	165s	2.7G	/	998s
DSIR	/	375s	/	/	1208s

TABLE IV: Ablation study for MELD-DS, where w. indicates a component tested in isolation, w/o means without. The best result is marked in **bold**.

Dataset	w. diversity	w. quality	w. diversity+quality	MELD-DS
SemTab19	81.20/71.66	81.37/70.11	87.12/75.72	**87.25/73.91**
WebTable	92.72/74.01	89.12/70.14	94.05/74.80	**95.41/78.24**
WikiTable	77.81/56.00	73.01/53.10	76.20/55.68	**78.02/58.91**
Semi-Text-w	73.38	65.16	72.78	**75.97**
Rayyan	84.01	83.90	**84.79**	84.24
walmart	86.53	85.57	87.50	**89.42**

In this section, we report the performance comparison of MELD-DS and baseline selection methods in Table II, which metrics are kept the same with Table I. In Table III, we also provide the runtime and resource requirement for different selection methods, among with the total reduction of runtime for initializing single expert model.

Next we report our findings.

MELD-DS Achieves State-of-the-Art Performance. As shown in Table II, MELD-DS outperforms all baselines on the vast majority of datasets (10 out of 13), achieving an average performance gain of 11.37% over competitors. This result validates the efficacy of using small proxy models (0.5B for MELD-DS) for data selection. They yield performance competitive with, and sometimes superior to, training on the full dataset (MELD) while drastically reducing computational overhead.

Additionally, table III shows that MELD-DS requires only 30% runtime than MELD to achieve comparable performance, which data selection process can also be accelerated via multi-GPU parallelism—a capability lacking in baselines like DataInf and LESS. Furthermore, MELD-DS ensures data privacy by not requiring extra pre-training (*e.g.,* QuRating) or access to the test distribution (*e.g.,* DSIR).

Influence Functions are Insufficient for LLMs but a Valuable Component. Contrary to research on smaller models, our experiments show that methods relying solely on IF, like DataInf and LESS, are computationally expensive and deliver suboptimal results (Tables II and III). For example, LESS requires 68G of VRAM and increases total processing time. This is because IF identifies data that is *impactful* for training, which is not synonymous with high quality, potentially causing the LLM to overfit on noisy augmented data. However, our ablation study (Table IV) confirms that when integrated as one of three core components in MELD-DS, IF significantly enhances final performance, The incorporation of IF score amplifies the weight of high-impact data, which in turn enhances the model's convergence rate and instruction-following ability under a limited computational budget, thereby significantly improving the data efficiency of MELD-DS..

Diversity and Quality Metrics Have Significant Roles. Diversity-based methods like DSIR are training-free and efficient, affirming that data diversity is a key factor for LLM training. However, their reliance on the test set distribution is a key limitation. Our ablation study (Table IV) further shows that using only diversity can degrade model's instruction-following ability. Similarly, PPL does not strongly correlate

with downstream task performance, by the relatively weak results of the PPL-based Superfilter baseline and our own ablation (w. quality). While PPL can help initialize a model's general capabilities, it is not a sufficient metric on its own for selecting data for downstream tasks.

TABLE V: Performance(Micro-F1) and selection runtime(second) comparison of MELD-DS under different batch size $|B|$. AG is abbreviation for dataset Amazon-Google for task ER.

Dataset	$\|\mathbb{X}^{\text{aug}}\|$	$\|B\|=1$		$\|B\|=4$		$\|B\|=8$		$\|B\|=16$	
		F1	Time	F1	Time	F1	Time	F1	Time
WebTable	14856	78.60	1818s	93.81	517s	95.42	467s	94.05	473s
RE	6369	68.08	650s	77.93	176s	78.02	153s	76.28	122s
AG	6612	64.13	867s	71.56	196s	79.29	160s	47.16	126s

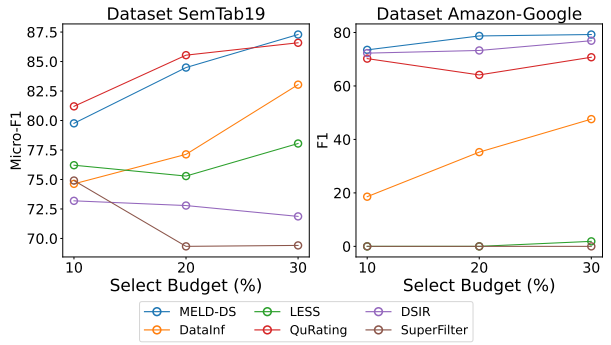


Fig. 7: Selection performance varying select budget $|S'|/|\mathbb{X}^{\text{aug}}|$

D. 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 [75]. 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 [75]. 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 approximate 50B 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 [76] 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.

TABLE VI: 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 VII: 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

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.

E. 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 VI. To prevent the domain overlap, we select 6 datasets with different domains, and limit the overall experts of MELD into 6.

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.

F. Ablation Study

We selected EM for ablation study with MELD, varying the following in Table VII:

- 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 $\mathcal{M}_{\mathsf{RAG}}$. 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.

G. Hyper-parameter Analysis

In this section, we discuss the impact of several key hyper-parameters, including expert number m for our main method MELD and the selection ratio c, as well as batch size B for our data-efficient method MELD-DS. We also provide additional experiment by varying backbone model $\mathcal{M}$, as well as comparing MELD and MELD-DS with online model GPT-4 in the full version [30].

Visualization of router network $\mathcal{N}$ Following [40], we present the t-SNE plot figure in Figure 2, 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.

Varying expert number m Figure 8 provides the sensitive analysis of number m. Initially, the performance rises with the number of experts. However, when $m \geq 4$, 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.

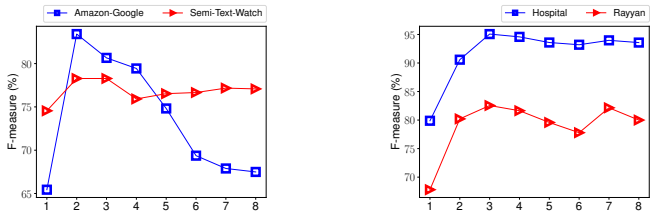
Varying selection ratio c. Figure 7 illustrates the performance of MELD-DS and other selection methods across different selection ratios $c \in \{10\%, 20\%, 30\%\}$. As the selection ratio increases, MELD-DS consistently outperforms other methods, demonstrating its robustness to varying budget constraints. Notably, MELD-DS achieves more stable improvements as the selection ratio grows, especially under larger data budgets, suggesting its ability to better leverage additional data.

Varying batch size $|B|$. Table V presents the performance and runtime for different $|B|$. As discussed, setting $|B| = 1$ to estimate per-sample gradients results in low gains due to high I/O overhead and inefficient GPU usage. Additionally, this disrupts the diversity-based cluster structure of MELD-DS, leading to noise data receiving high scores, which increases runtime and lowers performance. Conversely, excessively large batch sizes do not significantly improve efficiency, and the gains in I/O and IF computation are minimal when $|B| > 8$. Furthermore, larger batch sizes disrupt the clustering structure, increasing batch-IF estimation errors. Therefore, we default to $|B| = 8$ to balance computational efficiency and accuracy.

VII. Related Works

A. Data Preprocessing 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-4, [13], [14], [77], 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



(a) EM (b) DC

Fig. 8: Performance for different number of experts m

is unstable and costly. Worse still, data privacy cannot be guaranteed. There are also works on fine-tuning and deploying local LLMs [11], [69], [70] 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.

B. Mixture of Experts

MoE has been investigated in natural language processing [78], [79] 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 [78], [80]–[82], to sparsely select a single or m experts [35], [44], [83]. MoE has also be well invested in multi-task learning (MTL) [32], [84], including natural language generation [85], [86] and recommendation system [87].

