Can Foundation Models Wrangle Your Data?

Avanika Narayan
Stanford University
avanika@cs.stanford.edu

Ines Chami
Numbers Station
ines.chami@numbersstation.ai

Laurel Orr
Stanford University
lorr1@cs.stanford.edu

Christopher Ré
Stanford University
chrismre@cs.stanford.edu

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 five 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 domain-specific data, and opportunities to make data management systems more accessible to non-experts. We make our code and experiments publicly available at: https://github.com/HazyResearch/fm_data_tasks.

PVLDB Reference Format:
Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. Can Foundation Models Wrangle Your Data?. PVLDB, 16(4): 738 - 746, 2022. doi:10.14778/3574245.3574258

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/HazyResearch/fm_data_tasks.

1 INTRODUCTION
Foundation Models (FMs) [19] 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 [20], knowledge base construction [79], and information retrieval [39]. As they have scaled to hundreds of billions of parameters (e.g. GPT-3 [20], PaLM [22]), 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 [22]. 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) [49] and deep learning (DL) [57, 74] methods—namely pretrained language models (PLMs) like BERT [32]—to semantically complex data tasks. However, these approaches still require a significant amount of engineering effort as they rely on:

- **Task-specific architectures**: Data cleaning and integration encapsulate many different tasks such as entity matching [77], schema matching [92], and error detection [41]. Existing approaches, whether they are rule-, ML- or DL-based, vary greatly from one task to the other, often with complex, task-specific 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 [24, 84]. Consequently, systems can be brittle and fail to generalize to a diverse set of domains.

---

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 16, No. 4 ISSN 2150-8097.
doi:10.14778/3574245.3574258

---

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.
• **Labeled data:** ML- and DL-based solutions require copious amounts of hand-labeled data [9]. For instance, PLMs that have achieved state-of-the-art (SoTA) results on data tasks (e.g. Ditto [38]) 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 an extensive set of common entities, and thus do not rely on human-engineered rules to acquire knowledge [82].

• **Limited to no labeled data:** FMs can be applied to a breadth of tasks with little to no labeled data (e.g. few-shot and zero-shot). When a FM needs to be fine-tuned, it typically needs dramatically less labeled data to achieve competitive results [46].

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 advantages 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 several 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 five enterprise data tasks: entity matching, error detection, schema matching, data transformation, 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 since prior approaches are fully-finetuned on task-specific labeled data for these tasks, while GPT-3-175B is simply 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.

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: for entry $e_i \in D$ we have $e_i = \{e_{ij} \mid 1 \leq j \leq m\}$ where for attribute $j$, $e_{ij} = (attr_j, val_j)$.

**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 [77] for a survey) and methods broadly fall into three categories: rule-based, crowd-based [36, 96] and ML/DL-based [49, 74]. Recently, methods relying on PLMs [57] 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 $val_j$ has an error. The task is framed as a classification problem where the goal is to predict if $val_j$ is correct for a given $e$.

ED has been studied extensively in both academic and industrial settings [7]. There are a number of successful commercial
We now give an overview of language FMs, starting from very early, widely different from the FMs pretraining objective (e.g., code generation with only a few examples (few-shot prompting), and, in some cases, (175B parameters) has the capacity to solve natural language tasks 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 [22].

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 [78], BERT [33], RoBERTA [64]—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 [20] 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 have 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 [22].

Emergent Behaviors

Interestingly, the biggest GPT-3 variant (175B parameters) has the capacity to solve natural language tasks with only a few examples (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 (e.g., code generation [100], Trivia QA [20, 59] and arithmetic [20]). This in-context learning behavior can be thought of as the FM “locating” an already-learned behavior [85]. Smaller models (<10B parameters) typically require some form of task-specific finetuning. We study ways to use smaller FMs on data tasks in the full report [75].

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 [57] and serialize the attribute values and entry as follows:

\[
\text{serialize}(e) := \text{attr}_1 : \text{val}_1 \ldots \text{attr}_m : \text{val}_m
\]

If the attribute value is NULL, it is serialized 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 [105]. In Section 4.3 we study the sensitivity of FMs to sub-selection choices.

