arXiv:2306.09610v2 [cs.DB] 27 Sep 2023

Chorus: Foundation Models for Unified Data Discovery and Exploration

Moe Kayali
kayali@cs.washington.edu
University of Washington

Anton Lykov
alykov@cs.washington.edu
University of Washington

Ilias Fountalis
ilias.fountalis@relational.ai
RelationalAI

Nikolaos Vasiloglou
nik.vasiloglou@relational.ai
RelationalAI

Dan Olteanu
olteanu@ifi.uzh.ch
University of Zurich

Dan Suciu
suciu@cs.washington.edu
University of Washington

ABSTRACT

We apply foundation models to data discovery and exploration tasks. Foundation models are large language models (LLMs) that show promising performance on a range of diverse tasks unrelated to their training. We show that these models are highly applicable to the data discovery and data exploration domain. When carefully used, they have superior capability on three representative tasks: table-class detection, column-type annotation and join-column prediction. On all three tasks, we show that a foundation-model-based approach outperforms the task-specific models and so the state of the art. Further, our approach often surpasses human-expert task performance. We investigate the fundamental characteristics of this approach including generalizability to several foundation models, impact of non-determinism on the outputs and syntactic/semantic signals. All in all, this suggests a future direction in which disparate data management tasks can be unified under foundation models.

KEYWORDS

Data discovery, Data Exploration, Information Integration, Foundation Models, Large Language Models

1 INTRODUCTION

Data discovery and exploration are major components of the workflow of analysts and data scientists. A survey conducted by the Anaconda data-science platform in 2021 found that analysts spend 40% of their working hours on data loading and cleaning [2]. Even with this colossal effort, 60-70% of data within an enterprise still goes unused for analytics [25], remaining as *dark data* [28, 70].

Recent developments in large language-models (LLMs) have unlocked human-level performance on diverse domain tasks. The discovery that these models can generalize to diverse domain-specific tasks that they have not been trained on [3, 31, 64, 65] has led to emergence of the term *foundation models* [5].

Despite their promise, serious risks have hampered the reception of foundation models. These include: spurious generation (including "hallucination") [29], factual recall limitations [44], bias [24], dataset contamination [19], logical shortcuts [55] and fallacies [43]. Naïve deployment can lead to unanticipated problems: it has already led to legal action [15] and recalls by major corporations [27]. These risks are now acknowledged by even the creators of these models [7, 50, 59].

The goal of this paper is to demonstrate the utility of foundation models to the data discovery and exploration domain while mitigating the aforementioned risks. We select three representative tasks to show the promise of foundation models: ① *table-class detection*, ② *column-type annotation* and ③ *join-column prediction.* An outline of our approach is shown in Figure 1. We call this approach Chorus.

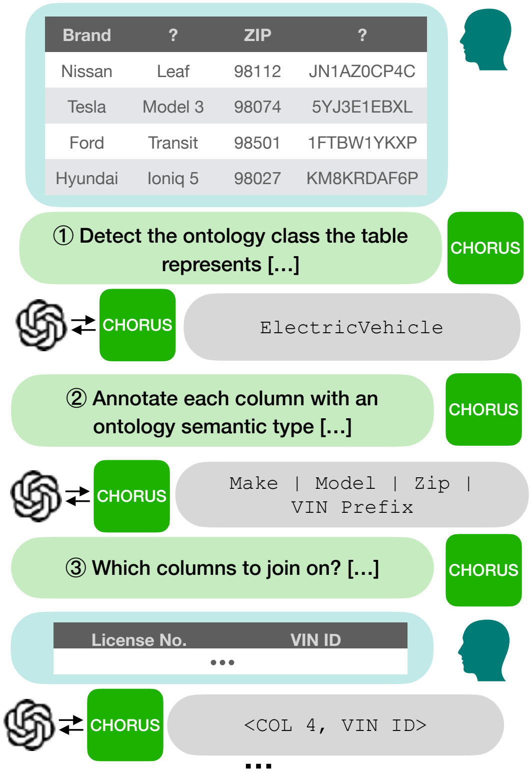


Figure 1: Data discovery tasks considered in this work. Given an ontology, such as DBPedia, ① we assign an overall type to the table and ② we annotate the columns with semantic types. Last, given another table, ③ we predict the join column. The user provides the data while Chorus interacts with the foundation model. Data from [49], full prompts in Figure 5.

Contributions. We summarize our contributions:

- The first work to use foundation models for the data discovery tasks of table-class detection, column-type annotation and join-column prediction;

- Propose a novel system, CHORUS, whose flexible architecture enables the synthesis of multiple data discovery tasks and deploying risk mitigations;
- Design task-specific approaches that exploit zero- and few-shot strategies and allow information flow between tasks;
- Introduce novel mitigations, including nearest-neighbor matching and *anchoring*, to reduce foundation-model risks specific to this domain;
- Empirically validate CHORUS, comparing its performance with the state-of-the-art baselines across three individual tasks.

Discussion. Prior work has addressed these tasks individually. Landmark approaches like Sherlock [32] trained deep model architectures for a specific task, requiring 100K-1M labeled data points. More recent work such as DoDuo [57] and TaBERT [69] has focused on *representation learning*, learning embeddings for structured data by improving their performance on one or more downstream tasks.

Foundation models allow a substantially different approach: rather than the classical architecture where the outputs of the model are task-specific, the inputs and outputs of the model are natural language text. Training occurs not on tables or data management tasks specifically, but on general text. Performance on domain-specific tasks is solely by generalization.

This results in a high degree of flexibility. Novel tasks can be specified in natural text, without need for expensive data collection—task examples, metadata and constraints are all incorporated into the task easily. Another advantage of this approach is a **unified architecture**: tasks can utilize the overall context and previous outputs. For example, in Figure 1 the table class of `ElectricVehicle` can help with deducing the outputs of `Make` and `Model` in the next task.

Outline. Section 2 defines the three tasks investigated in this paper. Section 3 describes the architecture of CHORUS and key approaches. We evaluate the performance of CHORUS in Section 4's experiments. In this section, we also investigate the fundamental characteristics of this approach. We offer a discussion of those results in Section 5. This includes a discussion of promising future directions. Finally, we place this work within the literature in Section 6, discussing related works.

2 BACKGROUND

2.1 Tasks

We assume to be given a *data collection* consisting of a number of relational tables $T_1, T_2, \ldots$. Each table T_i consists of a number of columns, or attributes, $A_1, A_2, \ldots$ and a number of rows, or tuples, $r_1, r_2, \ldots$ The name of a table T_i is, in general, non-informative, for example it may be simply a sequential ID. The columns may optionally have a name $H_1, H_2, \ldots$ or consist only of values.

In addition to the data collection, we are also given a reference ontology of table classes $C_1, C_2, \ldots$, and a reference ontology of column types $\tau_1, \tau_2, \ldots$. For example, the DBPedia.org types for the table classes include https://dbpedia.org/ontology/Lake, https://dbpedia.org/ontology/Actor and https://dbpedia.org/ontology/Continent and column types include https://dbpedia.org/ontology/areaTotal and https://dbpedia.org/ontology/birthDate.

We consider three tasks of interest to perform over this data collection.

Definition 2.1 (① Table-class detection)**.** For each table T_i, determine its appropriate class C_j, such that every row $r_1, r_2, \ldots$ represents an instance of the C_j type. We adopt this definition from [38].

For example, table-class detection on the table given in Figure 1 could output `ElectricVehicle`, since each row of that table is an instance of that class. Alternatively stated, the table is about `ElectricVehicles`.

Definition 2.2 (② Column-type annotation)**.** For each table T_i, find a mapping from its attributes (columns) $A_1, A_2, \ldots$ to the reference column types $\tau_1, \tau_2, \ldots$, such that each value in A_i is an instance of the τ_i type. See [1, 17].

For example, column-type annotation on the first column in Figure 1 could output `Manufacturer`, since the values are the respective manufacturers of each `ElectricVehicle`.

Definition 2.3 (③ Join-column prediction)**.** Assume an *execution log* L, a history of user actions including table joins and their join conditions, which maps many $(T_i, T_j) \rightarrow (A_k, A_l)$ where $A_k \in T_i, A_l \in T_j$. Given two tables T and T', with columns $A_1, \ldots$ and $A'_1, \ldots$ respectively, the *join-column prediction* task is to suggest a pair (A_k, A'_l) of columns such that the equality condition $A_k = A'_l$, which can be used to join the the tables, matches with the choice in the execution log L. For more discussion, see [68].

For example, given the table in Figure 1 and another table `car_registration(name, vehicle_id_number)`, join-column prediction could output `(VIN_prefix, vehicle_id_number)`. The correctness of the prediction depends on the ground truth of which columns the user did in-fact join on.

2.2 Foundation models

We discuss some properties of foundation models relevant to this project. For an overview of foundation models and their capabilities, we recommend this comprehensive treatment [7]. For the following sections, we examine the capabilities of the GPT-3.5 model [50].

