Can Foundation Models Wrangle Your Data?

Avanika Narayan, Ines Chami†, Laurel Orr, Christopher Ré
Stanford University and †Numbers Station
{avanika,lorr1,chrismre}@cs.stanford.edu,ines.chami@numbersstation.ai

ABSTRACT

Foundation Models (FMs) are models trained on large corpora of data that, at very large scale, can generalize to new tasks without any task-specific finetuning. As these models continue to grow in size, innovations continue to push the boundaries of what these models can do on language and image tasks. This paper aims to understand an underexplored area of FMs: classical data tasks like cleaning and integration. As a proof-of-concept, we cast three data cleaning and integration tasks as prompting tasks and evaluate the performance of FMs on these tasks. We find that large FMs generalize and achieve SoTA performance on data cleaning and integration tasks, even though they are not trained for these data tasks. We identify specific research challenges and opportunities that these models present, including challenges with private and temporal data, and opportunities to make data driven systems more accessible to non-experts. We make our code and experiments publicly available at: https://github.com/HazyResearch/fm_data_tasks.

1 INTRODUCTION

Foundation Models (FMs) [17] are models trained on broad data that can be adapted to a wide range of downstream tasks. These models have achieved substantial gains across many semantically challenging tasks such as question answering [18], knowledge base construction [66], and information retrieval [33]. As they have scaled to hundreds of billions of parameters (e.g. GPT-3 [18], PaLM [20]), large FMs have demonstrated surprising emergent behaviors and good zero-shot generalization to new tasks (i.e. no task-specific finetuning) on domains vastly different from the data they were pretrained on [20]. These large FMs are often autoregressive language models (e.g. GPT-3 and PaLM) that are trained to predict the next word in large text corpora and can be adapted to new tasks given a simple natural language description of the task (see Figure 1). These breakthrough capabilities have led to a race for building bigger and better models, and innovations continue to push the boundaries of what large FMs can do on a variety of hard *language tasks*.

A natural question that arises is whether these advances can benefit hard classical *data tasks* (e.g. data cleaning and integration). While it is clear that FMs benefit text-intensive tasks, it is not clear whether these models can be applied to data tasks over structured data. The symbols commonly found in structured data (e.g. dates, numbers, alphanumeric codes) are less frequent in natural language text so it is unclear that FMs possess the ability to reason over them. Moreover, since FMs are trained to predict the next word, it is non-obvious that they can work out-of-the-box on complex data tasks. This paper explores the aforementioned question and introduces a new research vision for leveraging FMs for data management, focusing on data cleaning and integration tasks—two keys steps in data-driven enterprise pipelines.

Recently, a large body of research has applied machine learning (ML) [41] and deep learning (DL) [46, 62] methods—namely pretrained language models (PLMs) like BERT [27]—to semantically-complex data tasks. However, these approaches still require a significant amount of engineering effort as they rely on:

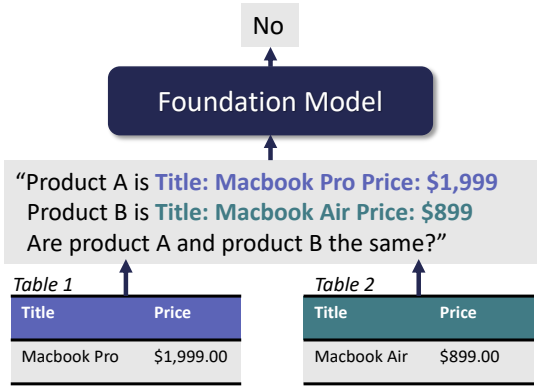

Figure 1: A large FM can address an entity matching task using prompting. Rows are serialized into text and passed to the FM with the question "Are products A and B the same?". The FM then generates a string "Yes" or "No" as the answer.

- **Task-specific architectures**: Data cleaning and integration encapsulate *many different tasks* such as entity matching [64], schema matching [78], and error detection [34]. Existing approaches, whether they are rule-, ML- or DL-based vary greatly from one task to the other, often with task-specific, complex architectures. For instance, adapting BERT to data tasks requires architectural changes and finetuning the entire model for each task. This leads to siloed and hard-to-maintain systems.
- **Hard-coded knowledge**: Data tasks often rely on *domain knowledge* (e.g. understanding the relationship between a city and its zip code for data cleaning constraints) and commonsense reasoning. These are usually hard-coded with human-engineered rules or external knowledge bases [22, 71]. Consequently, systems can be brittle and fail to generalize to a diverse set of domains.
- **Labeled data**: ML- and DL-based solutions require a lot of hand-labeled data [9]. For instance, PLMs that have achieved state-of-the-art (SoTA) results on data tasks (e.g. Ditto [32]) require a significant amount of task-specific labeled data and fine-tuning to achieve good performance. Labeling data for each task is engineering intensive and adds to the difficulty of maintaining data cleaning and integration systems.

Excitingly, FMs display several useful properties that make them an appealing choice compared to traditional approaches:

- **Task-agnostic architecture**: As a result of their natural language interface, FMs can be applied to a wide-range of tasks. For instance, Figure 1 shows how an entity matching task—which requires identifying whether two table entries refer to the same

entity—can be cast as a prompting task. This unifying interface eliminates the need for siloed architectures, in contrast to existing learned approaches where architectures need to be carefully crafted for each task (e.g. task-specific classification layer).
- **Encoded knowledge**: Because FMs are trained on large, generic corpora of data, they contain knowledge about a wide range of common entities, and thus do not rely on human-engineered rules to acquire knowledge [69].
- **Limited to no labeled data**: FMs can be applied to a wide-range of tasks with little to no labeled data (e.g. few-shot and zero-shot). When a FM needs to be fine-tuned, it often needs dramatically less labeled data to achieve competitive results [38].

Our goal is to better understand if large FMs can be applied to data integration and cleaning tasks. We study the behavior of GPT-3—an early and promising FM. While GPT-3 is already a high quality model, we expect the significant investment in FMs from both academia and industry to lead to more performant and scalable FMs over time. Like many other communities, the data management community stands to benefit from these trends. As such, we aim to understand the benefits and limitations of FMs on data tasks, by focusing on three key questions.

How well do large FMs transfer to data tasks? To answer this, we cast a variety of data tasks as natural language generation tasks (Section 3) and explore whether a single FM can generalize well to these tasks. In Section 4.2, we quantify the zero- and few-shot performance of FMs on three modern enterprise data tasks: entity matching, error detection, and data imputation. We find that the largest GPT-3 variant (175B parameters) outperforms SoTA ML-and DL-based approaches on these tasks with few examples. This is particularly surprising given that prior approaches are fully-finetuned on task-specific labeled data for these data tasks, while GPT-3-175B is only pretrained to generate text.