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.

C. LLM-Based Data Selection

Data selection has become central to training and optimizing LLMs, mirroring a shift from model-centric to Data-Centric AI [88]. Two major lines have emerged. The first builds on influence functions, which estimate each sample's contribution to predictions [89]; because computing iHVPs is prohibitive for LLMs, efficient approximations are adopted [90]. LESS [52] estimates influence via cosine similarity of LoRA gradients for targeted instruction tuning, while DataInf [53] and HyperInf [63] derive closed-form, LoRA-specific estimators that markedly accelerate scoring. Still, how well IF scales to LLM-level data selection remains debated [91], and cost-efficient, multi-GPU approximations are underexplored. The second line targets the inherit probability/distribution for selecting data without training specific downstream model. Such methods include importance resampling [20], [92], proxy-model scoring [93], semantic deduplication [94], [95], and submodular optimization [96], however such training-free methods fails to select task-specific data regarding model parameter update.

With the escalating capabilities of LLMs, a "model-as-judge" paradigm now leverages LLMs to evaluate and filter data, reducing reliance on manual annotation [97], [98]. This

includes strong-to-weak approaches, such as QuRating [71] and DataMan [99], which use powerful models for pairwise quality comparisons to train specialized raters. In contrast, weak-to-strong methods [51], [60] employ smaller models for efficient filtering, capitalizing on a consistent perception of instruction difficulty.

VIII. Conclusions

In this paper, we study the problem of universal DP under low-resource settings and propose two frameworks, MELD and MELD-DS. We first revisit the MoE paradigm and design MELD with a standalone router network that enables domain-specific expert specialization and efficient deployment of open-source LLMs. We further propose MELD-DS, a data-centric framework that integrates diversity, quality, and complexity metrics into a unified optimization process, using efficient influence-function approximation to support scalable data selection. Together, MELD and MELD-DS significantly improve efficiency, scalability, and robustness, offering a practical solution for universal DP in constrained environments.

Acknowledgment

Portions of this work were presented at SIGKDD Conference on Knowledge Discovery and Data Mining, 2024 [29].

References

[1] S. García, S. Ramírez-Gallego, and et al., "Big data preprocessing: methods and prospects," *Big Data Analytics*, vol. 1, no. 1, pp. 1–22, 2016.
[2] S. A. Alasadi and W. S. Bhaya, "Review of data preprocessing techniques in data mining," *Journal of Engineering and Applied Sciences*, vol. 12, no. 16, pp. 4102–4107, 2017.
[3] M. Mahdavi *et al.*, "Raha: A configuration-free error detection system," in *SIGMOD*, 2019, pp. 865–882.
[4] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré, "Holoclean: Holistic data repairs with probabilistic inference," *PVLDB*, vol. 10, no. 11, 2017.
[5] M. Mahdavi and Z. Abedjan, "Baran: Effective error correction via a unified context representation and transfer learning," *PVLDB*, vol. 13, no. 12, pp. 1948–1961, 2020.
[6] Y. Mei, S. Song *et al.*, "Capturing semantics for imputation with pre-trained language models," in *ICDE*. IEEE, 2021, pp. 61–72.
[7] Y. Li, J. Li *et al.*, "Deep entity matching with pre-trained language models," *PVLDB*, vol. 14, no. 1, pp. 50–60, 2020.
[8] B. et al., "Tabel: Entity linking in web tables," in *ISWC*, 2015.
[9] D. et al., "Turl: Table understanding through representation learning," *ACM SIGMOD Record*, 2022.
[10] S. et al., "Reca: Related tables enhanced column semantic type annotation framework," *VLDB*, 2023.
[11] H. Zhang, Y. Dong *et al.*, "Jellyfish: Instruction-tuning local large language models for data preprocessing," in *EMNLP*, 2024, pp. 8754–8782.
[12] J. Kaplan, S. McCandlish *et al.*, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.
[13] A. Narayan, I. Chami *et al.*, "Can foundation models wrangle your data?" *PVLDB*, 2022.
[14] P. Li, Y. He *et al.*, "Table-gpt: Table fine-tuned gpt for diverse table tasks," *SIGMOD*, vol. 2, no. 3, pp. 1–28, 2024.
[15] M. et al., "Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond," in *SIGMOD*, 2021.
[16] P. Wang, X. Zeng *et al.*, "Promptem: prompt-tuning for low-resource generalized entity matching," *PVLDB*, vol. 16, no. 2, pp. 369–378, 2022.
[17] S. M. Xie, H. Pham *et al.*, "Doremi: Optimizing data mixtures speeds up language model pretraining," *NIPS*, 2023.
[18] A. Liu, B. Feng *et al.*, "Deepseek-v3 technical report," *arXiv preprint arXiv:2412.19437*, 2024.
[19] A. Dubey, A. Jauhri *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.
[20] S. M. Xie, S. Santurkar *et al.*, "Data selection for language models via importance resampling," *NIPS*, vol. 36, pp. 34 201–34 227, 2023.
[21] S. Dai, C. Xu *et al.*, "Bias and unfairness in information retrieval systems: New challenges in the llm era," in *KDD*, 2024, pp. 6437–6447.
[22] S. Farquhar, J. Kossen, L. Kuhn, and Y. Gal, "Detecting hallucinations in large language models using semantic entropy," *Nature*, vol. 630, no. 8017, pp. 625–630, 2024.
[23] O. Weller, M. Boratko *et al.*, "On the theoretical limitations of embedding-based retrieval," *arXiv preprint arXiv:2508.21038*, 2025.
[24] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.
[25] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.
[26] A. Q. Jiang, A. Sablayrolles *et al.*, "Mixtral of experts," *arXiv preprint arXiv:2401.04088*, 2024.
[27] X. Bi, D. Chen *et al.*, "Deepseek llm: Scaling open-source language models with longtermism," *arXiv preprint arXiv:2401.02954*, 2024.
[28] Q. Team, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.
[29] M. Yan, Y. Wang *et al.*, "Efficient mixture of experts based on large language models for low-resource data preprocessing," in *KDD*, 2024.
[30] "Code, datasets and full version," 2024, https://github.com/authurlord/MELD.
[31] D. Lepikhin, H. Lee *et al.*, "Gshard: Scaling giant models with conditional computation and automatic sharding," in *ICLR*, 2021.
[32] J. Ma, Z. Zhao *et al.*, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *KDD*, 2018, pp. 1930–1939.
[33] W. Fedus, B. Zoph *et al.*, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *Journal of Machine Learning Research*, vol. 23, no. 120, pp. 1–39, 2022.
[34] L. Gao, S. Biderman *et al.*, "The pile: An 800gb dataset of diverse text for language modeling," *arXiv preprint arXiv:2101.00027*, 2020.
[35] N. Dikkala, N. Ghosh *et al.*, "On the benefits of learning to route in mixture-of-experts models," in *EMNLP*, 2023.
[36] Y. Qin, X. Wang *et al.*, "Exploring universal intrinsic task subspace for few-shot learning via prompt tuning," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.
[37] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.
[38] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.
[39] J. Devlin, M.-W. Chang *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.
[40] R. Hendel, M. Geva *et al.*, "In-context learning creates task vectors," in *EMNLP*, 2023, pp. 9318–9333.
[41] F. Liu, T. Zhang *et al.*, "Few-shot adaptation of multi-modal foundation models: A survey," *Artificial Intelligence Review*, vol. 57, no. 10, p. 268, 2024.
[42] J. Zhao, P. Wang *et al.*, "Generalization error analysis for sparse mixture-of-experts: A preliminary study," in *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.
[43] M. Do, "Fast approximation of kullback-leibler distance for dependence trees and hidden markov models," *IEEE Signal Processing Letters*, vol. 10, no. 4, pp. 115–118, 2003.
[44] Z. Chen, Y. Deng *et al.*, "Towards understanding the mixture-of-experts layer in deep learning," in *NIPS*, 2022.
[45] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *EMNLP*, 2019, pp. 3982–3992.
[46] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
[47] S. Chen, E. Dobriban, and J. H. Lee, "A group-theoretic framework for data augmentation," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 9885–9955, 2020.
[48] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw)*. IEEE, 2015, pp. 1–5.
[49] E. J. Hu, P. Wallis *et al.*, "Lora: Low-rank adaptation of large language models," in *ICLR*, 2022.
[50] A. Albalak, Y. Elazar *et al.*, "A survey on data selection for language models," *Transactions on Machine Learning Research*, 2024.