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, 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

\[
\text{Product A is serialize}(e). \text{Product B is serialize}(e'). \text{Are Product A and Product B the same?}
\]

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

\[
\text{attr}_1 : \text{val}_1 \ldots \text{attr}_j ?
\]

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

\[
\text{Is there an error in attr}_j : \text{val}_j ?
\]

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 is to sample random examples from a labeled dataset. However, this approach often causes high variance in FM performance [62]. Moreover, the ordering of examples in the

1Interestingly, 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.
prompt can have a non-trivial impact on the downstream performance [62, 66, 105]. Although recent work has tried to systematize the prompt tuning process, it is still an open area of research [55, 90].

**Manual** Another approach is to manually construct examples that yield good performance on a held-out validation set that is 10% of the original labeled dataset. This approach is more costly (requires 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 prompt tuning is significantly lower. 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 [76].

**Datasets** For entity matching, we use the standard Magellan benchmark [49]. For schema matching, we choose a challenging dataset, Synthea, from the OMAP benchmark [104]. For data transformation, we choose two challenging datasets from the TDE benchmark [40]. For imputation, we choose two challenging datasets from [72]: Restaurants and Buy. Finally, for error detection, we evaluate on the benchmark Hospital and Adult datasets which are used across several data cleaning papers [23, 41, 84].

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 [41, 72] to generate the dataset splits. For the Adult dataset, we evaluate over a randomly sampled set of 1K rows due to budget constraints.

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

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

| Task          | Imputation | Error Detection |
|---------------|------------|-----------------|
| Dataset       | Restaurant | Buy | Hospital | Adult |
| HoloClean     | 33.1       | 16.2 | 51.4     | 54.5  |
| IMP           | 77.2       | 96.5 |          |       |
| HoloDetect    | -          | -   | 94.4     | 99.1  |
| GPT3-175B \((k=0)\) | 70.9       | 84.6 | 6.9      | 0.0   |
| GPT3-6.7B \((k=10)\) | 80.2       | 86.2 | 2.1      | 99.1  |
| GPT3-175B \((k=10)\) | 88.4       | 98.5 | 97.8     | 99.1  |

### Table 3: Data integration results, measured in accuracy for data transformations and F1 score for schema matching where \( k \) is the number of task demonstrations.

| Task          | Data Transformation | Schema Matching |
|---------------|---------------------|-----------------|
| Dataset       | StackOverflow | Bing-QueryLogs | Synthea |
| Previous SoTA | 63.0       | 32.0             | 38.5   |
| GPT3-175B \((k=0)\) | 32.7       | 24.0             | 0.5    |
| GPT3-175B \((k=3)\) | 65.3       | 54.0             | 45.2   |

**Evaluation Metrics** For the error detection, and schema/entity matching, we evaluate performance using F1 score. For imputation and data transformation, we evaluate using accuracy.

**Baselines** We compare against the SoTA methods for each task. For entity matching, we benchmark against Ditto [57], the current SoTA DL-based approach which finetunes BERT [33]. For data imputation, we benchmark against IMP [72], which finetunes RoBERTa [64], and HoloClean [84], a statistical-based SoTA data repair engine. For schema matching, we compare against the SoTA model, SMAT [104], which finetunes an attention-based BiLSTM. For data transformations, we compare against TDE [40], a SoTA search-based solution. Finally, for error detection, we compare against HoloClean and HoloDetect [41], a data-augmentation based ML approach.

### 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, 2 data transformation, 2 error detection and 1 schema matching benchmark dataset(s) (see Table 1, Table 2, Table 3). The FM outperforms fully-finetuned, SoTA PLM-based approaches for entity matching [57], and data imputation [72]. For error detection, the few-shot approach outperforms the ML-based SoTA method, HoloDetect, which uses data augmentation and weak supervision to perform well.