What are the caveats in applying FMs to data tasks? In Section 4.3, we unpack the few-shot "prompt tuning" process—serializing tabular data to text, casting data tasks as text generation tasks and constructing demonstrative task examples—for applying FMs to data tasks. We quantify the effects of prompt formatting variations on performance and the differences between manually and randomly selecting task examples. We find that FMs are brittle to differences in prompt formatting and that performance improves when prompts are manually selected versus randomly selected.

What opportunities do FMs present for data tasks and what are the relevant research challenges? Finally, in Section 5, we discuss the potential challenges and related research questions with using FMs in data management pipelines. We discuss the forthcoming shift in how ML systems are built, challenges around updating FM knowledge, and opportunities and considerations pertaining to private, temporal and local data.

We hope that our preliminary exploration will encourage the data management community to explore the effectiveness of FMs for other data tasks and develop techniques to overcome the shortcomings of FMs in this setting.

2 BACKGROUND

We first give some background on the different data tasks considered in this paper and then provide a brief review of FMs.

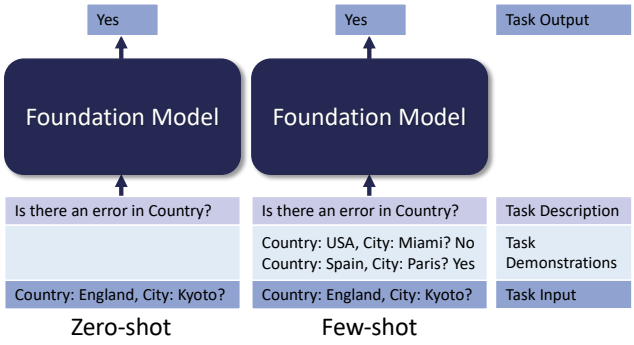

Figure 2: Different ways to use FMs with "in-context" learning [18] on an error detection task. For zero-shot (left), the prompt is the task description and the example to complete. For few-shot (right), the prompt adds demonstrations of how to complete the task.

2.1 Problem Setup

We focus on entity matching (EM), error detection (ED), and data imputation (DI) and describe the setup for these tasks. We denote D, a structured dataset with n entries, such that each entry is represented by a collection of m attribute value pairs; i.e., for entry $e_i \in D$ we have $e_i = \{e_{i,j}\}_{1 \le j \le m}$ where for attribute j, $e_{i,j} = \{\texttt{attr}_j, \texttt{val}_j\}$.

Entity Matching The goal of EM is to match entities (real-world objects like people, places and things) across different datasets. Formally, given two structured datasets (D, D') and pairs of entries $e, e' \in D \times D'$, the goal is to predict whether these entries represent the same entity or not. This problem is usually solved as a classification problem, and real-world EM systems are often preceded by blocking heuristics which are used to remove obvious non-matches.

EM has been extensively studied over the past decade (see [64] for a survey) and methods broadly fall into three categories: rule-based, crowd-based [31, 81] and ML/DL-based [41, 62]. Recently, methods relying on PLMs [46] have become SoTA for this task.

Error Detection ED is an important step in data cleaning pipelines. Given an entry e, the goal is to detect attributes j where $\texttt{val}_j$ has an error. The task is framed as a classification problem where the goal is to predict if $\texttt{val}_j$ is correct for a given e.

ED has been studied extensively in both academic and industrial settings [7]. There are a number successful of commercial offerings including Trifacta [6] and Tamr [5]. Traditionally, ED systems have been heavily reliant on rule-based algorithms which enforce data constraints through functional dependencies or knowledge bases [21, 22, 24]. Additionally, there are statistical-based approaches such as pattern enforcement [22, 37], outlier detection [26], and record deduplication [75] algorithms. Recent efforts have developed SoTA ML models for ED [34].

Data Imputation DI is a critical step for repairing dirty data sources. Given an entry e with missing attribute values $\{\texttt{attr}_j, \texttt{NULL}\}$, the goal of DI is to infer the missing values. The full range of plausible values for the missing value is not known apriori.

Existing literature in DI falls into three categories: traditional clustering and/or statistical-based [58], generative model-based [30], and ML/DL-based [15, 60] approaches. Existing methods struggle when needing to impute values not seen in the training set [60].

2.2 Background on Foundation Models

We now give an overview of language FMs, starting from very early, smaller FMs (i.e. PLMs) and moving to large-scale FMs—the latter of which is the focus of this paper.

Pretrained Language Models Pretrained Language Models (PLMs) are neural networks pretrained on large corpora of publicly available text (e.g. web pages). The first breed of PLMs—ELMo [65], BERT [28], RoBERTA [52], T5 [67]–learned the semantics of natural language by predicting masked works during pretraining. These models have on the order of hundreds of millions of parameters. Traditionally, PLMs are adapted to downstream tasks through a task-specific prediction layer (e.g. classification layer) and a task-specific finetuning step wherein all model weights are updated.

Large Autoregressive Language Models In 2020, GPT-3 [18] marked a significant shift in the the ML community. It represented a new class of large-scale language models: autoregressive language models pretrained to predict the next word in a sequence. These models are *billions* of parameters and have been used for language generation tasks such as question answering and summarization. Since the release of GPT-3, bigger and better performing autoregressive language models have been developed [20].

Emergent Behaviors Interestingly, the biggest GPT-3 variant (175B parameters) has the capacity to solve natural language tasks with only a few examples (called few-shot prompting), and in some cases, just a task description (e.g. "Translate French to English"). Unlike traditional finetuning, no model parameters are updated to fit the task. Few-shot prompting has proven to be effective on tasks widely different from the FMs pretraining objective. Some examples include code generation [84], Trivia QA [18, 48] and common sense reasoning tasks [18]. Smaller models (less than 10B parameters) typically require some form of task-specific finetuning to perform well. We explore ways to use smaller FMs on data tasks in Appendix A.

3 FOUNDATION MODELS FOR DATA TASKS

Our goal is to understand whether FMs can benefit data cleaning and integration tasks. The procedure for applying FMs to data tasks requires adapting structured data inputs to textual inputs (Section 3.1), casting data tasks as text generation tasks (Section 3.2) and, for few-shot prompting, constructing demonstrative task examples to help the FM learn new data tasks (Section 3.3).

3.1 Tabular Data Serialization

FMs take text as input and generate text as output. In order to apply these models to data tasks, we first need to convert structured tabular data inputs to text representations.