[51] M. Li, Y. Zhang *et al.*, "Superfiltering: Weak-to-strong data filtering for fast instruction-tuning," in *ACL*, 2024.

[52] M. Xia, S. Malladi *et al.*, "Less: selecting influential data for targeted instruction tuning," in *ICML*, 2024.

[53] Y. Kwon, E. Wu, K. Wu, and J. Zou, "Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models," in *ICLR*, 2024.

[54] T. Bai, L. Yang *et al.*, "Efficient pretraining data selection for language models via multi-actor collaboration," in *ACL*, 2025.

[55] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in data subset selection and active learning," in *ICML*, 2015.

[56] K. Kirchhoff and J. Bilmes, "Submodularity for data selection in machine translation," in *EMNLP*, 2014.

[57] R. Iyer, N. Khargoankar, J. Bilmes, and H. Asanani, "Submodular combinatorial information measures with applications in machine learning," in *Algorithmic Learning Theory*. PMLR, 2021, pp. 722–754.

[58] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[59] V. Kaushal, G. Ramakrishnan, and R. Iyer, "Submodlib: A submodular optimization library," *arXiv preprint arXiv:2202.10680*, 2022.

[60] M. Li, Y. Zhang *et al.*, "From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning," in *NAACL*, 2024.

[61] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[62] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *ICLR*, 2018.

[63] X. Zhou, S. Fan, and M. Jaggi, "Hyperinf: Unleashing the hyperpower of the schulz's method for data influence estimation," *arXiv preprint arXiv:2410.05090*, 2024.

[64] S. Thirumuruganathan, H. Li *et al.*, "Deep learning for blocking in entity matching: A design space exploration," *PVLDB*, 2021.

[65] J. Peng, D. Shen *et al.*, "Self-supervised and interpretable data cleaning with sequence generative adversarial networks," *PVLDB*, 2022.

[66] K. Wu, J. Zhang *et al.*, "Conschema: Schema matching with semantics and constraints," in *ADBIS*, 2023.

[67] J. Zhang, B. Shin *et al.*, "Smat: An attention-based deep learning solution to the automation of schema matching," in *ADBIS*, 2021.

[68] L. Yang, Q. Wang *et al.*, "Mave: A product dataset for multi-source attribute value extraction," in *WSDM*, 2022.

[69] T. Zhang, X. Yue *et al.*, "Tablellama: Towards open large generalist models for tables," in *NAACL*, 2024, pp. 6024–6044.

[70] A. Brinkmann, R. Shraga *et al.*, "Extractgpt: Exploring the potential of large language models for product attribute value extraction," in *International Conference on Information Integration and Web Intelligence*, 2024.

[71] A. Wettig, A. Gupta *et al.*, "Qurating: Selecting high-quality data for training language models," in *ICML*, 2024.

[72] X. et al., "C-pack: Packaged resources to advance general chinese embedding," 2023.

[73] A. Q. Jiang, A. Sablayrolles *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[74] C. Zheng, H. Zhou *et al.*, "Large language models are not robust multiple choice selectors," in *ICLR*, 2023.

[75] W. Kwon, Z. Li *et al.*, "Efficient memory management for large language model serving with pagedattention," in *SOSP*, 2023.

[76] E. Frantar, S. Ashkboos *et al.*, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[77] H. Zhang, Y. Dong, C. Xiao *et al.*, "Large language models as data preprocessors," *arXiv preprint arXiv:2308.16361*, 2023.

[78] Y. Zhou, T. Lei *et al.*, "Mixture-of-experts with expert choice routing," *NIPS*, vol. 35, 2022.

[79] A. Komatsuzaki, J. Puigcerver *et al.*, "Sparse upcycling: Training mixture-of-experts from dense checkpoints," in *ICLR*, 2023.

[80] H. Hazimeh, Z. Zhao *et al.*, "Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning," *NIPS*, 2021.

[81] M. Lewis, S. Bhosale *et al.*, "Base layers: Simplifying training of large, sparse models," in *ICML*, 2021.

[82] S. Roller, S. Sukhbaatar *et al.*, "Hash layers for large sparse models," *NIPS*, 2021.

[83] S. Zuo, X. Liu *et al.*, "Taming sparsely activated transformer with stochastic experts," in *ICLR*, 2022.

[84] Q. Liu, X. Wu *et al.*, "Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications," *arXiv preprint arXiv:2310.18339*, 2023.

[85] T. Zadouri, A. Üstün *et al.*, "Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning," in *ICLR*, 2024.

[86] H. Wu, H. Zheng *et al.*, "Parameter-efficient sparsity crafting from dense to mixture-of-experts for instruction tuning on general tasks," in *EMNLP*, 2024.

[87] S. Zhang, X. Yan *et al.*, "Out of the box thinking: Improving customer lifetime value modelling via expert routing and game whale detection," in *CIKM*, 2023.

[88] D. Zha, Z. P. Bhat *et al.*, "Data-centric artificial intelligence: A survey," in *ACM Computing Surveys*, 2023.

[89] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *ICML*, 2017.

[90] R. Grosse, J. Bae *et al.*, "Studying large language model generalization with influence functions," *arXiv preprint arXiv:2308.03296*, 2023.

[91] T. Xia, B. Yu *et al.*, "Rethinking data selection at scale: Random selection is almost all you need," *arXiv preprint arXiv:2410.09335*, 2024.

[92] D. Grangier, S. Fan *et al.*, "Task-adaptive pretrained language models via clustered-importance sampling," in *ICLR*, 2025.

[93] Z. Yu, S. Das *et al.*, "Mates: Model-aware data selection for efficient pretraining with data influence models," *NIPS*, 2024.

[94] K. Lee, D. Ippolito *et al.*, "Deduplicating training data makes language models better," in *ACL*, 2022.

[95] A. K. M. Abbas, K. Tirumala *et al.*, "Semdedup: Data-efficient learning at web-scale through semantic deduplication," in *ICLR Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023.

[96] J. Qian, M. Sun *et al.*, "Sub-sa: Strengthen in-context learning via submodular selective annotation," in *ECAI*, 2024.

[97] L. Zheng, W.-L. Chiang *et al.*, "Judging LLM-as-a-judge with MT-bench and chatbot arena," in *NIPS*, 2023.

[98] J. Pang, J. Wei *et al.*, "Improving data efficiency via curating llm-driven rating systems," in *ICLR*, 2025.

[99] R. Peng, K. Yang *et al.*, "Dataman: Data manager for pre-training large language models," in *ICLR*, 2025.

[100] C. Wu, Y. Gan *et al.*, "Llama pro: Progressive llama with block expansion," in *ACL*, 2024.

