

THE UNIVERSITY OF CHICAGO

LLM-AS-A-CROWD: HOW TO USE LLMS FOR DATA INTEGRATION TASKS

AN M.S. PAPER SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

BY
PRANAV SUBRAMANIAM

CHICAGO, ILLINOIS

SEPTEMBER 22, 2023

Copyright © 2023 by Pranav Subramaniam

All Rights Reserved

TABLE OF CONTENTS

LIST OF FIGURES	iv
LIST OF TABLES	v
ACKNOWLEDGMENTS	vi
ABSTRACT	vii
1 INTRODUCTION	1
2 THE OPPORTUNITY OF LLMS	5
2.1 Data Integration and Semantic Ambiguity	5
2.2 Why LLMS for Data Integration?	6
2.3 Challenges to Leveraging LLMS	8
3 LLM AS A CROWD	9
3.1 A Primer on Crowd Truth Inference	9
3.2 Eliciting Crowds from LLMS	11
4 LLMS IN DATA INTEGRATION PIPELINES	14
4.1 Entity Resolution	14
4.2 Joinability and Unionability	17
5 EVALUATION	18
5.1 RQ1. How do we use LLMS?	18
5.2 RQ2. How do LLMS and the state-of-the-art compare?	28
5.3 RQ3. Generalization to other Tasks	32
6 RELATED WORK	35
7 CONCLUSIONS	37
REFERENCES	38

LIST OF FIGURES

1.1	ChatGPT handles semantic ambiguity well	2
2.1	ChatGPT solves an Entity Resolution instance.	6
2.2	Illustration of input/output procedures to: i) transform prompt template and instance into a prompt; and ii) transform output into an answer to the instance. The functions (blue background) are explained later.	7
4.1	ChatGPT Joinability Example	16
4.2	ChatGPT Unionability Example	17
5.1	Comparison of LLM results with various prompt engineering strategies. X-axis is the benchmark. Y-axis is the F1 score (starts at 30). (a) Comparison of LLM results with and without few-shot learning and crowd truth inference on all datasets. (b) Comparison of LLM results with and without multiple repetitions at high temperature on LSPM benchmarks. (c) Comparison of LLM results with and without chain-of-thought prompting on Products benchmark.	20
5.2	An example of a common false negative failure mode when using ChatGPT CoT (chain-of-thought): the model is fixated on the many non-meaningful differences.	23
5.3	(left) Performance of various workers and crowd methods. Y-axis is the average F1 score. (a) Results for our standard crowd used in ChatGPT Crowd + Few-shot, averaged across all datasets. Error bars show the maximum and minimum dataset F1 scores. (b) Results from a crowd formed from various prompt engineering techniques - Basic + Few-shot, Basic, CoT + Few-shot, and CoT.	24
5.4	Partial correlations (controlling for the true label as a covariate) between the prompt templates of our default crowd across all datasets. (a) ChatGPT. (b) ChatGPT + Few-shot.	26
5.5	Comparison of LLM results with in- and out-of-distribution few-shot examples. X-axis is the benchmark. Y-axis is the F1 score (starts at 50).	27
5.6	(left) Comparisons of Ditto and LLM techniques. X-axis is the benchmark. Y-axis is the F1 score (starts at 40). (a) Comparison of techniques when available training data are out-of-distribution (the training data is from LSPM cameras). (b) Comparison of techniques on variants of the Products dataset that require more or less generalization.	29

LIST OF TABLES

5.1	Overview of benchmarks for each data integration task.	19
5.2	Top 3 prompt templates (by F1 score) from our default crowd (with and without Few-shot).	25
5.3	Performance (F1 score) of Ditto, Finetuned Ada, and ChatGPT on the entity resolution benchmarks, with access to the training sets of various sizes. Numbers for ChatGPT are repeated as ChatGPT always draws its few-shot examples from the small training set. Bold numbers are the highest F1 score for the given size and benchmark. We highlight Products as we believe it to be the most informative benchmark for entity resolution.	31
5.4	Precision of LLM Methods vs SANTOS, by Dataset.	34

ACKNOWLEDGMENTS

I would first like to thank my advisor Prof. Raul Castro Fernandez. I admire his approach to research problems, and his clear thought process. He has been a consistent source of thoughtful guidance. I am also grateful for Prof. Aaron Elmore and Prof. Sanjay Krishnan for devoting their time and energy in being on my committee. I express gratitude towards Zachary Hempstead for his exceptional assistance on this project.