Given a structured dataset, we first convert an entry to text. Concretely, for entry e, we follow previous work [46] and serialize the attribute value and entry as follows:

$$\texttt{serialize}(e) \coloneqq \texttt{attr}_1 : \texttt{val}_1 \ldots \texttt{attr}_m : \texttt{val}_m$$

If the attribute value is NULL, we serialize it as the empty string. Based on the task and dataset, serialization only happens over a subset of attributes relevant to the task. FMs can be sensitive to the prompt format and the specific serialization used [87]. In Section 4.3 we study the sensitivity of FMs to different sub-selection choices.

Dataset	Magellan	Ditto	GPT3-175B (k=0)	GPT3-175B (k=10)
Fodors-Zagats	100	100	87.2	**100**
Beer	78.8	94.37	78.6	**100**
iTunes-Amazon	91.2	97.06	65.9	**98.2**
Walmart-Amazon	71.9	86.76	60.6	**87.0**
DBLP-ACM	98.4	**98.99**	93.5	96.6
DBLP-Google	92.3	**95.60**	64.6	83.8
Amazon-Google	49.1	**75.58**	54.3	63.5

Table 1: Entity matching results measured by F1 score where k is the number of task demonstrations.

3.2 Data Tasks as Natural Language Tasks

Next, we need to convert data tasks to text generation tasks. We construct natural language descriptions of each task (i.e. prompts) that use the serialized representations from Section 3.1. The prompts are passed to the FM, whose generated output is the answer to the given task. For entity matching and error detection, the model generates a "Yes" or "No" response,[1] and for imputation it generates the missing value. We now enumerate the prompts for each task.

Entity matching: given two entries (e, e') the template is

```
Product A is serialize(e). Product B is serialize(e').
Are Product A and Product B the same?
```

Data imputation: given an entry e and attribute j to infer, we use the template

$$\texttt{attr}_1 : \texttt{val}_1 \ldots \texttt{attr}_j \ ?$$

Error detection: given an entry e and attribute j to classify as erroneous, we use

$$\texttt{Is there an error in attr}_j : \texttt{val}_j ?$$

These templates highlight the generality of this framework which could be extended beyond the tasks considered in this paper.

3.3 Task Demonstrations

Task demonstrations can be included in the prompt to help the model learn a new task (see Figure 2 for an error detection example). These demonstrations are used to show the model how the task should be completed (e.g. should it generate Yes/No or a missing value) as well as understand the finer-grained semantics of D. We explore two approaches for selecting task demonstration examples.

Random One approach to is to sample random examples from a labeled dataset. However, this approach often causes high variance

[1] Interestingly, the model is not constrained to produce a Yes/No answer on the output side, but we find that this happens most of the time. In the rare examples where the model does not predict a Yes/No answer, we use No as the default answer.

in FM performance [50]. Moreover, the ordering of examples in the prompt can have a non-trivial impact on the downstream performance [50, 54, 87]. Although recent work has tried to systematize the prompt tuning process, it is still an open area of research [45, 76].

Manual Another approach is to manually construct examples that yield good performance on a held-out validation set. This approach is more costly—requiring more time—in comparison to random sampling but improves performance when examples are carefully constructed.

In our manual prompt tuning experiments, we spend *at most 1 hour* per task analyzing errors on the validation set. We then manually construct demonstrations that help the model correct these errors. We liken this step to the canonical hyperparameter tuning process in ML where time and compute are spent finding the best model parameters for the given task. However, the engineering effort for the prompt tuning process is significantly less. Concretely, it takes less time (minutes vs. hours and days), is more compute effective (inference vs. full training) and gives the user finer-grained control (natural language guidance vs. blackbox parameter updates).

4 EXPERIMENTS

We compare FMs to SoTA methods on a variety of data cleaning and integration tasks. Our goal is to understand whether large FMs can transfer to data tasks in zero- and few-shot settings (Section 4.2), and the nuances in applying FMs to data tasks (Section 4.3).

4.1 Experimental Setup

We begin by describing our experimental protocol, including models, datasets, metrics and baselines used.

Models For few-shot prompting we use the GPT-3-175B parameter model (text-davinci-002) in the OpenAI API endpoint [63]. We access these models by passing our input prompts to the endpoint for a per-sample fee.

Datasets For entity matching, we use the standard Magellan benchmark [41]. For imputation, we choose two challenging datasets from [60]: Restaurants and Buy. Finally, for error detection, we evaluate on the benchmark Hospital dataset that is used across several data cleaning papers [21, 34, 71].

We use the provided train/test/dev splits for all entity matching datasets. For cleaning tasks, the splits are not available but we follow the protocol of [34, 60] to generate the dataset splits.

Evaluation Metrics For both the error detection and entity matching datasets, we evaluate performance using F1 score. For imputation, we evaluate performance using accuracy.

Baselines We compare against the SoTA methods for each task. For entity matching, we benchmark against Ditto [46], the current SoTA DL-based approach which finetunes BERT [28]. For data imputation, we benchmark against IMP [60], which finetunes RoBERTa [52] and HoloClean [71], a statistical-based SoTA data repairing engine. Finally, for error detection, we compare against HoloClean and HoloDetect [34], a data-augmentation based ML approach.

Task	Imputation		Error Detection
Dataset	Restaurant	Buy	Hospital
HoloClean	33.1	16.2	51.4
IMP	77.2	96.5	-
HoloDetect	-	-	94.4
GPT3-175B (k=0)	70.9	84.6	6.9
GPT3-175B (k=10)	**88.4**	**98.5**	**97.8**

Table 2: Data cleaning results, measured in accuracy for data imputation and F1 score for error detection where k is the number of task demonstrations.

4.2 Zero/Few-shot Performance of Large FMs

In this section we explore the zero and few-shot performance of GPT-3-175B. Our goal is to understand whether large FMs transfer to data tasks in zero- and few-shot settings.

4.2.1 Results. We first review the zero- and few-shot results.

Few-shot Performance Our results show that in the few-shot setting, with manually curated task demonstrations, *GPT-3-175B achieves SoTA performance on 4 entity matching, 2 imputation and 1 error detection benchmark dataset(s)* (see Table 1). The FM outperforms fully-finetuned, SoTA PLM-based approaches for entity matching [46] and data imputation [60]. For error detection, the few-shot approach outperforms the current ML-based SoTA method, HoloDetect, which uses both data augmentation and weak supervision to perform optimally on the task (see Table 2).

Zero-shot Performance In the zero-shot setting, we observe that the FM outperforms statistical-based approaches and standard data repair engines [71] for imputation. On entity matching, the zero-shot performance is significantly lower than the few shot performance. This performance gap suggests that demonstrations are very important for the task, and we study the impact of task demonstrations in more detail in Section 4.3.