[101] T. Zhu, X. Qu *et al.*, "Llama-moe: Building mixture-of-experts from llama with continual pre-training," in *EMNLP*, 2024.

[102] C. Goddard, S. Siriwardhana *et al.*, "Arcee's mergekit: A toolkit for merging large language models," in *EMNLP*, 2024.

[103] S. Mangrulkar, S. Gugger *et al.*, "Peft: State-of-the-art parameter-efficient fine-tuning methods," https://github.com/huggingface/peft, 2022.

[104] R. Rombach, A. Blattmann *et al.*, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.

[105] Y. Sheng, S. Cao *et al.*, "Slora: Scalable serving of thousands of lora adapters," in *MLSys*, 2024.

[106] A. Maurer, "A note on the pac bayesian theorem," *arXiv preprint cs/0411099*, 2004.

[107] A. Sicilia, K. Atwell *et al.*, "Pac-bayesian domain adaptation bounds for multiclass learners," in *Uncertainty in Artificial Intelligence*, 2022.

[108] K. Crammer, M. Kearns, and J. Wortman, "Learning from multiple sources." *Journal of Machine Learning Research*, vol. 9, no. 8, 2008.

[109] S. Varadarajan, P. Miller, and H. Zhou, "Spatial mixture of gaussians for dynamic background modelling," in *AVSS*, 2013.

[110] P. Erdos and A. Renyi, "On a classical problem of probability theory," *Magyar Tud. Akad. Mat. Kutató Int. Közl*, vol. 6, no. 1, pp. 215–220, 1961.

[111] L. Sagun, U. Evci *et al.*, "Empirical analysis of the hessian of over-parametrized neural networks," *arXiv preprint arXiv:1706.04454*, 2017.

[112] D. Ting and E. Brochu, "Optimal subsampling with influence functions," *Advances in neural information processing systems*, vol. 31, 2018.

[113] A. W. Van der Vaart, *Asymptotic statistics*. Cambridge university press, 2000, vol. 3.

Mengyi Yan received the Ph.D. degree from the School of Computer Science and Engineering at Beihang University, China, in 2025. He is currently an assistant professor in the School of artificial intelligence at Shandong University, Shandong, Chine. His research interests include Large Language Models, Database and Data Mining.

Yaoshu Wang received the Ph.D. degree from The University of New South Wales, Australia, in 2018. He is currently a Senior Researcher at Shenzhen Institute of Computing Science, Shenzhen, China. His research interests include data quality, big data, machine learning, and large language models.

Kehan Pang is currently working toward his Ph.D. degree in the School of Computer Science and Engineering at Beihang University, China. His research interests include graph neural networks and graph data mining.

Min Xie received her B.E. degree in a joint program of The Hong Kong University of Science and Technology (HKUST) and Sun Yat-sen University in 2014, and the Ph.D. degree in HKUST, in 2019. She is currently a resercher in Shenzhen Institute of Computing Sciences. Her research interests include database and data mining.

Jianbin Qin received the B.E. degree from Northeastern University, China, in 2007, and the Ph.D. degree from The University of New South Wales, Australia, in 2013. He is currently a Distinguished Professor in the College of Computer Science and Software Engineering at Shenzhen University, and a Research Scientist at Shenzhen Institute of Computing Sciences. His research interests include database theory, database systems, similarity query, information retrieval, and data management.

Rui Mao received the M.S. degree (2000) in Computer Science from the University of Science and Technology of China, and the Ph.D. degree (2007) in Computer Science from The University of Texas at Austin. He is currently a Distinguished Professor in the College of Computer Science and Software Engineering at Shenzhen University, and the Executive Director of Shenzhen Institute of Computing Sciences. His research interests include metric-space data processing, parallel computing, data mining, and machine learning.

Jianxin Li received his Ph.D. degree from the School of Computer Science and Engineering at Beihang University, China, in 2007. He is currently a professor with the School of Computer Science and Engineering, Beihang University, and a senior researcher with Beijing Advanced Innovation Center for Big Data and Brain Computing. His current research interests consist of social networks, machine learning, Big Data, and trustworthy computing.

TABLE VIII: 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 $\mathcal{T}_i$
$\widetilde{\mathcal{X}}_i$	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 $\mathcal{T}_i$
$\mathcal{M}_{\mathsf{RAG}}$,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 $\mathcal{T}_i$
$\theta_{\mathcal{M}_{\mathsf{RAG}}}, \theta_{\mathcal{M}_G}$	The parameter for $\mathcal{M}_{\mathsf{RAG}}, \mathcal{M}_G$
$\mathcal{N}$	Router network for MoE

TABLE IX: Task, Datasets, and few-shot labeled sample number.

Task	Dataset	#Instance (few-shot)	#Instance (All)
Entity Matching (EM) & Blocking(BLK)	Amazon-Google [7]	100	6874
	Walmart-Amazon [7]	100	6144
	WDC-All [7]	100	7229
	Ant-Buy [7]	100	5743
	Semi-Text-Watch [16]	80	5540
	Semi-Text-Computer [16]	80	12538
Error Detection(ED) & Data Cleaning(DC)	Hospital [5]	20	1000
	Rayyan [5]	20	1000
	Beer [5]	20	2410
Column Type Annotation(CTA)	SemTab19 [10]	1920	7603
	WebTables [10]	15420	61023
Relation Extraction(RE)	WikiGS-RE [9]	6502	65026
Entity Linking(EL)	WikiGS-EL [9]	5441	54410
Schema Matching(SM)	CMS [67]	20505	20505
	Synthea [67]	23709	23709
Data Imputation(DI)	Walmart [6]	242	2421
	Amazon [6]	2001	20013
	Restaurant [6]	86	864
Attribute Value Extraction(AVE)	OA-mine [70]	286	1452

APPENDIX A
ADDITIONAL EXPERIMENT RESULT

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

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

In Figure 10, we report the attention weights of different tasks over different experts, generated by the router network $\mathcal{N}$ with selecting top-3 experts.

APPENDIX B
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:

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

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

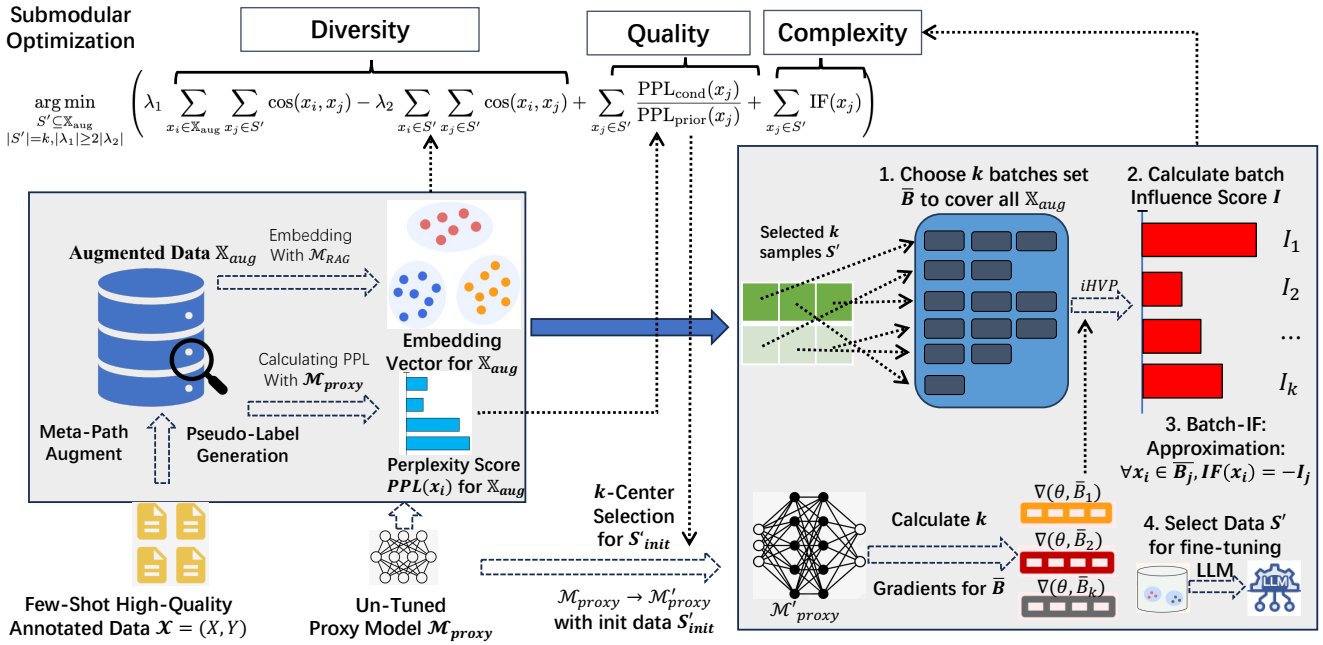


Fig. 9: Overall Framework of MELD-DS

TABLE X: Baseline Model Selection

Task	Dataset	Non-LLM Baseline	LLM Baseline
Entity Matching (EM)	Amazon-Google	Ditto	JellyFish
	Walmart-Amazon	Ditto	JellyFish
	WDC-All	Ditto	JellyFish
	Ant-Buy	Ditto	JellyFish
	Semi-Text-Watch	PromptEM	JellyFish
	Semi-Text-Computer	PromptEM	JellyFish
Error Detection(ED)	Hospital	Raha	JellyFish
	Rayyan	Raha	JellyFish
	Beer	Raha	JellyFish
Data Cleaning(DC)	Hospital	Baran	JellyFish
	Rayyan	Baran	JellyFish
	Beer	Baran	JellyFish
Column Type Annotation(CTA)	SemTab19	RECA	TableLLAMA
	WebTables	RECA	TableLLAMA
Relation Extraction(RE)	WikiGS-RE	TURL	TableLLAMA
Entity Linking(EL)	WikiGS-EL	TURL	TableLLAMA
Schema Matching(SM)	CMS	CONSchema	JellyFish
	Synthea	CONSchema	JellyFish
Data Imputation(DI)	Walmart	IPM	JellyFish
	Amazon	IPM	ExtractGPT
	Restaurant	IPM	JellyFish
Attribute Value Extraction(AVE)	OA-mine	MAVE	ExtractGPT

and repairs such cell with the correct values, with a few annotated tuples D^{DC}. Formulated as:

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

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:

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

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

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

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:

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

RE is a close-domain ranking task.

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:

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

EL is a close-domain ranking task. We do not apply LLM-based method to solve blocking problem for MELD.

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:

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

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:

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

AVE is an open-domain generation task.

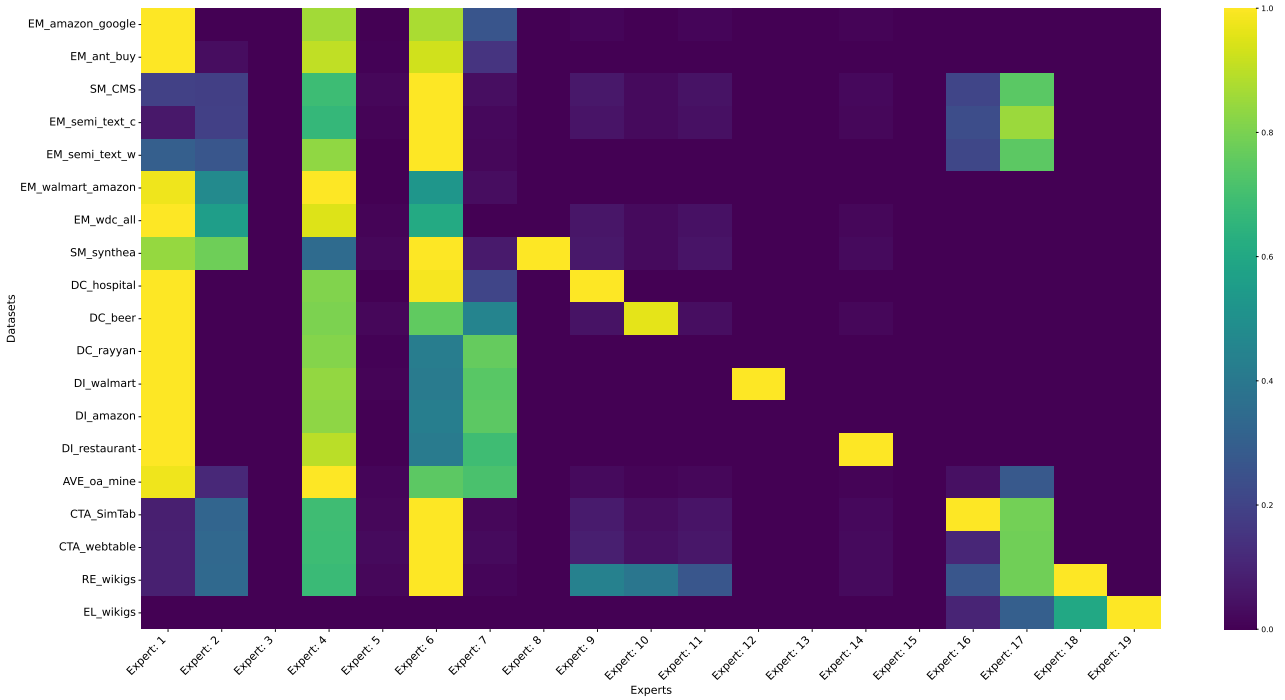


Fig. 10: The heat map of the assignment by Router Network $\mathcal{N}$ for each task $\mathcal{T}_i \in \mathcal{T}$ and each expert $e_i \in \mathbf{E}^{aug}$. The vertical axis indicates different tasks(*i.e.,* dataset), while the horizontal axis represents its attention weights across all experts. 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).

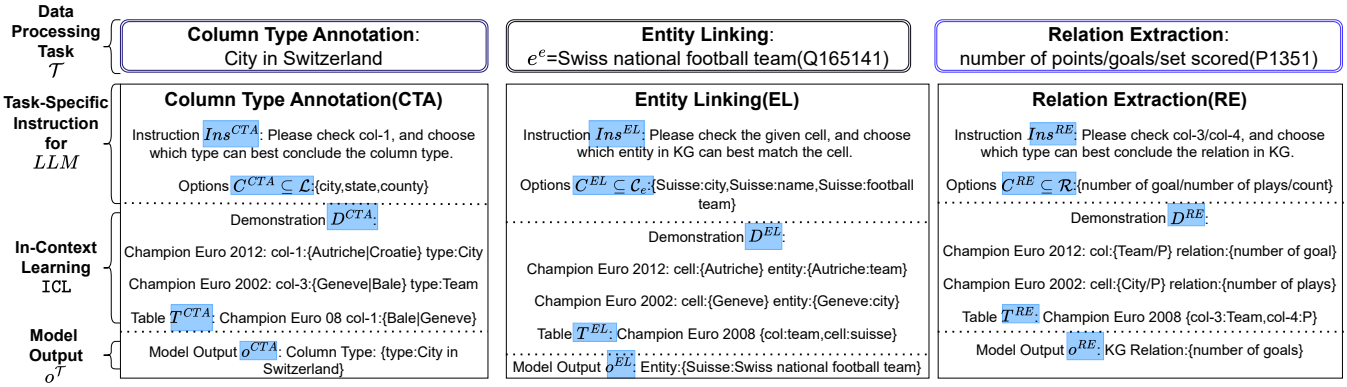


Fig. 11: Illustration of Definition 2.1 in Column Type Annotation, Relation Extraction and Entity Linking tasks.