ABSTRACT

Large Language Models (LLMs) are capable of answering questions without task-specific training data, which creates an opportunity to address data integration tasks such as entity resolution, joinability, and unionability. Solving these tasks requires incorporating semantic knowledge, which has been a hard-to-address challenge. LLMs exhibit a tremendous capacity to understand nuance in language and thus are promising for this task. However, using them requires addressing two challenges: i) how to query them to obtain valid answers, i.e., prompt engineering; ii) how to best incorporate them along today’s software pipelines.

In this paper, we study the potential of LLMs for data integration tasks. We find that thinking of *LLMs-as-a-crowd* is a useful mental model to leverage them for data integration tasks because it results in high quality results without any access to training data—which many state of the art methods rely on—and without any dependence on good prompt engineering skills. We integrate LLMs into software pipelines that leverage the vast research in the areas of entity resolution, joinability, and unionability. We find that LLMs are effective as an aid—but not a replacement—to software integration pipelines, thus effectively building on previous efforts.

We obtain state of the art results for the three tasks we study, entity resolution, joinability, and unionability. And we demonstrate that thinking of LLMs-as-a-crowd is effective and complementary to other methods, such as *few-shot* learning. All in all, our experimental evaluation paves the way for further study of the use of LLMs for data integration tasks.

CHAPTER 1

INTRODUCTION

Integrating data to solve tasks is as important as ever but remains hard. A major source of difficulty continues to be the need for agreeing on “semantics”, which is necessary to perform data integration tasks, such as determining if two entities represent the same entity (*entity resolution*), or if two tables can be combined (*joinability* and *unionability*). Hardcoded rules were not sufficient to resolve semantically ambiguous tasks, due to the vast variety of these [27, 22]. To attack this problem, there have been attempts to use external reference data, such as ontologies and knowledge bases [8, 34, 32, 2], and to use pretrained models [46, 39, 20]. Still, despite tremendous progress, data integration is not solved. Many of the existing methods that achieve state of the art results rely on access to large training datasets, which require expensive human labor. As a result, solutions require large human effort to disambiguate meaning and make progress. While humans must arguably be involved in data preparation, overusing human time is expensive and hampers productivity.

Large Language Models (LLMs) such as ChatGPT exhibit an impressive ability to address complex semantic tasks without access to *task-specific training data*; they are trained once, and perform well on many tasks they were not explicitly trained to perform. For example, LLMs excel at NLP tasks such as closed-book question answering, text classification, and information extraction [4, 48] while only being trained to build a joint probability distribution over token sequences. Such NLP tasks resemble the core challenges of *entity resolution*, *joinability*, and *unionability*, tasks needed for integration. As an example, consider the LLM query and abridged response in Figure 1.1.

The success of LLMs on complex NLP tasks (and our preliminary analyses) suggests they introduce a new opportunity to address semantic-laden data integration tasks.

In this paper, we study the ability of LLMs to address entity resolution (*do two entities refer to the same one?*), joinability (*do two tables join on these attributes?*), and unionability

user: Consider the following two phrases:

"Eric Washington"

"Springfield, Washington"

Does "Washington" refer to the same thing?

model: In the context of the given phrases, "Washington" does not refer to the same thing.

In the phrase "Eric Washington," "Washington" most likely refers to a person's surname or last name. [...]

On the other hand, in the phrase "Springfield, Washington," "Washington" refers to a location. [...]

Although both instances include the word "Washington," their meanings differ depending on the context in which they are used. [...]