4.2.2 Discussion. The above results show that large FMs can transfer to data tasks. These results are particularly exciting given that FMs are trained to model English language and have no prior exposure to the semantics of data tasks nor the syntax of tabular data. Furthermore, the zero-shot performance on imputation suggests that large FMs not only have an understanding of how to complete the tasks, but also have encoded *knowledge* that is necessary to correct and complete records (e.g. functional dependencies between address and zip code). We provide an analysis of encoded large FM knowledge in Appendix B.

On the entity matching datasets that the FM does not achieve SoTA on, error analysis reveals that the FM struggles on data domains that contain jargon not commonly found in text. In such cases, the model lacks a strong semantic understanding of the input and has difficulty reasoning over the data. For example, in the Amazon-Google dataset, the model has difficulty matching samples due to the high volume of product-specific identifiers in the descriptions. Here, for instance, the model fails to accurately match the two entries: *"name: pcanywhere 11.0 host only cd-rom xp 98 nt w2k me. manufacturer: symantec. price: NULL"* and *"name: symantec pcanywhere 11.0 windows. manufacturer: NULL. price: 19.99."*. We

Prompt Format	Beer	iTunes-Amazon	Walmart-Amazon
Prompt 1 (w. Attr. & Example Select.)	**100** ± 0.00	**98.2** ± 0.00	88.9 ± 0.00
Prompt 1 (w/o Example Select.)	91.1 ± 0.05	86.6 ± 0.02	65.2 ± 0.04
Prompt 1 (w/o Attr. Select.)	76.9 ± 0.00	94.1 ± 0.00	75.0 ± 0.00
Prompt 2 (w. Attr. & Example Select.)	96.3 ± 0.00	84.7 ± 0.00	**100** ± 0.00
Prompt 1: "`Are Product A and Product B the same?`"			
Prompt 2: "`Are Product A and Product B equivalent?`"			

Table 3: Entity matching results (F1 score) for different prompt formats (k=10). For all datasets, we evaluate on up to 200 samples for cost purposes.

discuss ways to adapt FMs to domain-specific data in more detail in Section 5.

4.3 Prompt Tuning Ablations

In this section, we analyze the performance impact of the three different choices made during prompt tuning: attribute selection, prompt formatting, and task demonstration curation.

4.3.1 Results. We run our ablations on three entity matching datasets (see Table 3). For all datasets, we evaluate on up to 200 samples for cost purposes. We discuss our findings next.

Attribute Selection First, we find through experimentation that sub-selecting attributes during row serialization can have a non-trivial impact on entity matching performance. In particular, we observe that sub-selecting attributes that are essential in determining whether two entities match (e.g. name) and removing noisy attributes improves model accuracy. To better illustrate this point, we evaluate FM performance on three datasets when not sub-selecting attributes. We find that, on average, attribute selection results in a 13.7 F1 point performance improvement (Table 3).

Prompt Formatting Second, we observe that FMs can be brittle to subtle variations in prompt templates. We investigate this brittleness by replacing the span "`Are Product A and Product B the same?`" (Prompt 1) with "`Are Product A and Product B equivalent?`" (Prompt 2). This minor modification results in an average of 9.4 F1 point performance variance across the three datasets in Table 3.

Task Demonstrations Finally, we find that the choice of task demonstrations has a significant impact on downstream performance. We conduct an ablation where we replace manually curated task demonstrations with randomly selected demonstrations (see Prompt 1 (w/o Example Select.) in Table 3). We run this experiment over three different random seeds and report the results in Table 3. Across all cases, manually curated examples outperform randomly selected examples by an average of 14.7 F1 points.

4.3.2 Discussion. The aforementioned results demonstrate that successful prompt tuning requires (1) selecting an informative set of attributes required for the task, (2) crafting a well-formatted prompt that the FM understands, and (3) constructing a set of instructive task demonstrations that condition the model to the data at hand. For (1), we find that attribute sub-selection boosts performance by removing noisy attributes that hurt performance. For (2), our ablations show that prompt formatting (e.g. word choice, punctuation) can have significant impact on model performance. For (3), our results indicate that examples need to be carefully crafted for FMs to learn new tasks. We conjecture that on more reasoning-intensive tasks (e.g. matching), prompts are important as they help teach the model how to reason about entities and how to complete the task. Moreover, we emphasize that all three steps require some form of iteration to develop the most effective prompt (e.g. passing various inputs to the model and inspecting its outputs).

These findings are aligned with existing literature on prompt tuning that observe non-trivial variance in prompt-based learning settings [87]. This performance variance suggests that iterative prompt programming is an essential human-in-the-loop process for FM usage [51]. However, some works suggest that smarter, automatic example selection methods can help close the gap between random example selection and fully human-in-the-loop prompt engineering [50]. These results highlight the paradigm shift induced by building systems centered around FMs: instead of spending time tuning models, we now need to invest time finding the right examples and engineering useful prompts for each task. We discuss these paradigm shifts in more detail in Section 5.

5 RESEARCH AGENDA

Because of their natural language interface and vast internal knowledge, FMs provide an interface for unifying a wide-range of siloed, hand-engineered data integration pipelines. Consequently, *we envision that the data orchestration workbenches of the future will be centered around FMs.* For this vision to be realized, users need to be able to (1) iteratively interact with the model and provide the model with feedback (interface), (2) update the model with local information (local context), and (3) prompt the model through multi-step tasks (compositional reasoning). We discuss these directions in Section 5.1, Section 5.2 and Section 5.3 respectively.

5.1 Natural Language Interface

FMs are changing the way that data-centric systems are built. They usher in a new era of human-machine collaborative workflows wherein users spend less time finetuning and training models and more time constructing natural language prompts that are representative of the task at hand.

User Interaction Our work illustrates that the way in which data driven systems are built is fundamentally changing. As FMs become the central piece of our ML systems, engineering time is no longer spent on labeling data or on expensive hyperparameter tuning but instead on prompt tuning and constructing demonstrative examples that teach FMs a given task. Put simply, we are transitioning to a world of "worker and AI collaboration" [49] in which all communication is done through natural language. Because this interaction loop is conducted in language, this democratizes who the key stakeholders will be in model development and deployment. The burden of development shifts from data science and machine learning experts who have limited domain expertise but strong technical skills, to domain experts who have limited technical expertise but can use natural language to interact with and develop these models. While existing work has taken a first step in this direction by exploring the concept of prompt engineering and prompt tuning [72], there is less work in understanding the user interface disruption that will be

catalyzed by widespread adoption of FMs. In future work we seek to better understand the human-in-the-loop prompt engineering pipeline, especially in the context of data management practices.

5.2 Local Context

As we look to apply FMs in variety of data integration contexts, challenges arise when the models need to reason over private data or new streams of data. Additionally, there are huge economic opportunities in utilizing FMs to ingest the astonishing amounts of data exhaust accumulated by organizations.