**Zero-shot Performance** In the zero-shot setting, we observe that the FM outperforms statistical-based approaches and standard data repair engines [84] for imputation. On entity matching, the
Table 4: Entity matching ablation results (F1 score) for different prompt formats (k=10). For all datasets, we evaluate on up to 200 samples for cost purposes.

| Prompt Format                                      | Beer       | iTunes-Amazon | Walmart-Amazon |
|----------------------------------------------------|------------|---------------|----------------|
| Prompt 1 (w. Attr. & Example Select.)              | 100 ± 0.00 | 96.2 ± 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 1 (w. Attr. & w/o Attr. names)              | 80.0 ± 0.00| 94.5 ± 0.00   | 84.2 ± 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 equivalent?"|            |               |                |
| Prompt 2: "Are Product A and Product B the same?"  |            |               |                |

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. These 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 they also have encoded knowledge that is needed to correct and complete records (e.g., functional dependencies between address and zip code). We analyze encoded large FM knowledge in the full report [75].

On the entity matching datasets that the FM does not achieve SoTA on, we find 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: symantec pcanywhere 11.0 host only cd-rom win 98 nt w2k me. manufacturer: symantec. price: NULL" and "name: symantec pcanywhere 11.0 windows. manufacturer: NULL. price: 19.99." We 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 4). 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 4).

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 equivalent?" (Prompt 1) with "Are Product A and Product B the same?" (Prompt 2). This minor modification results in an average 9.4 F1 point performance gap in the datasets in Table 4.

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 4). We run this experiment over three different random seeds and report the results in Table 4. 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 [105]. This performance variance suggests that iterative prompt programming is an essential human-in-the-loop process for FM usage [63]. However, some works suggest that smarter, automatic example selection methods can help close the gap between random example selection and human-in-the-loop prompt selection [62]. 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. We discuss the opportunities, practical considerations, and technical challenges associated with this vision next.

5.1 Opportunities for FMs
Natural Language Interactions FMs usher in a new era of human-machine collaborative workflows wherein users spend less time labeling data and finetuning models and more time writing natural
language prompts that are representative of the task at hand. As the necessity of writing code decreases, we envision systems that are more accessible to non-machine learning experts (e.g., business users). In future work, we seek to better understand the human-in-the-loop prompt engineering pipeline, especially in the context of data management practices.

Model Prototyping The data integration and management pipeline can be categorized into three distinct stages: discovery and design, development, and deployment [43]. We propose that FMs will be most useful in the discovery and design phase when training data is less available. In this setting, FMs enable rapid prototyping of data models via prompting. In some cases, the FM’s out-of-the-box performance will be sufficiently high, obviating the need to train a task-specific model. In others, we can use the FM to label and generate data with human-in-the-loop feedback. When a sufficient amount of data has been collected, transitioning to the fully-supervised model development regime is the optimal choice.

Passive Learning From User Exhaust In the enterprise setting, organizations accumulate a staggering amount of data exhaust—the informational byproduct that streams from devices, products, and workforce management practices [1]. Because FMs are pretrained in an unsupervised fashion with a simple token prediction objective (see Section 2.2), they can learn over any raw and unlabeled sources of data [34, 83, 101]. As such, FMs can effectively ingest the exhaust from the entire data stack (from system logs to structured data). Learning from data analyst exhaust (e.g., clicks over GUI) is also an opportunity to improve FM performance for these data tasks.

5.2 Practical Considerations of FMs

Integration in Data Management Workflows FMs take text as input and generate text as output. As a result, they are limited in their ability to directly take actions over the graphical user interfaces (GUIs) of data management software (DMS) (e.g., Snowflake). Given that a majority of data analyst time is spent working in these environments, we need ways of translating natural language specifications of data tasks (e.g., “remove all nan cells”), to actionable operations on these GUIs. Excitingly, new work demonstrates how to augment FM capabilities such that they can effectively take actions on web and mobile interfaces [10, 95]. These works suggest the possibility of directly integrating FMs with DMS.

Interaction with Existing Systems FMs can interact with existing DMS by either replacing them or utilizing their outputs to conduct downstream tasks. In terms of system replacement, for tasks where the required rules and logic are not encoded in the FMs knowledge, it is an open question [31] as to how to systematically translate the rules and logic (e.g., domain-specific dependencies) to natural language inputs to the FMs. For system integration, we need ways of systematically incorporating the outputs of existing systems (e.g., dataset pattern discovery) in natural language prompts.