Relational data In order to allow the foundation model to parse relational data, we serialize it. This allows the foundation models to parse the relational data and then reason about it. Consider for example the input table in Figure 1. We serialize the table into a comma-separated values (CSV) format. Inputting this into a foundation model, in this case GPT-3.5, shows that the model can reason about the relational structure of the data, as seen in Figure 2.

User: Consider this CSV data:

```
Brand, ␣, ZIP, ␣,
Nissan, Leaf, 98112, JN1AZ0CP4C,
Tesla, Model 3, 98074, 5YJ3E1EBXL,
Ford, Transit, 98501, 1FTBW1YKXP,
Hyundai, Ioniq 5, 98027, KM8KRDAF6P,
```

User: Print the header row verbatim.

Model: The header row from your provided CSV data is:
`Brand, ␣, ZIP, ␣,`

User: Output the value in the second column of the third data row. Reply with just the value.

Model: `Transit`

Figure 2: Foundation models can parse relational data.

From the English utterances "header row" and "second column of the third data row", the model is able to reason about the provided table and output the intended values. This requires understanding of schemas, relations, tuples, attributes and values. Thus all the basic blocks of the relational model are present in the model.

Ontologies Foundation models contain knowledge of ontologies such as DPBedia.org, Freebase and Wikidata. We focus on universal ontologies, that is, ontologies that aim to represent all entities in general. This is in-line with findings that foundation models encode highly technical knowledge, such as clinical reasoning [56] or electrical engineering principles [58].

As earlier, we demonstrate with an example in Figure 3.

User: Which DBPedia.org classes represent Taipei 101? Answer briefly.

Model: [. . .] Based on my training, common DBPedia classes that could represent a notable skyscraper such as Taipei 101 might include: dbo:Skyscraper, dbo:Building, dbo:ArchitecturalStructure [. . .]

User: What is WikiData property P36?

Model: As of my knowledge cutoff in September 2021, the property P36 in Wikidata corresponds to the "capital" of a country, state, province, or region [. . .]

Figure 3: Foundation models contain information about popular ontologies. Here the model correctly memorizes facts about DBPedia and Wikidata.

This shows that the model encodes information about popular ontology classes and properties. Note that this information is not necessarily complete nor correct. We emphasize the current generation of foundation models does not have access to data lookup abilities, despite them occasionally generating output claiming to have done precisely that [15].

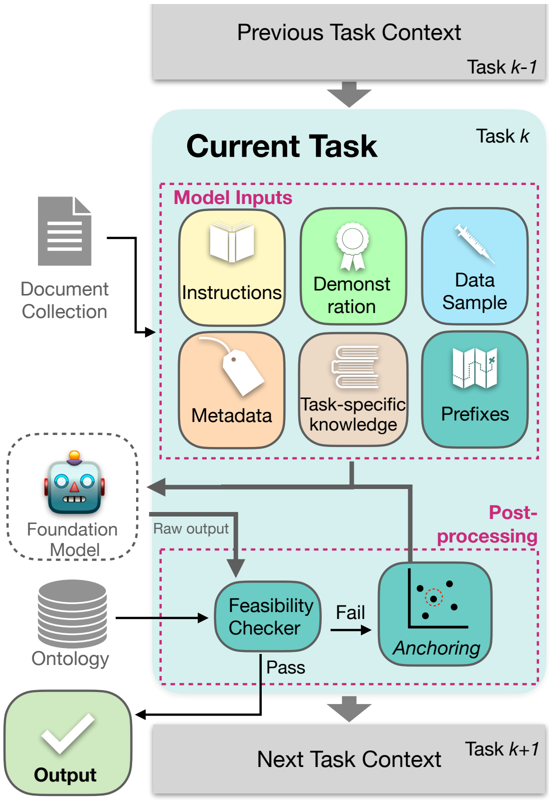


Figure 4: Chorus system architecture.

3 APPROACH

We outline the structure of Chorus in this section. First, we explore the core idea of ingesting relational data with foundation models and performing data exploration tasks in Subsection 3.1. Next, we describe the necessary post-processing and mitigations we develop in Subsection 3.2.

Figure 4 shows the architecture of the system. Chorus has a unified architecture which runs multiple tasks in the same context, allowing for information flow. Each task is run sequentially, with the output of one task fed as context into future tasks.

For each task instance, Chorus generates a prompt by concatenating six inputs: context, demonstration, data samples, metadata, task-specific knowledge, and prefixes. They form the "Model Inputs" box in Figure 4 and are color-coded so that they match the colored prompt components in Figure 5. This natural language input is then fed to the foundation model. The output is then subject to post-processing: checks of parsability and feasibility are conducted. If these pass, the output is extracted. Otherwise, we activate a mitigation process, called *anchoring*, in order to repair the error and prevent its propagation.

3.1 Model Inputs

We discuss what inputs are provided to the foundation model and how they are pre-processed and synthesized. We discuss the six components of the Model Inputs module in Figure 4, individually.

Legend: Instruction, Demonstration, Data sample, Metadata, Task-specific knowledge, Prefixes.

~~~
For the following CSV sample, select one DBpedia.org
ontology that represents the dataset from the following
list:
AcademicJournal, AdministrativeRegion, Airline, Air
port, Animal, BaseballPlayer, Bird, Book, Building,
City, Company, Country, Cricketer, Currency, Election,
FictionalCharacter, Film, GolfPlayer, Hospital, Hotel,
Lake, Mammal, Monarch, Mountain, Museum, Newspaper,
Novel, Person, Plant, PoliticalParty, RadioStation,
Saint, Scientist, Swimmer, TelevisionShow, University,
VideoGame, Work, Wrestler.
For example, for a dataset about hospitals, return
`https://dbpedia.org/ontology/Hospital `. Begin your
answer with'https://dbpedia.org/ontology'
```
Brand, ␣, ZIP, ␣,
Nissan, Leaf, 98112, JN1AZ0CP4C,
Tesla, Model 3, 98074, 5YJ3E1EBXL,
Ford, Transit, 98501, 1FTBW1YKXP,
Hyundai, Ioniq 5, 98027, KM8KRDAF6P,
```
~~~

(a) Table-class detection

~~~
Consider this example. Input:
```
Name, Famous Book, Rk, Year
Fyodor Dostoevsky, Crime and Punishment, 22.5, 1866
Mark Twain, Adventures of Huckleberry Finn, 53, 1884
Albert Camus, The Stranger, -23, 1942
```
Output:
`dbo:author, dbo:title, Unknown, dbo:releaseDate`.
For the following CSV sample, suggest a DBPedia.org
Property for each column from the `dbo:`namespace.
``` [...] ```
~~~

(b) Column-type annotation

~~~
Given two Pandas Dataframes, suggest what `pd.merge`
parameters to use to join the dataframes.
df1 =
``` [...] ```
df2 =
``` [...] ```
Complete the correct Pandas merge command.
`pd.merge(df1, df2, left_on=
~~~

(c) Join-column prediction

Figure 5: Prompts used in this paper, materialized with examples. Most prompt elements are fixed—only the data sample and metadata change for each instance.