Private Data With the proliferation of important data privacy regulations and legal frameworks [3, 79], the separation between private data held by individuals and organizations and publicly trained models becomes increasingly well-defined. As a result, there are three key challenges in applying FMs to private data: FM selection, limited labeled training data, and user feedback. There is an ever-expanding set of off-the-shelf FMs, and the failure modes incurred on the data tasks depend on the chosen FM. We need methods to match models to user tasks, without sacrificing the privacy of the user's personal data. Individuals also typically lack labeled private data, raising a challenge for generating data for lightweight fine-tuning. Further, when users and organizations identify model failures, we need methods to incorporate feedback either into the FM (e.g. via prompt design), without sacrificing privacy. Finally, beyond operating over data in a *single* privacy scope, an underexplored setting is how to reason over both public and private knowledge simultaneously. While existing research benchmarks are designed over one privacy scope, Arora et al. [12] recently released the first benchmark for multiple privacy scopes. Excitingly, Arora and Ré [13] identify that the powerful adaptation capabilities of FMs enable the use of strong *access control* based privacy frameworks for reasoning over private data.

Organizational Digital Exhaust In the enterprise setting, along with the data that is intentionally collected, organizations accumulate a staggering amount of data exhaust—the informational byproduct that constantly streams from devices, products, and workforce management practices [1]. This data is multimodal and rich with signal. It can drive customer insights, business model innovation and workflow automation [1]. Because FMs are pretrained in an unsupervised fashion with a simple token prediction objective (see Section 2.2), they have the capacity to learn over *any* raw and unlabeled sources data [29, 70, 85]. As such, FMs can effectively ingest the exhaust from all levels the data stack (from system logs to structured data). We will need methods to facilitate FM exhaust ingestion and, more importantly, research to explore new opportunities and applications enabled by FMs customized to enterprise data.

Domain-Specific Data There is a growing number of specialized settings (e.g. medical, legal, individual enterprises) in which FMs have an opportunity to play a critical role. These domains are data rich, generating millions to trillions of data artifacts yearly [1, 10]. In the legal domain, there are hundreds of thousands of new cases filed in federal court yearly [2], the documents of which can be used to train a specialized FM. Prior work demonstrates that the performance of FMs degrades on out-of-domain tasks [32], so learning from these large databanks is a necessary and tractable step given the volume of readily available data. Because we can train FMs on domain specific data, we can use them to solve similar data integration and orchestration tasks in the specialized domain thereby creating a feedback loop where FMs are helping create the very data they are trained on. Understanding the tradeoffs around training a specialized FM versus utilizing an existing off-the-shelf FM is a critical research question. Beyond academic efforts, AI21 Labs recently released MRKL—a system for augmenting FMs with external knowledge sources and symbolic reasoning experts—demonstrating how FMs can be utilized to ingest, reason and operate over varied, domain-specific data stores [39].

Temporal Data In industry, data is continuously gathered, resulting in large stores of timeseries data. A key data analytics challenge is ingesting and reasoning over these data streams. Because FMs are pretrained on a static corpus of text, they perceive the world as a snapshot of the web. As such, we need methods for managing incremental model updates based upon new streams of data. It is still an open question as to the best ways to update models such that they incorporate new information while also maintaining prior knowledge of unchanged facts. Recent work on lifelong pretraining has identified ways of continually updating FMs in order to adapt them to emerging data [36, 77]. Excitingly, new methods have demonstrated that it is possible to selectively update knowledge in FMs, further demonstrating that making targeted changes to model knowledge is feasible [61].

5.3 Compositional Reasoning

The reasoning abilities of FMs can allow them to go beyond simple tasks—potentially even enabling them to automate entire data integration workflows.

Code Generation Data integration is a costly practice that relies on significant manual effort. It is projected that manual integration tasks will be reduced by 45% as a result of automated service-level tools [86]. Recently, Codex—a FM trained to translate language to code—demonstrated the ability to generate functional code given a high-level description of the desired code snippet [19]. Moreover, FMs have demonstrated the ability to generate coherent chain of thought reasoning [83]. Because of their ability to generate operational code and reason compositionally, it is fitting to question whether FMs, guided by the natural language feedback of IT specialists, can act as drop-in replacements for data integration tools.

6 CONCLUSION

In this work we investigate the applicability of FMs to classical data tasks. We find that large FMs can achieve SoTA performance on many data tasks with 0-to-few natural language task demonstrations. *The ability of these models to transfer to data tasks with no task-specific finetuning is particularly interesting given that these models are simply trained to predict next words.* Our work builds upon years of important work on integration and cleaning tasks in the data management community. We hope that our results gesture towards the possibilities of using language guided models for human-in-the-loop data integration practices across a broader range of data management tasks.

ACKNOWLEDGMENTS

We are thankful to Ihab Ilyas, Theo Rekatsinas, Mike Cafarella, Ce Zhang, Sen Wu, Christopher Aberger, Neel Guha, Simran Arora, Beidi Chen and Xiao Ling for their helpful discussions and feedback. We gratefully acknowledge the support of DARPA under Nos. FA86501827865 (SDH) and FA86501827882 (ASED); NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Swiss Re, Brown Institute for Media Innovation, Department of Defense (DoD) through the National Defense Science and Engineering Graduate Fellowship (NDSEG) Program, Fannie and John Hertz Foundation, National Science Foundation Graduate Research Fellowship Program, Texas Instruments Stanford Graduate Fellowship in Science and Engineering, and members of the Stanford DAWN project: Teradata, Facebook, Google, Ant Financial, NEC, VMWare, and Infosys. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

REFERENCES

[1] 2018. How to turn data exhaust into a competitive edge. https://knowledge.wharton.upenn.edu/article/turn-iot-data-exhaust-next-competitive-advantage/

[2] 2020. Federal Judicial Caseload Statistics 2020. https://www.uscourts.gov/statistics-reports/federal-judicial-caseload-statistics-2020

[3] 2022. California Consumer Privacy Act (CCPA). https://oag.ca.gov/privacy/ccpa

[4] 2022. Decreasing cost of storage. https://www.iotone.com/term/decreasing-cost-of-storage/t172

[5] 2022. Tamr | Enterprise Data Mastering at Scale - Tamr Inc. https://www.tamr.com/

[6] 2022. Trifacta: Data Wrangling Software and Tools. https://www.trifacta.com/

[7] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting data errors: Where are we and what needs to be done? *Proceedings of the VLDB Endowment* 9, 12 (2016), 993–1004.

[8] Abubakar Abid, Maheen Farooqi, and James Zou. 2021. Large language models associate Muslims with violence. *Nature Machine Intelligence* 3, 6 (2021), 461–463.