Prior work has demonstrated how to “ground” FM-based text-to-SQL prompts with database schemas and metadata [29]. Similar ideas could be used when incorporating system outputs with FMs.

Debuggability FMs are non-deterministic and can make unexpected errors. In order to use these models in data management pipelines, we need mechanisms for increasing transparency and debuggability of pipelines. One possible approach is to collect and monitor model confidence scores. Prior work demonstrates that a FM can “learn to express uncertainty about its own answers in natural language” [60]. Another approach is to decompose tasks into chains of “primitive operations”, enabling for more transparency and visibility of specific failpoints [99].

5.3 Technical Challenges of FMs

Domain Specificity A key challenge in applying FMs to data management tasks is that the best performing models (e.g., GPT-3), can only be accessed via API requests. This motivates the need for better open-source models which are competitive with closed-source models. Excitingly, recent work [14] has demonstrated that techniques such as prompt ensemble and prompt reframing can enable open-source models like GPT-J-6B [94] to out-perform GPT3-175B on popular natural language understanding benchmarks. Extending this work to data management tasks is an interesting research direction.

Prompt Engineering and Automation Prompting is the primary mechanism for adapting FMs to new tasks. However, prompting requires some manual effort to design performant prompts. In the DI setting, prompts can be sensitive to things such as data schemas and minor formatting variations. A few automated approaches have been developed which reduce the manual effort needed to construct prompts: soft prompt tuning (e.g., optimizing a sequence of task-specific vectors which are appended to the text prompt) [52, 56] and learning to retrieve better in-context examples [86]. Adapting these works to the challenges of prompting for data tasks is an open area of research.

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-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 pretrained 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.
REFERENCES

[1] 2018. How to turn data exhaust into a competitive edge. https://knowledge.wharton.upenn.edu/article/tur-tot-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/terms/decreasing-cost-of-storage

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

[6] 2022. Decreasing cost of storage. https://www.iotone.com/term/decreasing-cost-of-storage

[7] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In Proceedings of the 2013 ACM SIGMOD international conference on management of data. 601–612.

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

[9] 2021. ACT-1: Transformer for Actions. https://www.adept.ai/act

[10] Adept. [n. d.]. ACT-1: Transformer for Actions. https://www.adept.ai/act

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

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, et al. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. arXiv preprint arXiv:2206.12477 (2022).

[13] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. DataWig: Scaling laws for neural language models, external knowledge sources and discrete reasoning. arXiv preprint arXiv:2205.00445

[14] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In Proceedings of the 2011 SIGCHI conference on human factors in computing systems. 1809–1818.

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

[16] Aminah Alzahra, Chaitanya Gokhale, Sanjib Das, Anh Dao, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojun Zhu. 2014. Corleone: Hands-off crowdourcing for entity matching. In Proceedings of the 2014 ACM SIGKDD international conference on Management of Data. 601–612.

[17] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. DataWig: Scaling laws for neural language models, external knowledge sources and discrete reasoning. arXiv preprint arXiv:2205.00445

[18] Alireza Heidari, Joshua McGrath, Ihab F Ilyas, and Theodoros Rekatsinas. 2019. DataWig: Scaling laws for neural language models, external knowledge sources and discrete reasoning. arXiv preprint arXiv:2205.00445

[19] Suchin Gururangan, Ara Marasovic, Swabha Swayamdipta, Kyle Lo, It Belbagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: adapt language models to domains and tasks. arXiv preprint arXiv:2004.10994 (2020).

[20] Suchin Gururangan, Ara Marasovic, Swabha Swayamdipta, Kyle Lo, It Belbagy, 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.

[21] Chaitanya Gokhale, Sanjib Das, Anh Dao, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojun Zhu. 2014. Corleone: Hands-off crowdourcing for entity matching. In Proceedings of the 2014 ACM SIGKDD international conference on Management of Data. 601–612.

[22] Adept. [n. d.]. ACT-1: Transformer for Actions. https://www.adept.ai/act
science stacks. Proceedings of the VLDB Endowment 9, 13 (2016), 1581–1584.