```
Brand, ␣, ZIP, ␣,
Nissan, Leaf, 98112, JN1AZ0CP4C,
Tesla, Model 3, 98074, 5YJ3E1EBXL,
Ford, Transit, 98501, 1FTBW1YKXP,
Hyundai, Ioniq 5, 98027, KM8KRDAF6P,
```

Figure 6: The contents of the table from Figure 1 serialized in a comma-separated value format. This corresponds to the blue text in Figure 5.

These correspond to the six color-coded prompt components in Figure 5. Once generated, all the above inputs are concatenated to into a single prompt provided to the model.

Instructions. A description of the specific task (table-class detection, column-type annotation or join-column prediction) is provided to the foundation model in natural language. These are shown in yellow in Figure 5. For example, we translate the formal Definition 2.1 of the first task, table-class detection, into the English sentence "For the following CSV sample, select one DBpedia.org ontology that represents the dataset." For the third task, join-column prediction, we utilize a code-completion approach. We frame the task as code-completing a Pandas fragment that performs a join, with the code to complete shown in Figure 5c. We choose Pandas because it is a very popular framework, with more than millions of example lines of code on the web. This is the *zero-shot prompt* setting: the model can be provided with instructions for a novel task and performs them directly.

Demonstration. For the first two tasks, we use the foundation models with task examples as an additional input: this is called the *few-shot prompt* setting. The model is given a few demonstrations of task completion, including inputs and outputs. This is shown in Figure 5 as green text.

Data sample. Foundation models can understand relational data. By serializing the input tables, we can input them into foundation models. For example, consider the example table from Figure 1 in the introduction. Serializing the table allows the foundation model to ingest the data. In this paper, we use the comma-seperated values (csv) format. This gives us the representation in Figure 6, which would be inserted as the blue text in Figure 5.

Because the models have a limited context window size—typically in the few thousands of tokens—tables cannot always be ingested as a whole. Instead, we always serialize a sample of the rows. Intuitively, this is acceptable because the tasks we consider can be completed without reviewing all the rows, *i.e.* it is sufficient to consider a few values when determining the type of a column. We find that a sample of size five is sufficient for the tasks in this work.

Metadata. Schema information including column names (headers) and keys can be incorporated into the input, above the serialized data sample. We found that foundation models can adaptively infer whether the first column of the input is a header or data row, with no modification of the input required. This is shown in orange in Figure 5. Due to the flexibility of the input format, we add supplemental information to the prompt where available. For example, contextual information about the data source is added, such as "this dataset is from a page titled *Washington State Open Government*."

Task-specific knowledge. For some tasks, additional information can be used to guide the model. For ① table-class detection, if only certain output classes are desired, these can be listed to the model. The model will take these instructions into account when generating an output but they are not hard constraints. The encoding of such additional constraints for the table-class detection task is shown in Figure 5a.

Prefixes. We also provide the model with *prefixes* with which to complete. This includes the DBPedia format for the table-class detection task and a Pandas code fragment for the join-column prediction task. Both prefixes are highlighted in pink in Figure 5. Prefixes increase the likelihood the model will provide the output in a parsable format rather than deviating into a natural language description.

3.2 Post-processing

Once the foundation model has run and provided its natural language output, we perform post-processing to parse into a symbolic representation and mitigate common errors.

Constraint checks. Because the model is not constrained in its outputs, it may not always output a feasible answer. In this setting we impose three constraints: table types must belong to the ontology classes, column types must belong to the ontology properties and joins must be on existing columns. An output is infeasible if in particular, it is not parsable or if it violates any of the three constraints. To guard against this, we parse the output and check the constraint corresponding to the current task. If the constraint fails, the CHORUS performs anchoring.

Anchoring. If the constraint is violated, we do not simply reissue the prompt. This may seem like an appealing option because the non-determinism of the model might allow for a different answer—but this is not a good idea. The reason is that an insidious risk in our setting is hallucination snowballing [72]. This recently formalized phenomena describes how, once a foundation model makes a spurious generation, subsequent generations are more likely to be erroneous: after a misstep, the system tends to make mistakes it would otherwise be able to avoid. We provide an example in Figure 7a: once nonexistent class `animalName` is suggested, another nonexistent class `iucnStatus` follows. Because we maintain context across tasks, we are particularly vulnerable to this.

Instead, we resolve this by a novel domain-specific mitigation we call *anchoring*, shown in Figure 7b. CHORUS maintains the list of embeddings of all feasible answers, e.g. all table classes from DBPedia. The embeddings are computed by running another foundation model and extracting the final layer weights. We use for this purpose an alternative foundation model (GPT-3) that allows us access to its weights, because our main model GPT-3.5 is only available via API access at the time of writing. Then, for each output that violates the constraints, we conduct the following repair process: (1) We extract the embedding for the incorrect output from GPT-3 in the same process as above. (2) We conduct a nearest-neighbor search on the pre-computed embeddings of the feasible answers. (3) We replace the infeasible output by the nearest-neighbor, transformed into the correct answer format. To the best of our knowledge, our anchoring technique is novel, and only applies to our specific problem where the output is constrained. In contrast, for general-purpose natural language processing the domain is unconstrained.

[…]
```
Panthera pardus, Vulnerable,
Falco peregrinus, Least concern,
Graptemys oculifera, Vulnerable,
```
Chorus: Which DBPedia.org property corresponds to the first column?
Model: https://dbpedia.org/ontology/animalName
Chorus: Which DBPedia.org property corresponds to the second column?
Model: https://dbpedia.org/ontology/iucnStatus

(a) *Hallucination snowballing*: once the model outputs the non-existent property `animalName`, it makes the mistake again, producing the non-existent `iucnStatus`.

[…]
Chorus: Which DBPedia.org property corresponds to the first column?
~~**Model:** https://dbpedia.org/ontology/animalName~~
Chorus: https://dbpedia.org/ontology/binomial
Chorus: Which DBPedia.org property corresponds to the second column?
Model:https://dbpedia.org/ontology/conservationStatus

(b) *Anchoring*: After detecting that `animalName` is an error (it is not a DBPedia class), we expunge it from the model's context, and replace it with the nearest-neighbor (see text) `binomial`. This breaks the chain of errors: the next property returned is valid. Note that `binomial` refers to DBPedia class for *binomial nomenclature*: https://dbpedia.org/ontology/binomial.

Figure 7: Example of *anchoring*.

The expunging process works by overriding the dialog history provided to the foundation model. The model takes as input all the previous messages generated by both the client and the model. This allows for generating synthetic histories in which the model outputs are modified. It is illustrated in Figure 7b as a strikethrough.

In the example in Figure 7b, we expunge the model's history of outputting non-existent class `animalName`, replacing it with the correct `binomial` class. The `binomial` class is chosen because it is the nearest neighbor. Because this intervention guarantees a valid class in the prompt history, we avoid hallucination snowballing. After this correction (anchoring), the model is able to generate the correct class for the next column (`conservationStatus`) without further intervention.

4 EXPERIMENTS

We empirically evaluate CHORUS on the three tasks defined in Section 2.1. For each task, we select a task-specific benchmark and compare with baselines representing the state of the art. Table-class

Table 1: *Capabilities of related systems.* Only our system supports all studied tasks out-of-the-box and without additional training.

System	Table-class detection	Column-type annotation	Join-column prediction
DoDuo [57]	●	✓	✗
TaBERT [69]	●	✓	✗
Sherlock [32]	✗	✓	✗
Trifacta Wrangler [61]	✗	●	✓
Chorus	✓	✓	✓

✓ supported out-of-the-box, ✗ no support
● required modification or training data collection (see text)

Table 2: Summary of the datasets used in the paper. Numbers indicate the size of the data used.

Dataset Title	# Tables	Avg. # Columns	Avg. Rows
T2D-class v2	237	7.41	118
VizNet	$\sim$ 330	3.03	5 200
GitNotebooks	300	10.3	78
Overall	867	6.67	2 038

detection ① is evaluated in Section 4.1, ② column-type annotation in Section 4.2, and ③ join-column prediction in Section 4.3.