[9] Amina Adadi. 2021. A survey on data-efficient algorithms in big data era. *Journal of Big Data* 8, 1 (2021), 1–54.

[10] Julia Adler-Milstein, Jason S Adelman, Ming Tai-Seale, Vimla L Patel, and Chris Dymek. 2020. EHR audit logs: a new goldmine for health services research? *Journal of biomedical informatics* 101 (2020), 103343.

[11] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. *arXiv preprint arXiv:2204.14198* (2022).

[12] Simran Arora, Patrick Lewis, Angela Fan, Jacob Kahn, and Christopher Ré. 2022. Reasoning over Public and Private Data in Retrieval-Based Systems. arXiv:2203.11027 [cs.IR]

[13] Simran Arora and Christopher Ré. 2022. Can Foundation Models Help Us Achieve Perfect Secrecy? (2022). Under Review.

[14] Felix Biessmann, Tammo Rukat, Phillipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing Value Imputation for Tables. *Journal of Machine Learning Research* 20, 175 (2019), 1–6. http://jmlr.org/papers/v20/18-753.html

[15] Felix Biessmann, Tammo Rukat, Philipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing Value Imputation for Tables. *J. Mach. Learn. Res.* 20, 175 (2019), 1–6.

[16] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.* https://doi.org/10.5281/zenodo.5297715

[17] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[19] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[20] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

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

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

[23] EasyClosets Coupons. 2021. 21 best GPT-3 tools, examples and use cases - nogood™: Growth Marketing Agency. https://nogood.io/2021/06/25/gpt-3-tools/

[24] Michele Dallachiesa, Amr Ebaid, Ahmed Eldawy, Ahmed Elmagarmid, Ihab F Ilyas, Mourad Ouzzani, and Nan Tang. 2013. NADEEF: a commodity data cleaning system. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 541–552.

[25] Nilesh Dalvi, Vibhor Rastogi, Anirban Dasgupta, Anish Das Sarma, and Tamás Sarlós. 2013. Optimal hashing schemes for entity matching. In *Proceedings of the 22nd international conference on world wide web*. 295–306.

[26] Tamraparni Dasu and Ji Meng Loh. 2012. Statistical Distortion: Consequences of Data Cleaning. *Proceedings of the VLDB Endowment* 5, 11 (2012).

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

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[29] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A Generative Model for Music. *CoRR* abs/2005.00341 (2020). https://arxiv.org/abs/2005.00341

[30] Eric Ghysels, Arthur Sinko, and Rossen Valkanov. 2007. MIDAS regressions: Further results and new directions. *Econometric reviews* 26, 1 (2007), 53–90.

[31] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. 2014. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 601–612.

[32] Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 8342–8360.

[33] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909* (2020).

[34] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. Holodetect: Few-shot learning for error detection. In *Proceedings of the 2019 International Conference on Management of Data*. 829–846.

[35] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. https://doi.org/10.48550/ARXIV.1902.00751

[36] Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2021. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534* (2021).

[37] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the sigchi conference on human factors in computing systems*. 3363–3372.

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

[39] Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, Kevin Leyton-Brown, et al. 2022. MRKL Systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv preprint arXiv:2205.00445* (2022).

[40] Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving Forgetting Prevention and Knowledge Transfer in Continual Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[41] Pradap Konda, Sanjib Das, AnHai Doan, Adel Ardalan, Jeffrey R Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, et al. 2016. Magellan: toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1581–1584.

[42] AI21 Labs. 2022. STANDING ON THE SHOULDERS OF GIANT FROZEN LANGUAGE MODELS. *preprint* (2022).

[43] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*. https://openreview.net/forum?id=H1eA7AEtvS

[44] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 3045–3059. https://doi.org/10.18653/v1/2021.emnlp-main.243

[45] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 4582–4597.

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

[47] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*. PMLR, 6565–6576.

[48] Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. TruthfulQA: Measuring How Models Mimic Human Falsehoods. (2021).

[49] Alisa Liu, Swabha Swayamdipta, Noah A Smith, and Yejin Choi. 2022. WANLI: Worker and AI Collaboration for Natural Language Inference Dataset Creation. *arXiv preprint arXiv:2201.05955* (2022).

[50] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[51] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586* (2021).

[52] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[53] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[54] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786* (2021).

[55] Li Lucy and David Bamman. 2021. Gender and Representation Bias in GPT-3 Generated Stories. *NAACL HLT 2021* (2021), 48.

[56] Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Marzieh Saeidi, Lambert Mathias, Veselin Stoyanov, and Majid Yazdani. 2022. PERFECT: Prompt-free and Efficient Language Model Fine-Tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

[57] Bernard Marr. 2021. What is unstructured data and why is it so important to businesses? https://www.forbes.com/sites/bernardmarr/2019/10/16/what-is-unstructured-data-and-why-is-it-so-important-to-businesses-an-easy-explanation-for-anyone/?sh=999b04d15f64

[58] Chris Mayfield, Jennifer Neville, and Sunil Prabhakar. 2010. ERACER: a database approach for statistical inference and data cleaning. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 75–86.

[59] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On Faithfulness and Factuality in Abstractive Summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. 1906–1919.

[60] Yinan Mei, Shaoxu Song, Chenguang Fang, Haifeng Yang, Jingyun Fang, and Jiang Long. 2021. Capturing Semantics for Imputation with Pre-trained Language Models. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 61–72.

[61] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast Model Editing at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=0DcZxeWfOPt

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

[63] OpenAI. 2021. OpenAI API. https://openai.com/api/

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

[65] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2227–2237. https://doi.org/10.18653/v1/N18-1202

[66] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language Models as Knowledge Bases?. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2463–2473.

[67] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21 (2020), 1–67.

[68] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3505–3506.

[69] Simon Razniewski, Andrew Yates, Nora Kassner, and Gerhard Weikum. 2021. Language Models As or For Knowledge Bases. *arXiv preprint arXiv:2110.04888* (2021).

[70] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A Generalist Agent. *arXiv preprint arXiv:2205.06175* (2022).

[71] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. HoloClean: Holistic Data Repairs with Probabilistic Inference. *Proceedings of the VLDB Endowment* 10, 11 (2017).

[72] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.

[73] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*. https://openreview.net/forum?id=9Vrb9D0WI4

[74] Timo Schick and Hinrich Schütze. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2339–2352.

[75] Michael Stonebraker, Daniel Bruckner, Ihab F Ilyas, George Beskales, Mitch Cherniack, Stanley B Zdonik, Alexander Pagan, and Shan Xu. 2013. Data Curation at Scale: The Data Tamer System.. In *Cidr*, Vol. 2013. Citeseer.