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:

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

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 11.

APPENDIX C
SYNTHETIC DATA AUGMENTATION STRATEGY FOR DIFFERENT TASKS

APPENDIX D
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 [49](*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 [85], [100]–[102]. 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 [103] officially, which merge and concat the LoRA weights to generate a new LoRA weight in same parameter size, a prevailing method in diffusion models [104].

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 [105] and vLLM [75]. 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.

APPENDIX E
ADDITIONAL PROOF

1) Proof of Theorem 1: Following in the finding of [40], 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 2, 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 [36], 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 2.

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 [41], we give the proof of Theorem 2.

Proof. Maurer et al. [106] focused on the relationship between mean error and expected error in the target domain:

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

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. [107] derived an empirical calculation of the domain gap from source and target domain data:

$$\Delta_{h_N}(S,C) \leq D(S,C). \quad (11)$$

And Crammer et al. [108] proposed the triangle inequality for errors:

$$\Delta_{h_N}(S,\mathbb{C}) \leq \Delta_{h_N}(S,C) + \Delta_{h_N}(C,\mathbb{C}). \quad (12)$$

Based on the above lemmas, Sicilia et al. [107] proposed the PAC-Bayesian domain adaptation bound theory for multi-class classifiers:

$$\epsilon_{\mathbb{C}}(h_N) \leq \epsilon_S(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + \ln\sqrt{4N} - \ln(\delta)}{2N}}. \quad (13)$$

Summarizing the proof in [41], we choose to replace $\epsilon_S(h_N)$ in (13) with $\epsilon_S(h_N) - \epsilon_C(h_N) + \epsilon_C(h_N)$, which is organized into the form of (14):

$$\epsilon_{\mathbb{C}}(h_N) \leq \epsilon_S(h_N) - \epsilon_C(h_N) + \epsilon_C(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \quad (14)$$

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

$$\epsilon_{\mathbb{C}}(h_N) \leq |\epsilon_S(h_N) - \epsilon_C(h_N)| + \epsilon_C(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \quad (15)$$

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

$$\epsilon_{\mathbb{C}}(h_N) \leq \Delta_{h_N}(S,C) + \epsilon_C(h_N) + D(S,C) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}}. \quad (16)$$

substituting (11) into (16), the collation gives:

$$\epsilon_{\mathbb{C}}\left(h_N\right) \leq \epsilon_C\left(h_N\right) + \sqrt{\frac{KL(h_N||h_0) + ln\sqrt{4N} - ln(\delta)}{2N}} + 2D(S,C) \tag{17}$$

And for the error upper bound of the mixture of experts, we directly follow the proof delineated in [42], leading to the following conclusion that the error bound holds with a probability of at least $1-\delta$ when selecting from N training samples:

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

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. □

3) Proof of Theorem 3:: The Mixture of Gaussians (MoG) distribution is widely recognized for its efficacy in practical applications [109], and it closely approximates the distribution of task vectors $\mathcal{T}_\Theta$. In alignment with the findings of [35], 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 [35], we proceed to provide a proof for Theorem 3 by summarising the proof in [35].

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 [110]'s argument ensures with a probability not less than $1 - \frac{1}{poly(k)}$. 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 $i \neq j$:

$$\Pr_{x_i \sim G_i, x_j \sim G_j}[\|x_i - x_j\| \leq C\sqrt{d}] \leq \frac{1}{d^{10}} \tag{18}$$

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. Proof of Theorem 5

Proof. For simplicity, here we use X to replace $\mathbb{X}_{\text{aug}}$ for all dataset, and S to replace S' for selected dataset. We denote $A \subseteq B \subseteq X, S \subset X, a \in X$. for $\mathcal{F}_{\text{div}}$, we denote $\lambda_1 \geq 0, \lambda_2 \geq 0, |\lambda_1| \geq 2|\lambda_2|$. We proof that for $\forall a_1, a_2, a_3 \geq 0$,

$\mathcal{F}(S) = a_1\mathcal{F}_{\text{div}}(S)+a_2\mathcal{F}_{\text{qual}}(S)+a_3\mathcal{F}_{\text{compl}}(S)$ is a submodular function.

We first prove the monotonicity of $\mathcal{F}_{\text{qual}}(S)$, and $\mathcal{F}_{\text{compl}}(S)$ holds similarly.

Following Eq. 6, since each element $f(\frac{\text{PPL}_{\text{cond}}(x_j)}{\text{PPL}_{\text{prior}}(x_j)})$ in $\mathcal{F}_{\text{qual}}(S)$ is non-negative, we have:

$$\mathcal{F}_{\text{qual}}(S \cup \{a\}) - \mathcal{F}_{\text{qual}}(S) = f(\frac{\text{PPL}_{\text{cond}}(a)}{\text{PPL}_{\text{prior}}(a)}) \geq 0 \tag{19}$$

holds. For submodularity, we also have $\mathcal{F}_{\text{qual}}(A \cup \{a\}) - \mathcal{F}_{\text{qual}}(A) \geq \mathcal{F}_{\text{qual}}(B \cup \{a\}) - \mathcal{F}_{\text{qual}}(B)$. So $\mathcal{F}_{\text{qual}}$ is the submodular function.

Similarly, since $I(a)$ is also non-negative, $\mathcal{F}_{\text{compl}}$ is also a submodular function.

Next we concentrate on the monotonicity of $\mathcal{F}_{\text{div}}$. For $\mathcal{F}_{\text{div}}(S)$, we have:

$$\begin{aligned}\mathcal{F}_{\text{div}}(S \cup \{a\}) = \lambda_1 \sum_{i\in X}\sum_{j\in S}\omega_{i,j} + \lambda_1 \sum_{i\in X} w_{i,a} - \lambda_2 \sum_{i\in S}\sum_{j\in S} w_{i,j} \\ - \lambda_2 \sum_{i\in S} w_{i,a} - \lambda_2 \sum_{j\in S} w_{a,j} - \lambda_2 w_{a,a}\end{aligned} \tag{20}$$

Recall the definition of $\mathcal{F}_{\text{div}}$, while w holds symmetry and non-negative, *e.g.,* cosine similarity and radial basis function kernel. We have:

$$\begin{aligned}\mathcal{F}_{\text{div}}(A \cup \{a\}) - \mathcal{F}_{\text{div}}(A) &= \lambda_1 \sum_{i\in X} w_{i,a} - 2\lambda_2 \sum_{i\in A} w_{i,a} - \lambda_2 w_{a,a} \\ &\geq 2\lambda_2(\sum_{i\in X\setminus A} w_{i,a} - w_{a,a})\end{aligned} \tag{21}$$

Since $a \in X \setminus B$ and $X \setminus B \subseteq X \setminus A$, naturally $a \in X \setminus A$, thus $\mathcal{F}_{\text{div}}(A \cup \{a\}) - \mathcal{F}_{\text{div}}(A) \geq 0$ holds.