Baselines. We considered the following state-of-the-art systems for data exploration: relevant systems include TaBERT [69], DoDuo [57], Sato [71], TURL [17], TaBBIE [34], Auto-suggest [68], Trifecta Wrangler [61], Paxata, Tableau Prep, and Sherlock. DoDuo is reported to outperform TURL and Sherlock on column-type annotation [57], so we select it for evaluation. Sato and Sherlock are similar, with Sato utilizing additional signals not found in our benchmarks, so we evaluate the better-established Sherlock. TaBBIE can embed tables but is not trained on column-type annotation unlike DoDuo and Tabert, so we avoid it for the column-type annotation task. TaBERT is a work similar to DoDuo and TURL, but from the NLP community rather than the data management community, so we also test it too. For join-column prediction, Trifacta Wrangler outperforms Paxata and Tableau Prep [68]. Auto-Suggest is reported to outperform Trifacta Wrangler, but is a proprietary research project not released publicly. Thus we select Trifacta Wrangler for testing.

For the evaluated prior works TaBERT, DoDuo, Trifecta Wrangler and Sherlock [32, 57, 61, 69], we utilize each tool if applicable to the task. If the baseline is not designed for a particular task, but can be straightforwardly adapted, we do so. We describe all modifications in the task subsection and always use established adaptations if available. If the modifications required would be extensive enough to become their own research project, we consider that task unsupported. In all cases, we use the pretrained embeddings without modification, as provided by the authors. Table 1 outlines the systems we tested and tasks they support.

DoDuo provides two embedding variants: one trained on the WikiTables dataset and another on VizNet. When using DoDuo as a baseline we test against both, labelling them DoDuo-Wiki and DoDuo-Viz respectively.

Datasets. Table 2 outlines the three experiment benchmarks we use. For the table-class detection task, we test on the T2D-class v2 dataset [53], a "gold standard" corpus of 237 tables, manually annotated by experts with one of 39 DBPedia.org classes. These tables were in turn selected from the Common Crawl corpus of web tables [22]. For column-type annotation, we sample a subset of the VizNet dataset [30], extracted by the Sherlock team [32], comprising 1000 columns with one of 10 types from approximately 330 tables. Sampling the types was necessary because some column types overlapped and we needed to ensure mutually exclusive labels as our setting was single-label rather than multi-label classification.

This is in line with prior work that uses VizNet [33]. For the join-column prediction task, we use a dataset we call GitNotebooks, extracted by the Auto-suggest team [68]. We select 300 tables from that dataset for which we have join data. Here we use a sample as one of the baselines, Trifacta Wrangler, does not have an API but instead predictions must be produced manually. For the first two tasks, which require defining a type system for classes and properties, we use the DBPedia ontology [45] for our experiments. This is a community-sourced ontology and is the standard in previous studies.

Setup. We use the GPT-3.5 model [50] as it is the most widely-available large model with API access at the time of writing. All other code was run on a commodity laptop with 8 physical ARM cores and 16GB of main memory. Running all experiments, including development, came to $2 in API costs.

We evaluate using the metrics *precision*, *recall* and F_1 score. Precision is the proportion of true positive results out of the total predicted positive results, while recall is the proportion of true positive results out of the total actual positive results in the dataset. The F_1 score is the harmonic mean of precision and recall. Since we deal with a multiclass setting, we calculate these metrics for each class separately then aggregate by taking the mean, weighted by the class size. Class-weighted precision, recall and F_1 are the standard metrics in prior work [8, 32, 57, 71].

4.1 Table-class detection

For the first task, ① table-class detection, we tag each table with the DBPedia ontology entry that represents the row-type of the data. Of the 1000 datasets that comprise the T2Dv2 dataset, 237 tables have table-class correspondences available while 763 do not. We denote this subset T2D-class v2. We note that only 40 classes are utilized in this "gold standard" mapping, while DBPedia ontology has 769 classes.

We compare against the baselines DoDuo and TaBert. No approach in the prior work provides out-of-the-box capabilities on this task, so we add a classification layer on top of the pretrained embedding layer. After computing the column embeddings using DoDuo or TaBert, predictions are extracted by adding a pooling layer, fed to a multi-layer perceptron, and then finally taking the soft-max. This is a straightforward method of adapting the embeddings to our multi-class setting, used in prior benchmarks for table-class detection [38]. We fix the embeddings to their pretrained values and learn the weights of the classification layer using five-fold cross-validation.

Supervised variant. To allow for comparisons with prior work, we initially restrict our system to picking out of the 33 classes. This is because all other approaches require training on labelled instances—the baselines cannot predict outside those classes. We test 33 classes rather than 40 because the classes that occur only once cannot be tested on baselines that require supervised training (DoDuo and TaBERT), since a meaningful result requires a disjoint training and test set.

Table 3 shows the results. CHORUS improves on the three baselines—DoDuo-Viz, DoDuo-Wiki and TaBERT—on all metrics. F_1 score is improved by 0.169 points, precision by 17.5 percentage points and recall by 15.5 percentage points. Of the baselines, DoDuo-Wiki provides the best F_1 and precision, while TaBERT provides the comparable recall. The best performing models, TaBERT and DoDuo-Wiki are trained on CommonCrawl, a superset of the T2Dv2 benchmark. DoDuo-Viz which is trained on the VizNet, a dataset disjoint from T2Dv2, has the weakest performance. The numbers for TaBERT are in line with prior replications [38], while to the best of our knowledge this is the first benchmarking of DoDuo on this task.

Unsupervised variant. Next, we relax the classification domain, allowing the foundation model to choose any of the 768 classes of the DBPedia ontology. We then compare the quality of the classes with that of the human-expert labels. DoDuo and TaBERT are not evaluated in this task setting as they cannot predict outside the classes they have observed in training.

For 93% of tables, our system produces correct results. Of that portion, 83 percentage points are comprised of exact matches, while 10 percentage points are *better-than-correct* results. This means the predicted labels are clearly and unambiguously better than those selected by the human expert. Because this is a strong claim, we list all such datasets in Table 4.

For example, consider the dataset with ID 9 in Table 4, which is a list of ancient Egyptian temples and their properties (date of construction, etc.) A text label associated with the dataset, when translated from German, says "these are mainly Egyptian temples." The human expert annotator provided the label `Monarch`. Some of the temples are named after pharaohs, so this is an understandable mistake. Still, CHORUS is able to disambiguate and resolves to the correct class: `Temple`; we conjecture that one important additional signal was the mention of the entity *Faustina Senior*—she is the namesake of a temple and was the wife of a Roman emperor but never a ruler herself.

In another example, dataset ID 176, CHORUS generates a more specific subclass, `RomanEmperor`, than that chosen by the human expert, `Monarch`, most likely because the table is called "emperors".

For the final 6% the answer is incorrect: this can mean the answer is completely wrong or simply worse than the label provided by the expert. This means that on the relations where CHORUS and the expert-label disagree, our system is 1.6× more likely to be correct.

Table 3: Weighted F_1 scores for *table-class detection* on T2Dv2 dataset. Systems are compared with the expert-annotated classes for each table. The $n = 237$ tables each correspond to one of 33 `DBPedia.org` classes.

	Table-class correctness		
	F_1-score	Precision	Recall
DoDuo-Viz	0.654	66.8%	68.3%
DoDuo-Wiki	0.757	78.6%	76.9%
TaBERT	0.746	76.3%	76.8%
CHORUS	**0.926**	**96.1%**	**92.4%**

4.2 Column-type annotation

Next, we compare the ability of our system to assign classes to table columns.

VIZNET is a collection of tables, extracted by the Sherlock [32] team from the VizNet repository [30] of data visualizations and open datasets. VizNet comprises 31 million datasets in total. We selected 10 mutually exclusive DBPedia.org classes to test: `artist`, `address`, `city`, `director`, `album`, `currency`, `product`, `isbn`, `manufacturer`, `birthday`. We then used stratified sampling to select 1000 columns of these types to predict.

Baselines. We compare against TaBERT [69], DoDuo [57] and Sherlock [32] on this task. Since Sherlock is designed for column annotation, we use the out-of-the-box model provided by the original team. We restrict Sherlock to the ten target classes. For TaBERT we train an additional classification layer on top of the pre-trained embeddings that these frameworks provide. DoDuo provides a classification layer, however unlike Sherlock the current API does not expose raw probabilities, so we cannot restrict it to the target classes. Instead, we chose to add a multilayer perceptron classification layer, as for TaBERT, to give DoDuo fair odds. We fix the embeddings to their pretrained values and learn the weights of the classification layer using five-fold cross-validation.

Results. Table 5 contains the results for the VIZNET dataset. Our FM-based approach improves performance on the measured metrics of F_1-score, precision and recall. The best performing method is Sherlock, narrowly beating DoDuo-VizNet, with a 0.930 F_1 score. If we consider methods which are not specifically pretrained on VizNet (note, which is also the test set) CHORUS is the best performing on all three metrics. It has comparable F_1 and precision to Sherlock, but 6 percentage points lower recall.