[76] Fan-Keng Sun and Cheng-I Lai. 2020. Conditioned natural language generation using only unconditioned language model: An exploration. *arXiv preprint arXiv:2011.07347* (2020).

[77] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 8968–8975.

[78] Edhy Sutanta, Retantyo Wardoyo, Khabib Mustofa, and Edi Winarko. 2016. Survey: Models and Prototypes of Schema Matching. *International Journal of Electrical & Computer Engineering (2088-8708)* 6, 3 (2016).

[79] P. Voigt and A. Von dem Bussche. 2017. The EU General Data Protection Regulation (GDPR). In *Springer International Publishing*. https://doi.org/10.1007/978-3-319-57959-7

[80] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

[81] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. *Proceedings of the VLDB Endowment* 5, 11 (2012).

[82] Albert Webson and Ellie Pavlick. 2021. Do Prompt-Based Models Really Understand the Meaning of their Prompts? *arXiv preprint arXiv:2109.01247* (2021).

[83] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903* (2022).

[84] Frank F Xu, Uri Alon, Graham Neubig, and Vincent Josua Hellendoorn. 2022. A Systematic Evaluation of Large Language Models of Code. In *Deep Learning for Code Workshop*.

[85] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157* (2021).
[86] Ehtisham Zaidi and Sharat Menon. 2022. Magic Quadrant for Data Integration Tools.
[87] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*. PMLR, 12697–12706.

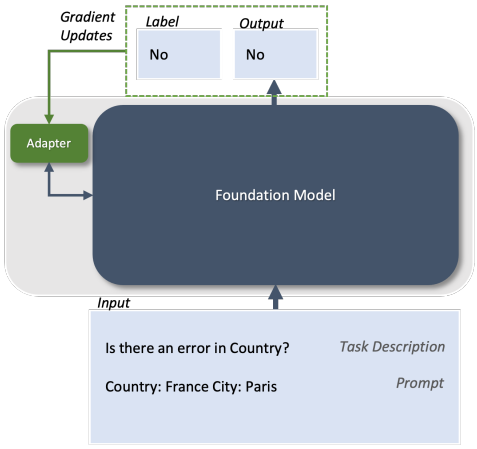

Figure 3: Adapter architecture

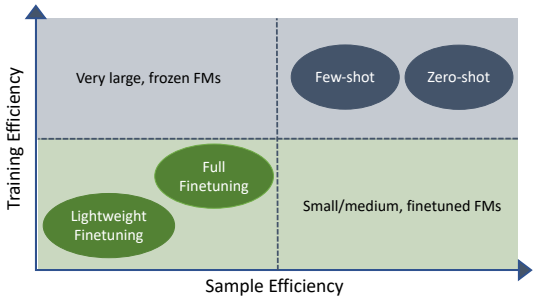

Figure 4: Sample-training efficiency tradeoffs. Larger FMs can be directly used in a zero/few-shot fashion. To achieve similar performance, smaller FMs need additional finetuning.

Appendix Overview We structure our appendix into three sections. In Appendix A, we demonstrate how to bridge the performance gap between small GPT-3 variants (e.g. 1.3B and 6.7B parameters) and the largest GPT-3 model (175B parameters) through an in-depth discussion of full and lightweight finetuning experiments on data tasks. In Appendix B, we explore the knowledge encoded in FMs and the effect of different model update strategies on this knowledge. Finally, in Appendix C, we discuss the role of FM bias on data tasks.

A SMALL FM FINETUNING EXPERIMENTS

In this section, we explore ways to bridge the performance gap between the smaller and larger FMs, focusing on (1) full finetuning (updating all parameters) and (2) lightweight finetuning (updating a small number of parameters). In Appendix A.1, we evaluate the two approaches and unpack their sample (as measured by number of labeled samples) and training (as measured by number of parameter updates) efficiency tradeoffs. We find that both lightweight and full finetuning can be used to reduce the performance gap between a 6.7B and a 175B parameter model to an average of 3.1 points. The lightweight finetuning requires more data than the fully finetuned approach, but updates only 5% of the model parameters.

Full/Lightweight Finetuning The typical practice for achieving optimal task performance in LLMs is to finetune pretrained models with task-specific data. This approach is usually training inefficient as all weights in the model are updated. Interestingly, FMs have been shown to achieve optimal task performance by simply training a lightweight, non-linear layer (i.e. adaptor) (Figure 3) on the outputs of a frozen model [42, 44]. This is a training efficient method that has proven to be competitive with fully-finetuned approaches [42]. However, because this approach trains a layer from scratch, it is usually less sample efficient than the fully finetuned approach. Figure 4 visualizes the differences in sample and compute efficiency between the two approaches.

For lightweight finetuning, we recursively join a single frozen FM with a small trainable network (the adapter) [42] (Figure 3). The prompt input is first passed to the frozen FM whose output embeddings are transformed by the adapter and then passed again as input embeddings to the frozen FM [42]. The generated output of the second pass yields the final answer of the model. In this setup, updates are only made to the adapter, i.e. no updates are made to the FM weights alleviating the need for costly training.

For both the full and lightweight finetuning, the model is fed the natural language prompt in Section 3.2 as input, and trained to generate the task output (e.g., a Yes/No string, or missing value).

A.1 Finetuned Performance of Small FMs

In this section we explore the fully finetuned and lightweight finetuned performance of two small FMs (namely GPT-3-6.7B and GPT-3-1.3B). Our goal is to understand whether finetuning can bridge the performance gap with larger models, and analyze the tradeoffs between sample efficiency and training efficiency.

A.2 Experimental Setup

For adapters, we use the GPT-Neo 1.3B [16] and the 6.7B parameter Neo GPT-J [80] models available on HuggingFace. All finetuning experiments are run on 4-8 A100 GPU machines using Deepspeed [68] with ZeRO-2 optimizations. We use the AdamW optimizer [53] with a 10% learning rate warmup followed by a linear decay. We train for a maximum or 30 epochs with a learning rate of 1e-4 for adapters and 2e-5 for full finetuning, and save the best model based on validation metrics.

A.3 Experimental Results

We first review the finetuning results on Walmart-Amazon (EM), Hospital (ED) and Restaurant (DI).

Full Finetuning Our results show that in the full finetuning setting, we can effectively reduce the performance gap between GPT-3-6.7B and GPT-3-175B on all datasets (Figure 5), using as little as 10% of the training set for Walmart-Amazon. GPT-3-1.3B also matches GPT-3-175B performance on Walmart-Amazon and Restaurant, and is within 8 points of the GPT-3-175B on Hospital. Compared to the 6.7B model, the 1.3B model is less sample-efficient and needs more example to bridge the performance gap.