50. AGI Labs. 2022. STANDING ON THE SHOULDERS OF GIANT FROZEN LANGUAGE MODELS. prepprint (2022).

51. Zhirong Huang, Minyu 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=H1eATkRt5V.

52. Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the 2022 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/v1/2022.emnlp-main.243

53. Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 3045–3059.

54. Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jianaji, Dor Muhl-gay, Yoni Osnir, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, et al. 2022. Standing on the shoulders of giant frozen language models. arXiv preprint arXiv:2204.01819 (2022).

55. Xiang Li, Yong Liu, 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.

56. Xiang Li, Yong Liu, 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). Associations for Computational Linguistics, Online, 4582–4597. https://doi.org/10.18653/v1/2021.acl-long.353

57. Yuliang Li, Jiemin Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2019. Blocking and filtering techniques for entity resolution: A survey. In Proceedings of the VLDB Endowment 14, 1 (2020), 50–60.

58. Paul Pu Liang, Chuyu 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.

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

60. Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching Models to Express Their Uncertainty in Words. In Proceedings of the 60th Annual Meeting 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

61. Fabio Petroni, Tim Rockiäschel, Sebastian Riedel, Patrick Lewis, Antin Baldwin, 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.

62. 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.

63. Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Fan-Keng Sun. 2020. Conditioned natural language generation with pre-trained language models. Proceedings of the 2018 International Conference on Management of Data. 3505–3506.

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

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

66. Theodoros Rekatsinas, Xu Chu, Ibhy F Ilyas, and Christopher Ré. 2017. How to Leverage Large Language Models. arXiv preprint arXiv:2209.08655 (2022).

67. Lalia Reynolds and Kyle McDowell. 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.

68. Ondar Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. arXiv preprint arXiv:2112.08633 (2021).

69. Victor Sanh, Albert Wehson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Charpion, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multi-task Prompted Training Enables Zero-Shot Task Generalization. In International Conference on Learning Representations. https://openreview.net/forum?id=9VhWD4W1

70. Timo Schick and Hirsch Schütte. 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. 2359–2352.

71. Michael Stonebraker, Daniel Bruckner, Ihab F Ilyas, and Christopher Ré. 2017. HologDB: A Declarative Light BERT for Self-supervised Representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 1470–1479.

72. Bashir Bhuniyar, Nishant Gupta, Suvina Prakash, and Yuliang Li. 2020. TECH: A Few-Shot Prompt Order Sensitivity. In Proceedings of the 2020 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 4896–4908.

73. Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast Model Editing at Scale. In International Conference on Machine Learning. arXiv preprint arXiv:2203.01872 (2022).

74. Bhairav Bhandari, Manali Rane, and Noah Constant. 2022. BERT4RE: A BERT-based Few-Shot Reasoning Framework. arXiv preprint arXiv:2208.02682 (2022).

75. P. Voigt and A. Von dem Bussche. 2017. The EU General Data Protection Regulation (GDPR). OpenAI. 2021. OpenAI API. https://openai.com/api/

76. Theodoros Rekatsinas, Xu Chu, Ibhy F Ilyas, and Christopher Ré. 2017. How to Leverage Large Language Models. arXiv preprint arXiv:2209.08655 (2022).
[96] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. 2012. CrowdER: Crowdsourcing Entity Resolution. Proceedings of the VLDB Endowment 5, 11 (2012).

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

[98] Jason Wei, Xiuzhi 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).

[99] Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. In CHI Conference on Human Factors in Computing Systems. 1–22.

[100] 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.

[101] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. Videogpt: Video generation using vq-vae and transformers. arXiv preprint arXiv:2104.10157 (2021).

[102] Yi Yang, Mark Christopher Siy Uy, and Allen Huang. 2020. Finbert: A pretrained language model for financial communications. arXiv preprint arXiv:2006.08097 (2020).

[103] Ehtisham Zaidi and Sharat Menon. 2022. Magic Quadrant for Data Integration Tools.

[104] Jing Zhang, Bonggun Shin, Jinho D Choi, and Joyce C Ho. 2021. SMAT. An attention-based deep learning solution to the automation of schema matching. Advances in databases and information systems. ADBIS 12843 (2021), 260.

[105] 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.