Note in particular DoDuo-Wiki, which does not have access to VizNet at pretraining time, has a large regression in performance compared to DoDuo-Viznet, losing 0.085 F_1 points. We sanity-checked the low scores of TaBERT by replicating its previously reported [38] annotation scores from prior work and were successful.

4.3 Join-column prediction

Finally, we evaluate our approach's ability to suggest which columns are the correct choice for a join, the join-column prediction task. We use the *GitNotebooks* dataset from [68], a collection of 4 million

Table 4: *Outperforming expert-annotated labels*: all examples where our system exceeds human performance for table-class detection on the T2Dv2 dataset.

ID	Chorus label	Expert label	Evidence and explanation
7	Military-Person	Monarch	Table includes the values "emperor," "general" and "rebel"
9	Roman-Emperor	Monarch	Page titled *Crococalana History*, a Roman settlement
37	Temple	Monarch	Text on page: "Das sind vor allem ägyptische Tempel" translates to "These are mainly Egyptian temples"
47	Dam	Lake	Table is from `swiss-dams.ch`
65	Roman-Emperor	Monarch	Page description is *Roman Rulers*
103	Skyscraper	Building	Page titled *The Tallest Buildings*
111	Person	Scientist	Famous figures including "Walt Disney"
113	Nobel-Prize	Scientist	*Chemistry Nobel* page has one row for each year, including "Not awarded this year"
130	Currency	Country	Table titled *Currency Exchange Rates*
132	Political-Party	Election	Rows of *British Elections 2014* are political parties including "Labor" and "Tories"
136	Dam	Lake	Rows exclusively river dams *e.g.* "Tinaroo Falls Dam"
154	Song	Work	From URL `.../SongList.asp`
155	Fictional-Character	Television-Show	Rows for "Ryan Atwood" and "Caleb Nicol", characters from the TV show *The O.C.*
161	GolfTournament	Golf-Player	Column headers are "Year", "Winner" and "Second Place"
176	Roman-Emperor	Monarch	Page mentions ancient Rome, headers are "Emperor", "reign"
192	Roman-Emperor	Monarch	URL: `.../Rome/Emperors.html`
202	Mathematician	Scientist	Table from the *Mathematics Genealogy Project*
204	Nobel-Prize	Scientist	Nobel prizes list includes columns "Year", "Winner1", "Winner2" and "Winner3"
211	Political-Party	Election	Rows include "Labour Party" and "Conservative Party"
226	BodyOf-Water	Lake	Values include "Moose River", "Loon Stream" and "Alder Pond"
232	Religious-Organization	Country	Text on page: "The table above is generated [...] for each diocese"

Table 5: Weighted F_1 scores for *column-type annotation* on VizNet, with $n = 1000$ columns. Systems are compared with the "gold standard" classes for each column. Methods which are also pre-trained on VizNet are marked with an asterisk *.

	F_1-score	Precision	Recall
DoDuo-VizNet∗	0.900	90.3%	89.9%
Sherlock∗	**0.930**	**92.2%**	**93.1%**
TaBERT	0.380	38.9%	38.3%
DoDuo-Wiki	0.815	82.6%	81.4%
Chorus	**0.865**	**90.1%**	**86.7%**

Table 6: F_1 scores, precision and recall for the *join-column prediction* task on $n = 300$ tables.

	F_1-score	Precision	Recall
Jaccard	0.575	60.7%	54.7%
Levenshtein	0.718	72.3%	71.3%
Trifacta Wrangler	0.823	82.6%	82.0%
Chorus	**0.895**	**91.0%**	**88.0%**

Python notebooks (and their associated relational tables) including 24 thousand joins collected from Github. One of the baslines, Trifacta Wrangler, requires manual execution and recording of each prediction. For that reason we restrict this benchmark to 300 randomly sampled tables.

Baselines. For this task, we compare with three baselines. Jaccard similarity, J, is the first. Two columns are selected such that $\text{argmax}_{c \in C^T, c' \in C^{T'}} J(c, c')$ where $J(X, Y) = |X \cap Y| / |X \cup Y|$. This is a commonly used approach in the literature [12, 16, 48, 68]. Another baseline is Levenshtein distance [42], which selects the pair of column names with the smallest edit distance between them. The final baseline is Trifacta Wrangler [61], a commercial product spun off from the Wrangler research line [35]. When joining two tables in this product, it suggests the keys on which to join them. As no API was available, we obtain all Trifacta predictions by running the joins manually.

Results. Table 6 shows the quality of estimates for our approach and the baselines. We measure the quality of the predictions by the same criteria as the previous tasks. By these metrics, our approach improves the quality of predictions and beats the next-best approach by a clear margin: F_1 score is improved by 0.072, precision by 8.4 percentage points and recall by 6.0 percentage points.

4.4 Dataset contamination

Here we perform an experiment to validate whether any of the testing data occurred in the training corpus of the large-language model, an issue called *dataset contamination* or *data leakage*. Because these models are trained on internet data [23] and we use public benchmarks, they may have seen the test data in training.

Table 7: Guaranteed-unseen tables. Collected from the federal data lake Data.gov

Dataset Name
HMPS Truck Historical
AccessAbility Decal Recipients
DC Fast Charing Stations
Wind Resource Areas
DOHMH School Cafeteria Inspections
Public Passenger Vehicle Inspection Schedule
Product Data [...] in the Medicaid Drug Rebate Program

Table 8: Data contamination experiment. Weighted F_1 scores for *table-class detection* on public benchmarks versus tables the foundation model is guaranteed to have not been trained on.

	Table-class correctness		
Dataset	**F_1-score**	**Precision**	**Recall**
Public benchmark (VizNet)	0.865	90.1%	86.7%
Guaranteed-unseen	0.857	90.0%	81.8%

We test on seven guaranteed-unseen tables and their columns, all uploaded between April–June 2023 to the federal data repository Data.gov. They are guaranteed-unseen because the foundation model training was completed on or before March 2023, these datasets cannot have contaminated its corpus. Repeating the supervised column-type annotation task as in Section 4.2, we measure a 0.857 F_1 score, 90.0% precision and 81.8% recall. This is within 0.01 F_1 points, 0.1% precision and 5% recall of the benchmark results. These results are shown in Table 8. The drop being larger in recall is expected, as this reflects the datasets being more diverse and therefore difficult to classify (slightly decreased recall). However, the model's confidence in the correctness of its predictions remains constant (maintained precision), which we attribute to our anchoring technique.

4.5 System characteristics

Determinism. We examine the impact of nondeterminism in the foundation model on the performance of Chorus. The randomness of the generation is controlled by the *temperature* hyperparameter. To assure that the results of Chorus are reliable, we conduct the following experiment: we run the T2D table-class detection benchmark 25 times, five trials for each value of T between $0, 1/4, \ldots, 1$. Figure 8 shows the result. Chorus's performance is consistent: at the ideal temperature setting the F_1 score sees error bars of 0.01 F_1 points. Further, across all metrics we find the best performance at the lowest temperatures. A higher temperature is often desired for many natural language tasks: for example, in story generation it is important to not repeat stories for a given prompt and unpredictability of the output is key to generating interesting stories. However, in our setting, where a single best answer exists, determinism is more valuable. Therefore lower temperatures are more suitable in this setting for Chorus. Throughout the prior experiments, we have used the default temperature setting, as to get Chorus running with minimal hyperparameter tuning.

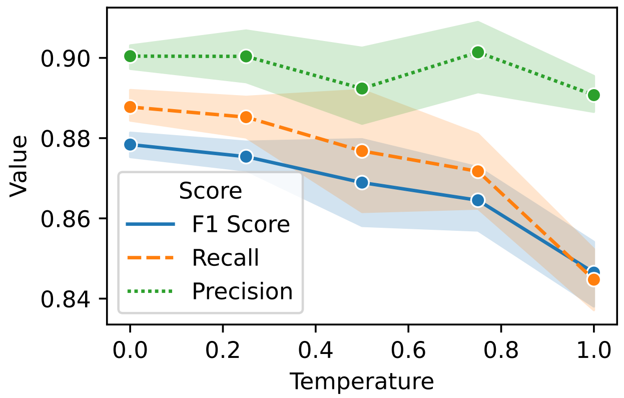


Figure 8: *Determinism vs. performance.* We conduct 25 runs of Chorus on the T2D table class benchmark. Shaded bands indicate confidence intervals. Temperature is a parameter controlling the randomness of the foundation model, with zero being the most (but not completely) deterministic.

Syntactic and semantic signals. We investigate whether Chorus's performance stems from its ability to extract signals from syntax features, from semantics features, or from both. Figure 9 shows the results of this experiment. We construct two column-type annotation tasks: (a) one which contains only a syntactic signal and (b) another which contains only a semantic signal. The results are shown in Figure 9.

Task (a) consists of classifying the national identification numbers of different countries. For example, the United States uses social security numbers with the format xxx-yyy-zzz, while China's resident identification numbers have the format of 17 consecutive numbers. No instruction is provided to the model about any of the national formats. Since we are using synthetic numbers, they carry no semantic signal—the only way to correctly classify them is to understand and parse the patterns. We synthesize 50 such numbers for five national formats: USA, China, UK, Czech Republic and Italy.

Subfigure 9a shows the results. The table in the subfigure shows a sample of the generated national IDs and Chorus' prediction. To demonstrate that the foundation model is able to represent the different syntax, we collect the activations of the final layer of the model for each input. We then visualize these high-dimensional datapoints in 2D using TSNE, a standard method. This shows clearly the clusters in activation patterns: numbers from the same country are tightly clustered together for each of the five nations. Chorus is able to correctly classify the labels and so must be able to parse syntactic signals in the data.

Task (b) consists of a purely semantic signal. It involves classifying politicians, actors, scientists and artists with English names. Since all names have the format `first-name last-name` the syntax provides no classification signal at all. Instead, any classification

Table 9: Alternative foundation models. Weighted F_1 scores for *table-class detection* on T2Dv2 dataset, for different choices of foundation model used by CHORUS. Parameter size in brackets. GPT-3.5 numbers identical to experiment in Figure 3.