Figure 1.1: ChatGPT handles semantic ambiguity well (*do two tables represent the same domains?*), which are important tasks that aid in many data integration tasks. In our study, we run into two key challenges to using LLMs effectively and propose solutions to both:

Challenge 1: *Prompt Engineering is brittle and unreliable.* We follow the emerging rules of thumb for querying LLMs, commonly referred to as “prompt engineering”, and rediscover—in the context of the target data integration tasks—what others have said before [47, 61]: that prompt engineering is brittle to changes in prompt, data, and model. We find that the intuition one develops in producing useful prompts for one task does not transfer easily to other tasks. Besides, there is no guarantee that such intuition transfers across different versions of data and models. Such brittleness translates in shifting human labor from annotating training data to preparing prompts, thus negating the opportunity of LLMs.

Approach 1: *Think of an LLM as a Crowd.* Our observation is that changes in the input prompts elicit different answers from LLMs, some of which are correct for a given task. Yet, prompts that work for one task do not necessarily work well for other tasks. Then, rather than investing time in selecting a task-specific prompt whose performance will not

transfer to other tasks, the key idea is to consider the LLM as a crowd that produces N answers per instance (via N different prompts) and then uses truth-finding techniques to aggregate the answers across multiple instances. Using a crowd avoids forcing engineers to make an uninformed decision on which prompt to use. Thinking of the LLM as a crowd has one advantage and one disadvantage. The advantage is that it boosts tasks’ performance without investing time in doing prompt engineering, and without access to training data, thus addressing Challenge 1. The disadvantage is that it requires ($N > 1$) queries to the LLM per task instance, resulting in higher cost and runtime. We believe that the performance advantage and resulting human time saved compensates for the disadvantage, and the tradeoff will become more beneficial because of ongoing trends to make LLMs faster, cheaper, and more environmentally friendly.

Challenge 2: *Given their current technical limitations, LLMs will not replace current tools for data integration because state-of-the-art tools solve problems that would be too expensive and inefficient to address using LLMs.* For example, the blocking stage of an entity matcher is in charge of identifying candidates for entity resolution, and thus requires exploring many pairs of entities. Consider the following example. Using ChatGPT as a blocker for two (tiny!) 5000-row datasets, requires 25 million API calls and would cost around \$9.5k at ChatGPT’s current prices (\$0.0015 for 1000 input tokens), assuming 250 tokens per request. There would also be substantial latency—OpenAI currently limits organizations to 90k tokens per minute, so this would take around 48 days at a minimum. While market forces are pushing LLMs to become cheaper, we anticipate that the better-performing LLMs will remain large and expensive in the short term. The challenge, then, is in deciding how to best make use of LLMs for different data integration pipelines.

Approach 2: *Make LLM work alongside traditional software pipelines.* We propose an integration of LLMs into data integration pipelines that reduces cost while boosting performance. The integration deals with the problem of adapting existing data into input

LLM prompts, and adapting output LLM prompts into answers. We propose a pipeline for each of the 3 target tasks and we discuss some pitfalls and lessons learned along the way.

We conduct a thorough evaluation to explore the ability of LLMs to aid in data integration tasks, which we believe is one of the paper’s main contributions. We conclude:

- Without access to training data, LLMs significantly boost the performance on data integration tasks such as entity resolution, joinability, and unionability, and achieve state of the art results. Even in the most favorable setting for state of the art methods (access to large training data and evaluated on in-distribution data), LLMs’ performance match that of those methods. This suggests that LLMs are useful to address many data integration tasks with minimum upfront cost.
- Treating the LLM as a crowd removes the need for careful (i.e., time-consuming) prompt engineering while boosting the performance across different data integration tasks. We show how prompt performance varies unpredictably across tasks, and how crowd methods are effective in selecting the right prompt at the right time.
- Of all popular prompt engineering techniques, few-shot learning is effective, and its effects compound with those of the crowd. On the other hand, Chain of Thought reasoning and self-consistency are not effective for data integration tasks.

Overall, the results of this study underscore the opportunity of integrating LLMs with data integration pipelines. The results also quantify the remaining challenges, including the need to reduce the cost of LLMs and identify better ways of eliciting crowds from LLMs, even when the models themselves are practically black boxes.