For submodularity, we have:

$$\begin{aligned}\mathcal{F}_{\text{div}}(A \cup \{a\}) &= \lambda_1 \sum_{i\in X} w_{i,a} - 2\lambda_2 \sum_{i\in A} w_{i,a} - \lambda_2 w_{a,a} \\ \mathcal{F}_{\text{div}}(B \cup \{a\}) &= \lambda_1 \sum_{i\in X} w_{i,a} - 2\lambda_2 \sum_{i\in B} w_{i,a} - \lambda_2 w_{a,a}\end{aligned} \tag{22}$$

Since $A \subseteq B$, therefore we have:

$$\sum_{i\in A} w_{i,a} \leq \sum_{i\in B} w_{i,a} \tag{23}$$

Naturally, $\mathcal{F}_{\text{div}}(A \cup \{a\}) \geq \mathcal{F}_{\text{div}}(B \cup \{a\})$ holds.

In a word, all $\mathcal{F}_{\text{div}}, \mathcal{F}_{\text{qual}}, \mathcal{F}_{\text{compl}}$ are all submodular functions, then $\mathcal{F}$ is also a submodular function.

In section V-B, we set $a_1 = a_2 = 1, a_3 = 0$, while in section V-D, we extend it to $a_1 = a_2 = a_3 = 1$. □

B. Proof of Theorem 6

Proof. We rewrite Theorem 6, under the following definitions and assumptions:

1) **Optimal Batch Division:** let batch $B_j \in \mathbf{B}$ contain $|B|$ samples, , selected such that their embeddings are the most similar within the entire dataset X.
2) **Gradient Aggregation**: the optimization goal for batch influence function, is to calculate the batch-averaged gradient:

$$\nabla L(B_j, \theta) = \frac{1}{|B|}\sum_{x_i\in B_j} \nabla L(x_i, \theta) \tag{24}$$

3) **Gradient Discrepancy:** we denote the maximum of per-sample gradient deviation within the batch as:

$$\epsilon_{\text{grad}} = \max_{x_i\in B_j} ||\nabla L(x_i, \theta) - \nabla L(B_j, \theta)|| \tag{25}$$

Recall Equation 8, we have the following batch-aggregated and per-sample IF score for sample $x_i \in B_j$:

$$\begin{aligned}I_\theta(B_j, \theta) &= -\nabla L(\theta, \mathcal{X})(H + \lambda I)^{-1}\nabla L(\theta, B_j) \\ I_\theta(x_i, \theta) &= -\nabla L(\theta, \mathcal{X})(H + \lambda I)^{-1}\nabla L(\theta, x_i)\end{aligned} \tag{26}$$

The absolute error between the per-sample and batch-averaged IF scores is bounded by:

$$|\mathcal{I}(x_i) - \mathcal{I}(B_j)| \leq \left\|(H + \lambda I)^{-1}\right\| \cdot \|\nabla L(x_i) - \nabla L(B_j)\| \cdot \|\nabla L(\mathcal{X})\| \tag{27}$$

The spectral norm of H^{-1} is proportional to the condition number $\kappa(H) = ||H|| \cdot ||H^{-1}||$, thus:

$$|\mathcal{I}(x_i) - \mathcal{I}(B_j)| \leq \epsilon_{\text{grad}} \cdot \kappa(H) \cdot \|\nabla L(\mathcal{X})\| \tag{28}$$

Then under the assumption that samples in batch B_j are embedding-similar, we further denote the per-sample gradients $\nabla L(x_i, \theta)$ are i.i.d. with variance σ_j^2, then the variance of the batch-averaged gradient is:

$$\text{Var}(\nabla L(B_j, \theta)) = \frac{\sigma_j^2}{|B|} \tag{29}$$

By Chebyshev's inequality, the gradient discrepancy scales as:

$$\epsilon_{\text{grad}} \leq O(\frac{\sigma_j}{\sqrt{|B|}}) \tag{30}$$

and we have error bound E_j for $|\mathcal{I}(x_i) - \mathcal{I}(B_j)|$ as:

$$E_j = O(\frac{\sigma_j \cdot \kappa(H)}{\sqrt{|B|}}) \tag{31}$$

while the average error across all batches can be bounded by

$$\mathbf{E} = \frac{1}{n}\sum_{j=1}^{n} E_j = O(\frac{\kappa(H)}{n\sqrt{|B|}}\sum_{j=1}^{n}\sigma_j) \tag{32}$$

replace $\sigma = \frac{1}{n}\sum_{i=1}^{n}\sigma_j$, then:

$$E = O(\frac{\sigma \cdot \kappa(H)}{\sqrt{|B|}}) \tag{33}$$

For accurate calculation of error bound, we can further replace $\kappa(H)$ with its original value $||(H + \lambda I)^{-1}||_2$. Denote the minimal singular value of H as σ_{min}, then after adding damping factor λ, $||(H + \lambda I)^{-1}||_2 \leq \frac{1}{\lambda+\sigma_{\text{min}}}$ holds.

We remark that Theorem 6 provides with several insights about lower the error bound for batch IF calculation:

- **Trade-off between batch size and cluster quality**. Larger batch size $|B|$ reduces ϵ_{grad}, however it significantly increase variance σ, as it may hamper the embedding similarity within batches, as well as significantly increase VRAM usage for selection. Thus we apply submodular optimization to strike such balance.
- **Redundant model parameter hampers batch-IF** As discussed in previous work [111], training over-parameterized models with limited labeled data can lead

to an ill-conditioned Hessian matrix, where the loss function's curvature becomes uneven, affecting optimization stability, with a significant higher $\kappa(H)$. Thus we apply batch IF under a relative small model M_{proxy}, instead of original large model M.

□

C. Proof of Theorem 8 and Theorem 9

The proof is inspired by [112]. Formally, Theorem 8 establishes the existence *and uniqueness* of an optimal regularized sampling distribution $Q_{\text{opt}} \in \Pi_{c,\alpha}$ that minimizes $\text{Tr}(V^Q)$, yielding PPS-style probabilities $\pi_i \propto \|\psi_{\theta_0}(x_i)\|$. In our paper's notation, Q corresponds to the data-selection policy used by MELD-DS, c is the target subsample ratio, and ψ_{θ_0} is the influence function of the large expert model M in MELD. Theorem 9 shows that replacing ψ_{θ_0} with a consistent estimate $\tilde{\psi}_\theta$ computed by a small pilot (proxy) model M_{proxy} trained on a representative subsample S preserves asymptotic optimality, so the plug-in estimator trained on the selected data attains the same limiting variance $V^{Q_{\text{opt}}}$. Together these results justify MELD-DS as a principled, low-compute selector that rapidly filters high-quality examples with M_{proxy}, thereby reducing labeled data requirements and GPU time while retaining the statistical efficiency of exact-influence sampling—consistent with the accuracy–efficiency gains observed in our experiments.

Theorem 8: *[Optimal Subsampling with Pilot Estimates] Let the full dataset $X = \{x_i\}_{i=1}^n$ follow a distribution P, and let $S = \{x_j\}_{j=1}^m$ be an i.i.d. subsample of X with $m = o(n)$. Suppose a pilot estimator M trained on S produces a consistent parameter estimate θ_0 (i.e., $\theta_0 \xrightarrow{m\to\infty} \theta$ in probability). Define the sampling distribution Q such that:*

1) ***Absolute Continuity**: $Q \ll P$ with Radon-Nikodym derivative $\frac{dQ}{dP}(x)$.*
2) ***Regularization**: $\exists \alpha > 0, c > 0$ where $\alpha \le \frac{dQ}{dP}(x) \le 1$ and $Q(\Omega) = c$.*

Then, for any estimator $\phi(\hat{\mathbb{P}}_n^Q)$ constructed from the weighted empirical measure $\hat{\mathbb{P}}_n^Q$, the asymptotic distribution satisfies:

$$\sqrt{\frac{n}{c}}\left(\phi(\hat{\mathbb{P}}_n^Q) - \phi(P)\right) \rightsquigarrow \mathcal{N}\left(0, V^Q\right),$$

where the asymptotic variance V^Q is:

$$V^Q = \int \psi(x)\psi(x)^\top \left(\frac{dP}{dQ}(x)\right)^2 dQ(x).$$

*Furthermore, there exists a unique **optimal sampling distribution** Q_{opt} that minimizes $Tr(V^Q)$, with sampling probabilities:*

$$\pi_i^{(opt)} \propto \|\psi_{\theta_0}(x_i)\|.$$

□

Proof. Weak Convergence of Weighted Empirical Process Under the assumptions on Q, the Radon-Nikodym derivative $\frac{dP}{dQ}$ is bounded and fixed. By the Donsker theorem for weighted empirical processes, the scaled difference converges weakly:

$$\sqrt{\frac{n}{c}}\left(\hat{\mathbb{P}}_n^Q - P\right) \rightsquigarrow R^Q,$$

where R^Q is a tight Gaussian process in $\ell^\infty(\mathcal{F})$, the space of uniformly bounded functions over a Donsker class $\mathcal{F}$.