	Table-class correctness		
Model choice	**F_1-score**	**Precision**	**Recall**
GPT-3.5 (175B)	0.926	96.1%	92.4%
LLaMA 2 (70B)	0.893	92.2%	86.5%
Vicuna/LLaMA (13B)	0.713	79.2%	64.1%
Vicuna/LLaMA (7B)	0.713	75.3%	67.5%

must rely on a semantic understanding of e.g. Louis Armstrong as a musician.

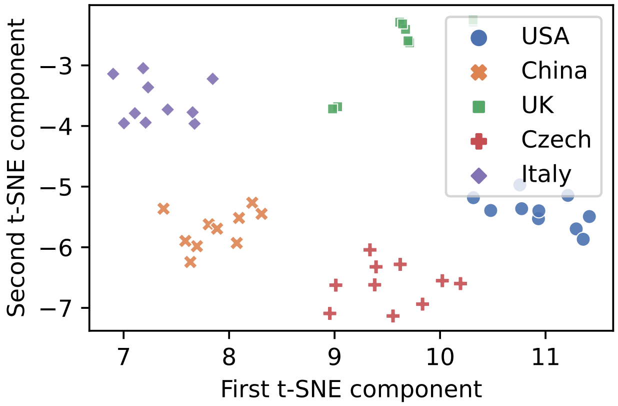
Subfigure 9b shows the result. We run CHORUS on ten entities each of the four classes. As with task (a), we outline sample outputs of CHORUS and plot the T-SNE visualization for the model activations for all data points. There is a clear clustering of the entities by their class type and therefore the system can also extract semantic signals. Given the results of the two experiments, we conclude that CHORUS utilizes a mixture of syntactic and semantic signals to perform data discovery tasks.

Alternative models. To demonstrate the versatility of this approach, we run CHORUS with three alternative, open-source foundation models on the table-class detection task. We consider Vicuna [74], a variant of LLaMA [59] at two sizes: 13 billion parameters and 7 billion parameters. The more advanced model is LLaMA 2 [60], the SOTA open-source model with 70 billion parameters.

Table 9 shows the results. While OpenAI's GPT model performs best, the best open-source model is very competitive. LLaMA 2 outperforms the best baseline model for this task—DoDuo-Wiki—by 0.136 F_1 points, on precision by 13.6 percentage points and on recall by 9.6 percentage points. This model lags behind the proprietary and larger GPT model by only a modest 0.03 F_1 points. This is a significant improvement over the older generation of LLaMA models, where the gap was a large 0.213 F_1 points. This rapid catch-up is remarkable considering the original LLaMA and LLaMA 2 were released 6 months apart. Open-source LLMs are now compelling alternatives on the tested task.

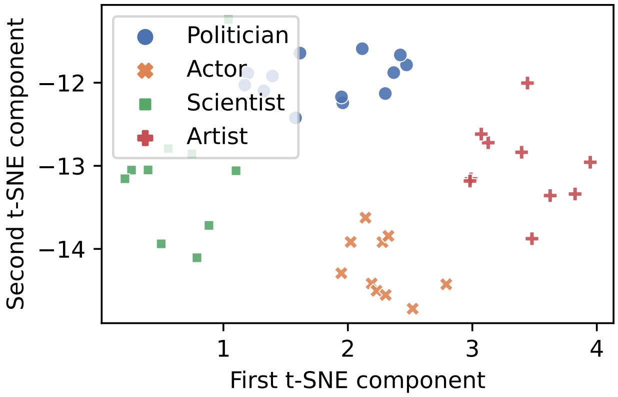
Ablations. We conduct ablation experiments to measure the contribution of individual components of CHORUS. We remove one component at a time and note the loss of scores compared to the unaltered model. Figure 10 shows the results. First, we remove the demonstration from the prompt. This results in an F1 score loss of 0.03, a recall loss of 4.7 and a precision loss of 4.7 percentage points. Next, we remove the metadata where it is available. This results in a cumulative F1 score loss of 0.04, a recall loss of 5.1 and a precision loss of 5.6 percentage points. After that, we disable anchoring. This results in a cumulative F1 score loss of 0.389, a recall loss of 47.4 and a precision loss of 31.9 percentage points. Finally, we remove the prefixes from the prompt. This results in a cumulative F1 score loss of 0.736, a recall loss of 92.3 and a precision loss of 53.1 percentage points.

National ID value	**CHORUS classification**
`610403197012096043`	China
`225130/7894`	Czech Republic
`865-50-6891`	United States
...	...



(a) Purely syntactic signal. National identification numbers (*e.g.* SSN) of various countries. All the IDs are valid but synthetic. These numbers can only be differentiated based on syntax, since they have no semantics.

Name	**CHORUS classification**
`Theodore Roosevelt`	Politician
`Barbara McClintock`	Scientist
`Louis Armstrong`	Artist
...	...



(b) Purely semantic signal. Names of politicians, actors, scientists and musicians can only be differentiated based on semantics, not syntax.

Figure 9: Semantic vs syntactic signals. CHORUS can extract both—for each task we provide a sample of classification outputs and also visualize t-SNE components of foundation model activations.

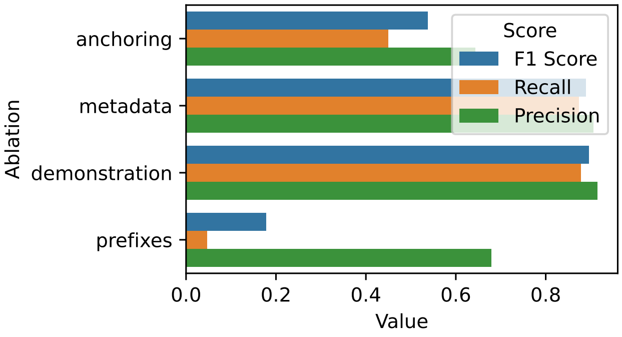


Figure 10: *Ablation experiments.* We ablate key features of Chorus and report performance characteristics on the T2D table-class detection task.

4.6 Summary

Our experiments show that chorus is a very promising approach for data-discovery tasks. The system is able to provide competitive performance on all three tested tasks. For ① table-class detection and ③ join-column detection, it surpasses all the baselines by a clear margin. On ② it performs best of the methods not pretrained specifically on the test dataset and comparably with those which are.

Importantly, the *performance is robust*: it consistently performs well, unlike other baseline methods. For example, TaBERT is the closest to the best performing baseline on T2Dv2 but the worst baseline on VizNet. Similarly DoDuo-Viz performs well on VizNet but loses 0.11 F_1 points when applied to T2Dv2 compared to DoDuo-Wiki. Conversely, DoDuo-Wiki loses 0.09 F_1 points compared to DoDuo-Viz when tested on the VizNet-based task.

Our study of the properties of chorus finds that its performance is not due to dataset contamination, that it extracts signal from both semantic and syntactic inputs, that it is compatible with alternative foundation models, and that it is robust against non-determinstic nature of foundation models.

5 DISCUSSION

We discuss some insights gleaned from the experiments on chorus and the baselines.

Training data collection. A major advantage of a foundation-model approach is that there is no need for training on specific tasks. In contrast, TaBERT requires 26 million tables for training its embeddings. Such representation learning approaches require large amounts of data for learning the embeddings as well as data for learning the task. Even with the embeddings fixed, lots of task-specific labels are needed. In [17], the use of 250 labels for one task is considered a "small dataset" by the authors and leads to subpar performance. In contrast, our prompts in Figure 5 use one example each for the table-class detection and column-type annotation and zero examples for the join-column prediction task.

Out-of-domain performance. We note a troubling pattern of a lack of cross-domain generalization in representation-learning approaches. In particular, the tested baselines degrade when used to embed tables not from the dataset the embeddings were trained on. Despite being described as "pretrained", this is in contrast to word embeddings such as word2vec, which perform consistently across a number of data distributions. This finding is in-line with prior work: regressions of up to 0.40 and 0.30 F_1 points when generalizing to new datasets have been observed by two other studies [17, 33].

Training costs. The aforementioned need to train prior approaches for each application domain is made worse by the computational cost of training the embeddings: tabert was trained by its authors using a cluster of 128 of Nvidia's Tesla V100 gpus [69, A.2], at the suggested retail price of $14000 *each* [14]. The cost of training fms is even greater, of course, but need only be borne once by the fm developer—not the end user; for chorus, the total cost was under $2, including development and experiments.

Flexibility. Another advantage of chorus we observe in the experiments is task adaptability. In the ① table-class detection task, we are able to switch the prediction domain easily. Restricting to the 33 classes used by the benchmark can be done by providing the permitted classes to the foundation model; allowing the model to generalize to other DBPedia classes (the *unsupervised* heading of Section 4.1) is as simple as omitting those instructions. Contextual information, such as table title or url, could be as easily added. In the representation learning setting, such modifications would require retraining the embeddings.

Limitations and risks. We control the risk of dataset contamination by testing for it in Section 4.4. The performance of chorus on guaranteed-unseen datasets is comparable to those in public benchmarks, so good performance on the those benchmarks cannot be explained away as simple data contamination. Separately, formal linguistic fluency means that errors may fool human reviewers. It is well known that linguistic fluency strongly influences the perception of correctness and quality [40, 41]. Foundation models may be able to obtain favorable evaluation by producing answers which are plausible to evaluators but incorrect. This has been called *subtle misinformation* in prior work [52]. Finally, we caution of large variances in result quality caused by minor prompt changes [73].

Future directions We outline ways in which this line of work may be expanded:

Additional tasks. The promising performance on our three tasks may extend to many more. Other tasks related to data discovery and exploration that could be explored include *schema autocompletion* [8], where missing parts of a partial schema are suggested to the user; *join-graph traversal*, where successive tables to join on are suggested [20]; and outlier detection, where potential erroneous data are detected.

Private or domain-specific datasets. As with all tested baselines, the foundation models are trained on public data. The distribution of data in the public sphere differs significantly from that in specialized domains or private data. It is worth investigating whether the observed capabilities continue to hold on e.g. enterprise data lakes. Further application to domain-specific ontology such as dron, a pharmaceutical ontology of drugs, would also be interesting to investigate. There is reason to think this will be challenging and require novel approaches. Namely foundation models virtually never recall facts that occur less than 10-100 times in the corpus [36]. This means that point lookups of data are unlikely to be supported by unassissted foundation models, at least for the foreseeable future. Further, we emphasize that many foundation models are trained less than one epoch, meaning they see each token in the corpus one time or not at all.