The rest of the paper is organized as follows. We present a background on LLMs in Chapter 2. We present the LLM-as-a-crowd in Chapter 3 to address **Challenge 1**, the integration with data integration tasks in Chapter 4 to address **Challenge 2**. And the experimental evaluation in Chapter 5. We conclude with related work (Chapter 6) and Conclusions.

CHAPTER 2

THE OPPORTUNITY OF LLMS

In this section, we explain why LLMs are promising for data integration tasks.

2.1 Data Integration and Semantic Ambiguity

In this paper, we use the phrase *data integration task* to refer to one of {entity resolution, joinability, unionability} and we use *task instance* to refer to an atomic example, e.g. whether two particular tables are joinable, or whether two rows refer to the same entity.

Consider the following entity resolution instance:

- (i) *title: WD Red NAS 3.5 Internal HDD SATA 6Gb/s, 2TB 6Gb/s 2TB - WD | CPC*
- (ii) *title: Buy Online | WD Red WD20EFRX 2TB Desktop Internal Hard Drive Price in India.*

A human may recognize the brand (Western Digital) and that 2TB refers to a storage unit, linking it to the *hard drive* keywords of the second row and concluding the rows refer to the same entity. But it is difficult to incorporate that *semantic knowledge* into a machine. Such knowledge is useful for many other integration tasks, including determining join keys, whether tables join, etc.

To incorporate semantic knowledge, existing work has used reference data such as ontologies and knowledge bases [34, 41, 2], different combinations of *signals* [46, 32, 5, 59, 8], and, more recently, pre-trained models such as word2vec, BERT, and more. Pre-trained models have been used for integration tasks such as unionability/joinability [46, 12, 18, 20], entity resolution [39, 6, 45] and more [27, 60]. These pre-trained models generally require fine-tuning before the systems that use them can perform well on the given task: this poses a barrier to their adoption because of i) the cost of collecting training data; ii) their dependence on training data to achieve high performance (as we show in the evaluation).

2.2 Why LLMs for Data Integration?

In contrast to existing efforts to deal with semantic knowledge in data integration, recent large language models (LLMs) do not need any training data or fine-tuning to solve data integration task instances, which is an important advantage. For example, in Fig. 2.1 we query ChatGPT on the task instance from Section 2.1. Not only does it give the correct answer, it follows up with a reasonable-sounding explanation. It is this ability of LLMs to answer questions without access to training data that makes them so attractive to data integration tasks, and that motivates our work. Note that we transform the instance into a text sequence that contains the instance itself along with additional context, such as a question and an instruction about the desired answer format. We refer to this text derived from a task instance as a *data integration prompt*.