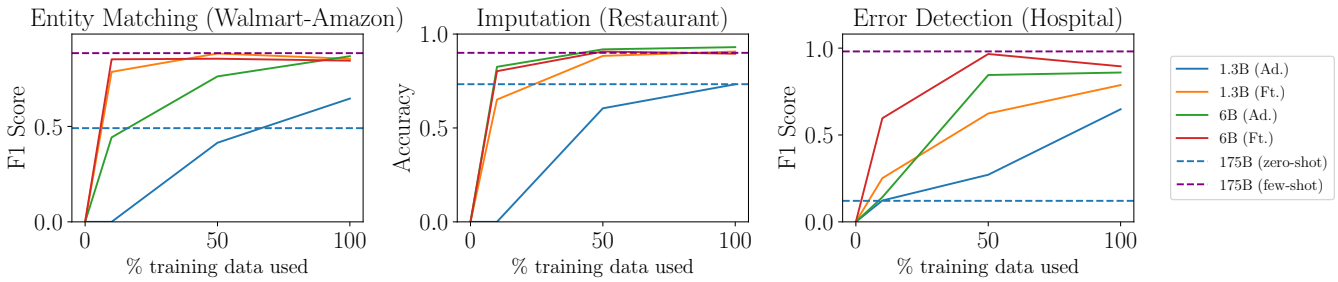

Figure 5: Finetuning experiments: Smaller FMs can be finetuned to bridge the performance gap with larger FMs. Full-finetuing bridges this gap with less data than adapters, but adapters are significantly less expensive to train.

Lightweight Finetuning In the lightweight adapter setting, we find that we can effectively bridge the performance gap between GPT-3-6.7B and GPT-3-175B on both Restaurant and Walmart-Amazon, but not Hospital. For GPT-3-1.3B, we are less effective in bridging the gap, and see an average 25 point performance difference between the GPT-3-175B across the three datasets. Furthermore, we find that the adapter model outperforms the fully-finetuned approach on two datasets with up to 4 points.

A.3.1 Discussion. These results show the feasibility of reducing the performance gap between GPT-3-175B and the smaller models. We observe some clear sample-training efficiency tradeoffs between the fully and lightly finetuned approaches, which we discuss next.

Sample Efficiency We find that fully finetuned models are usually more sample efficient than adapters, which could be explained by the need to train a layer from scratch in adapters. We also observe that sample efficiency improves with size as the performance of the GPT-3-6.7B trained with 10% of the data is equal to, or better than, the performance of the GPT-3-1.3B trained with 50% of the data in both the full and lightweight finetuning settings.

Finally, we comment on the performance difference between 6.7B adapter model and the 6.7B finetuned model on Hospital. We postulate that adapter model performed poorly on this dataset because the training set was particularly small (100 samples) which suggests that the adapter set-up requires more samples to learn generalizable patterns.

Training Efficiency We find that the adapter approach for GPT-3-6.7B *outperforms* the full finetuning approach on 2 of 3 datasets with 5% of the amount of trainable parameters, demonstrating that training efficiency need not be sacrificed for quality at this scale. However, at the smaller scale (GPT-3-1.3B), there is a non-trivial tradeoff between performance and training efficiency suggesting that training efficiency decreases as model size decreases.

B KNOWLEDGE ABLATIONS

In this section we seek to explore the encoded FM knowledge that is useful for data tasks. We begin with a qualitative analysis (Appendix B.1) and follow with a slice-based analysis that unpacks the impact of different training mechanisms on FM knowledge (Appendix B.2).

GPT-3 Model (% data)	freq = 0	0 < freq ≤ 10	freq > 10
175B (few-shot)	**100**	0.0	93.7
6.7B (adaptor, 100%)	0.0	**50.0**	**98.7**
6.7B (adaptor, 50%)	0.0	25.0	98.7
6.7B (adaptor, 10%)	0.0	0.0	87.3
6.7B (finetune, 100%)	0.0	25.0	96.2
6.7B (finetune, 50%)	0.0	0.0	98.7
6.7B (finetune, 10%)	0.0	0.0	89.9

Table 4: Impoverished city entity slice analysis: with lightweight finetuning, smaller model variants perform better than GPT-3-175B on infrequently occurring entities.

Input Prompt	GPT3-175B	GPT3-6.7B	GPT3-1.3B
"Address: 1720 university blvd State: AL ZipCode?"	32533	35205	35901
"Address: 26025 pacific coast hwy Phone number: 310/456-5733 City?"	Malibu	Torrance	Pacific Beach
'"Address: 804 north point st Phone number: 415-775-7036 City?""	San Francisco	San Francisco	San Francisco

Table 5: FMs have an inherent understanding of conditional functional dependencies that are traditionally hard-coded in error detection systems. GPT3-175B is able to accurately map addresses to zip codes and cities.

B.1 Qualitative Examples

To better illustrate the notion of *encoded knowledge* that large FMs possess, we conduct a qualitative analysis where we inspect the model predictions for the task of inferring missing zipcodes or cities given some context (Table 5) . GPT-3-175B is able to effectively apply its understanding of functional dependencies between address and zip code and dependencies between address, phone number and city to correctly impute the desired value. We further notice that while the smaller models fail to impute the correct values, their generated outputs have the correct semantic type of the missing attribute without out any demonstrative examples.

B.2 Slice analysis

We tease apart the differences in encoded knowledge across the different training regimes through a slice-based performance analysis on the Restaurant dataset. We use the frequency counts of

city names in the training set to define three frequency-based subclasses (Table 4). We find that for entities that do not occur in the training set, GPT-3-175B is the only model that is able to correctly impute the missing values. Moreover, we observe that for rare entities—a city value of West LA—these pattern can only be learned through full or lightweight finetuning. Interestingly, we find that the lightweight approach more accurately learns the rare subclasses relative to the fully-finetuned approach in both the 100% and 50% training set regime. We hypothesize that this is because the adapter model is smaller, and is thus less prone to overfitting the training set.

C EFFECTS OF FM BIAS

Because FMs are pretrained on large corpuses of web text, they inherit the biases of the data they are trained on. As a result, FMs have been found to contain a number of social biases (e.g. gender, religion, race and more) [8, 55]. It is important to recognize the effects of these biases when utilizing FMs in downstream applications. When applying FMs to data management tasks, we need to be wary of the fact that the inherent biases of these models may be propagated through their actions. Concretely, for data repair, the encoded biases of the FM may cause it to replace or correct entries in a biased manner (e.g. adding a suffix of Ms. to a traditionally female name). As a result, we need mechanisms for mitigating model bias and monitoring for bias in model outputs in deployment settings. Recent work takes a first step in this direction by proposing methods for automatically detecting bias-sensitive tokens and correcting these tokens to mitigate biases [47].