Larger and smaller models. Larger models are of interest because if scaling laws continue to hold, their performance should improve [66]. The larger GPT-4 model has been announced [7]. However, at the time of experiments the authors did not have API access to this model, so we leave exploring its capabilities to future work. On the other hand, the large cost of the current generation of foundation models has sparked interest in model distillation: reducing the size of the model without incurring a reduction in capabilities [18, 67]. However, these downsized models have significantly degraded emergent abilities [26] and so would likely perform worse on precisely the tasks of interest to us.

End-to-end evaluation. With foundation-model based data discovery systems becoming closer to reality, a holistic evaluation would be valuable to evaluate the utility of this approach to data analysts. This is important because many benchmarks are limited to sparsely-labelled portions of publicly available datasets.

6 RELATED WORK

The seminal early work in this area is WebTables [8]. WebTables aimed to extract relational tables from messy web data into a central repository, annotating them with predicted metadata to increase discoverability. This work introduced a constellation of related tasks: *schema auto-completion*, *attribute synonym finding*, and *join-graph traversal.* Additional work on *wrapper induction* [39] focused on developing shims to extracting table from heterogeneous sources.

The promise of foundation models for data profiling was outlined in a recent position paper [63]. This paper was based on evidence of foundation models being able to predict correlations in data from the column names [62]. Another work considered foundation models for data wrangling [46]: comprising the tasks of entity matching, error detection and data imputation. Finally, most recently foundation models have been applied to the classic problem of wrapper induction in the system EVAPORATE [4].

The currently deployed generation of approaches has focused on representation learning. These include TURL [17] and TABERT [69]. Both explore the use of fine-tuned language models for similar tasks. Other systems include Doduo [57] and TABBIE [34]. Prior to these table-embedding approaches, the prior generation of data tools involved deep learning for specific tasks. The standard-bearer for this approach is Sherlock [32], a tool specialized for column-type detection which utilizes a deep neural network with about 1 600 input features and trained over millions of examples. Sato [71] is another example of this approach.

Data discovery within data lakes is an active area of research in this community, with new tasks and approaches proposed regularly. Recent works related to this one include: unionability search [37], joinability search [75], new index structures for faster correlated dataset search [54] and end-to-end systems for data ingestion and profiling [10].

Recent tutorials [21, 47] outline the prevalence of the problem of unstructured document data management. A user-study of scientists conclude that "current systems fail to sufficiently support scientists in their data-seeking process" [51]. A dataset search survey [11] in VLDB, highlights the main open problems in this field: more natural query languages, better data integration, incorporating external knowledge, and interactive result presentation. Foundation models promise to help address many of these tasks.

Industry interest in this field is also keen. A large number of commercial solutions for data warehousing and data lakes are available. Commercial products derived from the original WebTables vision are described in the authors' follow-up paper [9]. Products including Amazon RedShift, Microsoft Azure Data Lake, Databricks Lakehouse are some of the commercial products in this space. Products with a more narrow focus, such as Trifacta Wrangler [61], Tableau Prep and Paxata, incorporate substantial data discovery components. Industry-led prototype systems for data discovery include Sigma Computing's WarpGate [13] and Google Research's Dataset Search [6]. More recently, startups such as Numbers Station raised substantial funding for fusing foundation models with enterprise data analytics. Gartner estimates the data warehouse market size at $22 billion USD. Governmental interest is also ramping up, with DARPA funding funding programs like KAIROS to process documents in multiple modalities and augment them with appropriate tags for identification, retrieval and linking.

7 CONCLUSION

In this work, we investigated foundation models for data discovery and exploration. We propose CHORUS, a system for integrating foundation models into data discovery tasks and show it provides superior performance on three example tasks: table-class annotation, column-type detection and join-column prediction. We find that CHORUS is more robust than prior representation-learning approaches on a variety of datasets and that its performance advantage cannot be attributed to dataset contamination. We conclude that foundation models provide a promising future as a core component of the next generation of data discovery and exploration systems.

ACKNOWLEDGMENTS

Our thanks to Yejin Choi for guidance on and productive discussions about language models. We are grateful to the authors of TaBERT, Sherlock and DoDuo for high-quality, replicable experiment code and documentation. We appreciate Cynthia Richey's and Kyle Deeds' assistance with the manuscript. Finally, thanks to Magdalena Balazinska for counsel during manuscript writing. This material is based upon work supported by the National Science Foundation under Grants NSF-BSF 2109922 and NSF IIS 2314527.

REFERENCES

[1] Nora Abdelmageed, Jiaoyan Chen, Vincenzo Cutrona, Vasilis Efthymiou, Oktie Hassanzadeh, Madelon Hulsebos, Ernesto Jiménez-Ruiz, Juan Sequeda, and Kavitha Srinivas. Results of semtab 2022. In Vasilis Efthymiou, Ernesto Jiménez-Ruiz, Jiaoyan Chen, Vincenzo Cutrona, Oktie Hassanzadeh, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, and Madelon Hulsebos, editors, *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2021, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference, October 23-27, 2022*, volume 3320 of *CEUR Workshop Proceedings*, pages 1–13. CEUR-WS.org, 2022. URL https://ceur-ws.org/Vol-3320/paper0.pdf.

[2] Inc. Anaconda. State of data science. https://www.anaconda.com/resources/whitepapers/state-of-data-science-2021, July 2021.

[3] Jacob Andreas. Language models as agent models, 2022.

[4] Simran Arora, Brandon Yang, Sabri Eyuboglu, Avanika Narayan, Andrew Hojel, Immanuel Trummer, and Christopher Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes, 2023.

[5] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

[6] Dan Brickley, Matthew Burgess, and Natasha F. Noy. Google dataset search: Building a search engine for datasets in an open web ecosystem. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1365–1375. ACM, 2019. doi: 10.1145/3308558.3313685. URL https://doi.org/10.1145/3308558.3313685.

[7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[8] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008. doi: 10.14778/1453856.1453916. URL http://www.vldb.org/pvldb/vol1/1453916.pdf.

[9] Michael J. Cafarella, Alon Y. Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. Ten years of webtables. *Proc. VLDB Endow.*, 11(12):2140–2149, 2018. doi: 10.14778/3229863.3240492. URL http://www.vldb.org/pvldb/vol11/p2140-cafarella.pdf.

[10] Sonia Castelo, Rémi Rampin, Aécio S. R. Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. Auctus: A dataset search engine for data discovery and augmentation. *Proc. VLDB Endow.*, 14(12):2791–2794, 2021. doi: 10.14778/3476311.3476346. URL http://www.vldb.org/pvldb/vol14/p2791-castelo.pdf.

[11] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. Dataset search: a survey. *VLDB J.*, 29(1):251–272, 2020. doi: 10.1007/s00778-019-00564-x. URL https://doi.org/10.1007/s00778-019-00564-x.

[12] Zhimin Chen, Vivek R. Narasayya, and Surajit Chaudhuri. Fast foreign-key detection in microsoft SQL server powerpivot for excel. *Proc. VLDB Endow.*, 7 (13):1417–1428, 2014. doi: 10.14778/2733004.2733014. URL http://www.vldb.org/pvldb/vol7/p1417-chen.pdf.

[13] Tianji Cong, James Gale, Jason Frantz, H. V. Jagadish, and Çagatay Demiralp. Warpgate: A semantic join discovery system for cloud data warehouses. *CoRR*, abs/2212.14155, 2022. doi: 10.48550/arXiv.2212.14155. URL https://doi.org/10.48550/arXiv.2212.14155.

[14] SHI International Corp. Nvidia tesla v100 - gpu computing processor, 2018. URL https://web.archive.org/web/20230304032322/https://www.shi.com/product/34625444/NVIDIA-Tesla-V100-GPU-computing-processor.

[15] District Court. Mata v. avianca, inc. (1:22-cv-01461). Southern District of New York, New York, June 2023.

[16] Tamraparni Dasu, Theodore Johnson, S. Muthukrishnan, and Vladislav Shkapenyuk. Mining database structure; or, how to build a data quality browser. In Michael J. Franklin, Bongki Moon, and Anastassia Ailamaki, editors, *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, pages 240–251. ACM, 2002. doi: 10.1145/564691.564719. URL https://doi.org/10.1145/564691.564719.

[17] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL: table understanding through representation learning. *Proc. VLDB Endow.*, 14(3):307–319, 2020. doi: 10.5555/3430915.3442430. URL http://www.vldb.org/pvldb/vol14/p307-deng.pdf.

[18] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[19] Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1286–1305. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.98. URL https://doi.org/10.18653/v1/2021.emnlp-main.98.

[20] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. Deepjoin: Joinable table discovery with pre-trained language models. *CoRR*, abs/2212.07588, 2022. doi: 10.48550/arXiv.2212.07588. URL https://doi.org/10.48550/arXiv.2212.07588.

[21] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. Table discovery in data lakes: State-of-the-art and future directions. In Sudipto Das, Ippokratis Pandis, K. Selçuk Candan, and Sihem Amer-Yahia, editors, *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 69–75. ACM, 2023. doi: 10.1145/3555041.3589409. URL https://doi.org/10.1145/3555041.3589409.