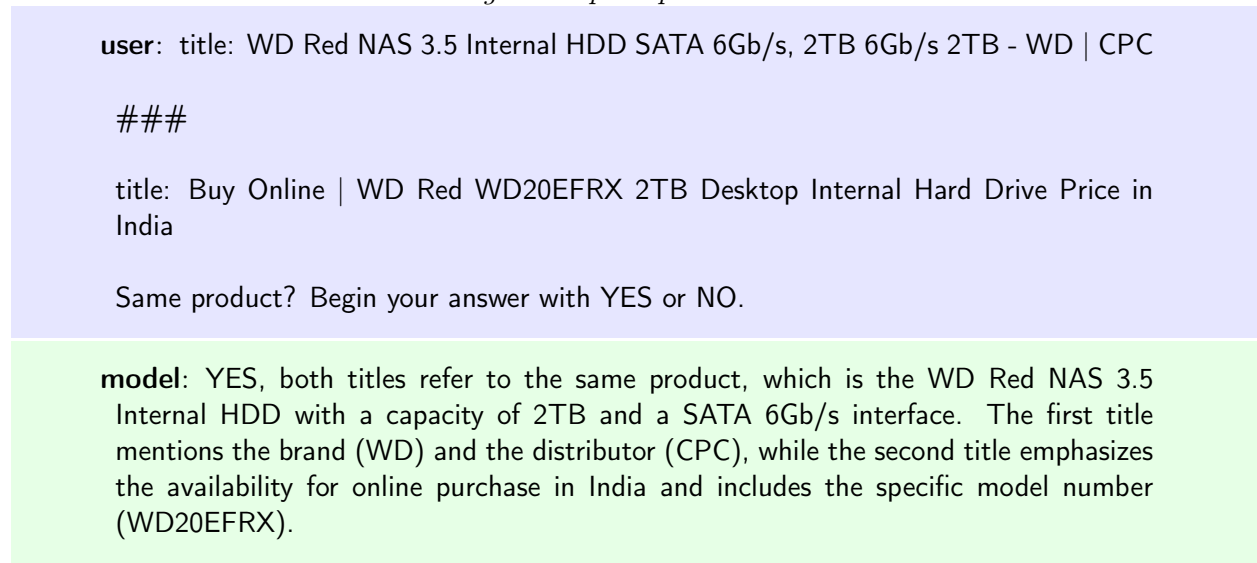


Figure 2.1: ChatGPT solves an Entity Resolution instance.

Language Model Overview. A *language model* (LM) assigns probabilities to sequences of text, which are split into discrete "tokens" that include words, parts of words, whitespace, and so on. Formally, a LM takes a sequence of tokens x_1, \dots, x_n as input and outputs joint probability $P(t_1 = x_1, t_2 = x_2, \dots, t_n = x_n)$ (probability that the first token is x_1 , second token is x_2 , etc.). It is common to calculate this by computing conditional probabilities

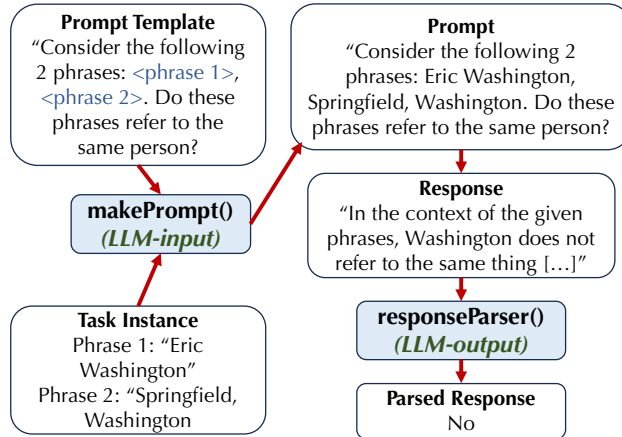


Figure 2.2: Illustration of input/output procedures to: i) transform prompt template and instance into a prompt; and ii) transform output into an answer to the instance. The functions (blue background) are explained later.

$P(t_i|t_1, \dots, t_{i-1})$ for all $1 \leq i \leq n$ and then using the chain rule of conditional probabilities to compute the full probability [29]. Then, given an input sequence x_1, \dots, x_k , such LMs perform *text generation* by sampling the joint distribution conditioned on x_1, \dots, x_k to generate x_{k+1}, \dots, x_n . The type of sampling used is referred to as a *decoding strategy* [24, 25]. Informally, the class of recent language models that are able to perform well on a diverse array of tasks without task-specific training data are referred to as *large language models* (LLMs); examples include the GPT family of models. These models generally have vastly more parameters and were trained on much larger text corpora than older language models.

How does an LLM solve an integration prompt?. An LLM takes an input sequence of tokens that represents the *prompt*, \vec{w}_{in} and produces a sequence of tokens in response, \vec{w}_{out} . Since \vec{w}_{out} is unstructured, it is necessary to post-process it with some function g into an answer to the task, $g(\vec{w}_{out})$. Different input prompts produce different outputs. Crafting the input prompt so the output contains the answer to the task is referred to as *prompt engineering*, and we refer to a particular mapping of instance to prompt as a *prompt template*. We illustrate this mapping in Fig. 2.2 and describe this process in more detail in Chapter 4. LLMs subtly differentiate word meaning and do not rely on training data to do so: this makes them attractive for data integration.

2.3 Challenges to Leveraging LLMs

Two hurdles complicate using LLMs for data integration tasks.