Functional Delta Method Apply the functional delta method [113] to the Hadamard-differentiable functional ϕ. This yields:

$$\sqrt{\frac{n}{c}}\left(\phi(\hat{\mathbb{P}}_n^Q) - \phi(P)\right) \rightsquigarrow \int \psi(x) dR^Q(x),$$

where $\psi(x)$ is the influence function of ϕ at P. The limit is a mean-zero Gaussian process with covariance:

$$V^Q = \int \psi(x)\psi(x)^\top \left(\frac{dP}{dQ}(x)\right)^2 dQ(x).$$

Convex Optimization for Q_{opt} Define the class of measures:

$$\Pi_{c,\alpha} = \left\{ Q \ll P \;\middle|\; Q(\Omega) = c,\ \alpha \le \frac{dQ}{dP} \le 1 \text{ a.s.} \right\}.$$

This set is closed and convex. The objective $\text{Tr}(V^Q)$ is strictly convex in Q because:

- V^Q is quadratic in $\frac{dP}{dQ}$,
- The trace operator preserves convexity.

By the strict convexity of $\text{Tr}(V^Q)$ over $\Pi_{c,\alpha}$, there exists a unique minimizer Q_{opt}.

Explicit Form of Q_{opt} The optimal distribution Q_{opt} satisfies:

$$\frac{dQ_{\text{opt}}}{dP}(x) \propto \min\left\{\lambda \|\psi(x)\|, 1\right\},$$

where λ is chosen such that $\int \frac{dQ_{\text{opt}}}{dP}(x) dP(x) = c$. This follows from solving the Lagrangian of the convex optimization problem.

Computational Considerations and Proxy Model Error In a low-resource environment, the small model M_{proxy} is used to estimate the influence function ψ_{θ_0}. The approximation error between M_{proxy} and M must be controlled to ensure that the sampling distribution Q_{opt} remains close to optimal. Using M_{proxy} reduces the overall computational cost by selecting a smaller but representative sample, thus improving the efficiency of the data preprocessing while maintaining a reasonable approximation of the large model's performance. The error introduced by using M_{proxy} must be sufficiently small, as discussed in the error analysis.

□

Theorem 9: *Let $\tilde{\psi}_\theta$ denote the estimated influence function derived from θ_0, and let $\tilde{\pi}_n$ be the regularized sampling probabilities generated by proportional-to-size (PPS) sampling with weights proportional to $\|\tilde{\psi}_\theta(x_i)\|$. Assume:*

1) ***Consistency of** $\tilde{\psi}_\theta$: $\tilde{\psi}_\theta = \psi_{\theta_0} + o_p(1)$, where ψ_{θ_0} is the true influence function under θ_0.*
2) ***Asymptotic scaling**: $n_{sub}, n \to \infty$ with $n_{sub}/n \to c > 0$, where n_{sub} is the expected subsample size.*

Then, the plug-in estimator $\hat{\theta}$ trained on the subsampled data satisfies:

$$\sqrt{n_{sub}}\left(\hat{\theta}-\theta\right) \rightsquigarrow \mathcal{N}\left(0, V^{Q_{opt}}\right),$$

where $V^{Q_{opt}}$ is the minimal asymptotic variance defined by the optimal sampling distribution Q_{opt}. □

Proof. *Optimal sampling design under true influence* By Theorem 8, the optimal sampling distribution Q_{opt} minimizes the trace of the asymptotic variance V^Q, and its sampling probabilities are proportional to $\|\psi_{\theta_0}(x_i)\|$. This follows directly from the convex optimization argument in Theorem 8.

Consistency of estimated influences Since S is an i.i.d. subsample of X, the pilot estimate θ_0 trained on S is consistent for θ under standard regularity conditions (e.g., correct model specification and finite variance). Consequently, the estimated influence function $\tilde{\psi}_\theta$ satisfies:

$$\tilde{\psi}_\theta(x_i) = \psi_{\theta_0}(x_i) + o_p(1),$$

where the $o_p(1)$ term vanishes as $m \to \infty$.

Error analysis of sampling probabilities Let $\pi_i^{(\text{opt})} \propto \|\psi_{\theta_0}(x_i)\|$ and $\tilde{\pi}_i \propto \|\tilde{\psi}_\theta(x_i)\|$. The estimation error in the sampling probabilities is:

$$\epsilon_i = \tilde{\pi}_i - \pi_i^{(\text{opt})} = o_p(1).$$

For Poisson sampling, the inclusion indicator $\tilde{Z}_i$ (based on $\tilde{\pi}_i$) differs from Z_i (based on $\pi_i^{(\text{opt})}$) only when $U_i \in [\min(\pi_i^{(\text{opt})}, \tilde{\pi}_i), \max(\pi_i^{(\text{opt})}, \tilde{\pi}_i))$, where $U_i \sim \text{Uniform}(0,1)$. The probability of such an event is $O(|\epsilon_i|)$, which is negligible asymptotically.

Asymptotic equivalence of estimators The weighted empirical measure $\hat{\mathbb{P}}_n^{\tilde{\pi}}$ constructed from $\tilde{\pi}_i$ satisfies:

$$\sqrt{\frac{n}{c}}\left(\hat{\mathbb{P}}_n^{\tilde{\pi}} - P\right) \rightsquigarrow R^{Q_{\text{opt}}},$$

where $R^{Q_{\text{opt}}}$ is the limiting process under Q_{opt}. By the functional delta method [113], the plug-in estimator $\hat{\theta}$ inherits this convergence:

$$\sqrt{n_{\text{sub}}}\left(\hat{\theta}-\theta\right) \rightsquigarrow \mathcal{N}\left(0, V^{Q_{\text{opt}}}\right).$$

Conclusion Under the i.i.d. subsampling of S and consistency of the pilot estimator M, the subsampled estimator $\hat{\theta}$ achieves the minimal asymptotic variance $V^{Q_{\text{opt}}}$.

Insights and Experimental Results

The two theorems provide theoretical guarantees for the effectiveness of our method, particularly when using small proxy models for data selection. The first theorem establishes the optimality of the sampling distribution Q_{opt} and provides the foundation for efficient data selection in low-resource environments. The second theorem extends this by showing that, even when the influence function is estimated from a proxy model, the subsampling method remains asymptotically optimal, which is crucial for maintaining model performance when resources are constrained.

In our experiments, we observed that the proposed method, especially with MELD-DS for data selection, outperforms traditional methods in both computational efficiency and accuracy. The ability to use small proxy models for estimating the influence function reduces the need for large labeled datasets, enabling fast fine-tuning with limited resources. □