[22] Common Crawl Foundation. Common crawl, 2011. URL https://commoncrawl.org.

[23] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021. URL https://arxiv.org/abs/2101.00027.

[24] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3356–3369. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.301. URL https://doi.org/10.18653/v1/2020.findings-emnlp.301.

[25] Mike Gualtieri and Noel Yuhanna. *The Forrester Wave: Big Data Hadoop Distributions, Q1 2016.* Forrester Research, Inc., January 2016.

[26] Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms, 2023.

[27] Will Douglas Heaven. Why meta's latest large language model survived only three days online. *MIT Technology Review*, November 2022. URL https://www.technologyreview.com/2022/11/18/1063487/meta-large-language-model-ai-only-survived-three-days-gpt-3-science/.

[28] P. Bryan Heidorn. Shedding light on the dark data in the long tail of science. *Library trends*, 57(2):280–299, 2008.

[29] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

[30] Kevin Hu, Neil Gaikwad, Michiel Bakker, Madelon Hulsebos, Emanuel Zgraggen, César Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çağatay Demiralp. Viznet: Towards a large-scale visualization learning and benchmarking repository. In *Proceedings of the 2019 Conference on Human Factors in Computing Systems (CHI)*. ACM, 2019.

[31] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

[32] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1500–1508. ACM, 2019. doi: 10.1145/3292500.3330993. URL https://doi.org/10.1145/3292500.3330993.

[33] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. Gittables: A large-scale corpus of relational tables. *CoRR*, abs/2106.07258, 2021. URL https://arxiv.org/abs/2106.07258.

[34] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. TABBIE: pretrained representations of tabular data. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3446–3456. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.270. URL https://doi.org/10.18653/v1/2021.naacl-

main.270.

[35] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Wrangler: interactive visual specification of data transformation scripts. In Desney S. Tan, Saleema Amershi, Bo Begole, Wendy A. Kellogg, and Manas Tungare, editors, *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 3363–3372. ACM, 2011. doi: 10.1145/1978942.1979444. URL https://doi.org/10.1145/1978942.1979444.

[36] Nikhil Kandpal, Eric Wallace, and Colin Raffel. Deduplicating training data mitigates privacy risks in language models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10697–10707. PMLR, 2022. URL https://proceedings.mlr.press/v162/kandpal22a.html.

[37] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. SANTOS: relationship-based semantic table union search. *Proc. ACM Manag. Data*, 1(1):9:1–9:25, 2023. doi: 10.1145/3588689. URL https://doi.org/10.1145/3588689.

[38] Aneta Koleva, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. Generating table vector representations. *CoRR*, abs/2110.15132, 2021. URL https://arxiv.org/abs/2110.15132.

[39] Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 729–737. Morgan Kaufmann, 1997.

[40] Ellen J Langer, Arthur Blank, and Benzion Chanowitz. The mindlessness of ostensibly thoughtful action: The role of" placebic" information in interpersonal interaction. *Journal of personality and social psychology*, 36(6):635, 1978.

[41] David Langford. Comp.basilisk faq. *Nature*, 402(6761):465–465, 1999. doi: 10.1038/44964. URL https://doi.org/10.1038/44964.

[42] Vladimir Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet Physics Doklady*, volume 10, page 707, 1966.

[43] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229.

[44] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories, 2022.

[45] Pablo N. Mendes, Max Jakob, and Christian Bizer. Dbpedia: A multilingual cross-domain knowledge base. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 1813–1817. European Language Resources Association (ELRA), 2012. URL http://www.lrec-conf.org/proceedings/lrec2012/summaries/570.html.

[46] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. Can foundation models wrangle your data? *Proc. VLDB Endow.*, 16(4):738–746, 2022. URL https://www.vldb.org/pvldb/vol16/p738-narayan.pdf.

[47] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. Data lake management: Challenges and opportunities. *Proc. VLDB Endow.*, 12(12):1986–1989, 2019. doi: 10.14778/3352063.3352116. URL http://www.vldb.org/pvldb/vol12/p1986-nargesian.pdf.

[48] Fatemeh Nargesian, Ken Q. Pu, Bahar Ghadiri Bashardoost, Erkang Zhu, and Renée J. Miller. Data lake organization. *IEEE Trans. Knowl. Data Eng.*, 35(1): 237–250, 2023. doi: 10.1109/TKDE.2021.3091101. URL https://doi.org/10.1109/TKDE.2021.3091101.

[49] Washington State Department of Licensing. Electric vehicle population data electric vehicle population data, 04 2023. URL https://catalog.data.gov/dataset/electric-vehicle-population-data.

[50] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *CoRR*, abs/2203.02155, 2022. doi: 10.48550/arXiv.2203.02155. URL https://doi.org/10.48550/arXiv.2203.02155.

[51] Andrea Papenmeier, Thomas Krämer, Tanja Friedrich, Daniel Hienert, and Dagmar Kern. Genuine information needs of social scientists looking for data. *Proceedings of the Association for Information Science and Technology*, 58(1):292–302, 2021.

[52] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446.

[53] Dominique Ritze and Christian Bizer. Matching web tables to dbpedia - A feature utility study. In Volker Markl, Salvatore Orlando, Bernhard Mitschang, Periklis Andritsos, Kai-Uwe Sattler, and Sebastian Breß, editors, *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, pages 210–221. OpenProceedings.org, 2017. doi: 10.5441/002/edbt.2017.20. URL https://doi.org/10.5441/002/edbt.2017.20.

[54] Aécio S. R. Santos, Aline Bessa, Christopher Musco, and Juliana Freire. A sketch-based index for correlated dataset search. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*, pages 2928–2941. IEEE, 2022. doi: 10.1109/ICDE53745.2022.00264. URL https://doi.org/10.1109/ICDE53745.2022.00264.

[55] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. *CoRR*, abs/2302.00093, 2023. doi: 10.48550/arXiv.2302.00093. URL https://doi.org/10.48550/arXiv.2302.00093.

[56] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Schärli, Aakanksha Chowdhery, Philip Andrew Mansfield, Blaise Agüera y Arcas, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *CoRR*, abs/2212.13138, 2022. doi: 10.48550/arXiv.2212.13138. URL https://doi.org/10.48550/arXiv.2212.13138.

[57] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. Annotating columns with pre-trained language models. In *Proceedings of the 2022 International Conference on Management of Data*. Association for Computing Machinery, 2022. ISBN 9781450392495. URL https://doi.org/10.1145/3514221.3517906.

[58] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *CoRR*, abs/2211.09085, 2022. doi: 10.48550/arXiv.2211.09085. URL https://doi.org/10.48550/arXiv.2211.09085.

[59] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/arXiv.2302.13971. URL https://doi.org/10.48550/arXiv.2302.13971.

[60] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/arXiv.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

[61] Trifacta. Trifacta wrangler. https://cloud.trifacta.com, 2023. Accessed: 2023-04-10.

[62] Immanuel Trummer. Can deep neural networks predict data correlations from column names? *CoRR*, abs/2107.04553, 2021. URL https://arxiv.org/abs/2107.04553.

[63] Immanuel Trummer. Towards nlp-enhanced data profiling tools. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org, 2022. URL https://www.cidrdb.org/cidr2022/papers/a55-trummer.pdf.

[64] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. Technical Report MSR-TR-2023-8, Microsoft, February 2023. URL https://www.microsoft.com/en-us/research/

publication/chatgpt-for-robotics-design-principles-and-model-abilities/.

[65] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[66] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022. URL https://openreview.net/forum?id=yzkSU5zdwD.

[67] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D. Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 4602–4625. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.341. URL https://doi.org/10.18653/v1/2022.naacl-main.341.

[68] Cong Yan and Yeye He. Auto-suggest: Learning-to-recommend data preparation steps using data science notebooks. In David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo, editors, *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1539–1554. ACM, 2020. doi: 10.1145/3318464.3389738. URL https://doi.org/10.1145/3318464.3389738.

[69] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.745. URL https://doi.org/10.18653/v1/2020.acl-main.745.

[70] Ce Zhang, Jaeho Shin, Christopher Ré, Michael J. Cafarella, and Feng Niu. Extracting databases from dark data with deepdive. In Fatma Özcan, Georgia Koutrika, and Sam Madden, editors, *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 847–859. ACM, 2016. doi: 10.1145/2882903.2904442. URL https://doi.org/10.1145/2882903.2904442.

[71] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çagatay Demiralp, and Wang-Chiew Tan. Sato: Contextual semantic type detection in tables. *Proc. VLDB Endow.*, 13(11):1835–1848, 2020. URL http://www.vldb.org/pvldb/vol13/p1835-zhang.pdf.

[72] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball, 2023.

[73] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 2021. URL http://proceedings.mlr.press/v139/zhao21c.html.

[74] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685, 2023. doi: 10.48550/arXiv.2306.05685. URL https://doi.org/10.48550/arXiv.2306.05685.

[75] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. JOSIE: overlap set similarity search for finding joinable tables in data lakes. In Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 847–864. ACM, 2019. doi: 10.1145/3299869.3300065. URL https://doi.org/10.1145/3299869.3300065.