Challenge 1. Prompt Engineering is brittle and unreliable. An input prompt may produce different output when the model or the query-specific tokens, e.g., those representing a row in a table, change. A prompt that performs well in one query may perform badly on another one. This means that to obtain good performance across varied tasks and queries, humans must be involved to do prompt engineering. And there are many techniques a human must try, ranging from Chain-of-thought reasoning [56] to least-to-most prompting [63], and many others [56, 36, 55, 28, 13, 63]. Although some of these techniques succeed in obtaining better quality answers, their performance still varies across queries and thus only ameliorate the problem for the human involved. In summary, the intuitions leveraged by prompt engineering techniques do not transfer across queries and tasks.

For example, we find that one prompt template outperforms another by 11 F1 points on one benchmark, and underperforms by 3 F1 points on another. The performance of different prompts on the same task can vary by as much as 70 F1 points (see Fig. 5.3).

Challenge 2: LLMs cannot Replace Integration Pipelines. In practice, solving integration tasks requires building software pipelines that in addition to obtaining good accuracy must also perform efficiently even on large inputs. Using LLMs in this scenario pose a challenge for two reasons: i) they have limited context, which prevents them from e.g., reading an entire table; ii) they are resource-intensive, e.g., the \$10K, 48-day example from the introduction. A better use of LLMs for data integration is alongside such existing software pipelines, but this requires adapting the data to input prompts, and then adapting the output of LLMs to answers that are meaningful to the software.

We introduce techniques to address Challenge 1 in Chapter 3 and those to address Challenge 2 in Chapter 4.

CHAPTER 3

LLM AS A CROWD

Given a data integration query, different prompts give different answers, some correct and some not. Prompt templates that lead to correct answers on one instance are not guaranteed to give good answers for a different instance. To make matters worse, we have no way of predicting what characteristics make a prompt good for a given instance, and we do not want to rely on access to large training data because this is precisely one of the challenges of state of the art techniques that we wish to avoid, i.e., collecting training data is expensive.

Our key observation is that while we do not know what prompt templates will perform well, we know that *some* prompt templates perform well on *some* tasks. We propose to create a *crowd* of prompt templates and then use *crowd truth inference methods* (or *crowd methods* for short) to identify the answer to the question. Rather than spending much effort on engineering very good prompts for different queries, we spend little effort in writing multiple prompts once for all queries. Such an approach removes the need for relying on any one prompt, and translates into good performance across different queries and tasks.

We give an overview of crowd truth inference methods, and then present our method for using LLMs as a crowd.

3.1 A Primer on Crowd Truth Inference

The goal of a crowd truth inference method is to find the true answers to a set of task instances from a crowd of noisy (poor-performing, adversarial) workers without access to the ground truth answers. Formally, crowd truth inference solves the following problem [62]: given a set of workers W and a set of task instances $T = \{t_1, \dots, t_n\}$ (e.g., pairs of rows that may be entity resolution matches), let v_i^w be the response of worker w for instance $t_i \in T$. Let V be the complete set of worker responses for all instances T and all workers W . That

is, $V = \{v_i^w | t_i \in T, w \in W\}$. Then, given V , the goal of crowd truth inference is to find the true answer for each instance, denoted as v_i^* for instance $t_i \in T$.

A familiar example of crowd truth inference is *majority voting*: choose each v_i^* to be the most frequently appearing value in $\{v_i^w | w \in W\}$. The performance of majority voting suffers when i) the performance of workers differs or; ii) the difficulties of the instances differ. The rich literature [62] in this area seeks to estimate worker reliability and instance difficulty without access to the *ground truth label* to improve over majority voting.

Methods that Explicitly Model Worker Reliability. A family of methods suitable to our setting is *probabilistic graphical model* (PGM)-based models, because these models explicitly model worker reliability using worker probabilities or confusion matrices, and can similarly model instance difficulty explicitly. They are also common [16, 57, 15, 35, 53, 42, 30]. Since they do not have access to the ground truth, they estimate worker reliability by observing apparent worker correlations and making modeling assumptions. For instance, a common assumption is that all workers are independent conditional on the true label. When this assumption holds, workers that more often adhere to the majority vote are likely high-quality. These crowd methods find the *most likely* v_i^* for each instance by estimating the probability distribution of the possible labels given worker responses, $P(v_i^* | \{v_i^w | w \in W\})$, and the probability distribution of worker responses given the possible labels, $P(v_i^w | v_i^*)$. To estimate these distributions, these methods typically perform maximum likelihood estimation via expectation-maximization.

Guarantees and Robustness. Since many of these works use expectation-maximization, they are only guaranteed to find a local optimum when: (i) the priors on worker answer distributions and the ground truth distribution do not differ drastically from the true distributions for worker answers and ground truth (ii) the modeling assumptions, such as conditional independence of workers, are met. There are PGM-based models that attempt to make weaker assumptions, including some that allow for some worker responses or tasks to be correlated

[38, 43].

3.2 Eliciting Crowds from LLMs

Here, we explain how we elicit crowds from LLMs to boost the performance of data integration tasks.

Prompt Templates as Workers

A *crowd* is a set of *prompt templates* drawn in some systematic way, with different phrasings, different prompt engineering techniques, or a combination of those.

Algorithm

We construct a *crowd* once per data integration task, and collect responses for each <prompt template, task instance> pair. Unstructured LLM responses are parsed into structured responses. Finally, we apply a crowd method on the structured responses to obtain final predictions for each task instance. Algorithm 1 shows this process in more detail.

Algorithm 1 LLMCrowdTruthInference

```
1: Input: LLM, CROWD, TASK, CROWDMETHOD, RESPONSEPARSER
2: RESPONSES =  $\emptyset$ 
3: for instance  $\in$  TASK do
4:   for template  $\in$  CROWD do
5:     prompt = template.makePrompt(instance)
6:     rawResponse = LLM(prompt)
7:     response = RESPONSEPARSER(rawResponse)
8:     RESPONSES.append((prompt.id, instance.id, response))
9:   end for
10: end for
11: PREDICTIONS = CROWDMETHOD(RESPONSES)
```

Given a LLM, a set of prompt templates CROWD, and a data integration task TASK, we gather crowd responses by using each template to map each instance to a concrete prompt (L5), querying the LLM with the prompt (L6), and parsing its response into a structured

format with some function `RESPONSEPARSER` (L7). We then run some crowd truth inference method `CROWDMETHOD` on the responses to come up with a prediction for each task instance (L11). We illustrate this process in Fig. 2.2.

How to construct *crowds*?

A crowd is characterized by a set of prompt templates, which may differ from each other in terms of both the prompt phrasing and which prompt engineering techniques are used. Crowd methods are generally designed to be robust against additional poorly-performing workers, so adding additional prompt templates will not generally drive performance down significantly. Ideally, we would choose *prompt templates* that are conditionally independent of one another. Techniques to elicit such a crowd require white box access to LLMs—so we can search over the full joint distribution—which we do not have. Instead, we construct the crowd template by providing different combinations of contexts, phrasings, and prompt engineering techniques, and then rely on crowd methods that are less sensitive to correlations between prompts.

We emphasize that while it is possible to handcraft prompt templates that perform well on a particular instance, there is no guarantee such prompts will perform well on other instances. We assemble a crowd without reference to individual prompt templates’ performance on the integration queries, to avoid biasing the choice. (We strictly follow this principle in the evaluation chapter).

Extending to Multiple Repetitions

Self-consistency [55] is a technique that improves LLM accuracy on certain tasks. It involves repeating prompts multiple times at high temperature to sample more of the tail of the LM joint probability distribution and taking a majority vote over all responses. Following this intuition, in some of our experiments we collect multiple responses for each prompt by

querying the LLM with the prompt multiple times. Crowd methods are able to incorporate multiple responses from the same worker on the same instance.

Summary

Instead of spending effort in crafting prompts that will break on new queries, thinking of the LLM as a crowd and using crowd truth inference techniques permits writing a crowd template once and still obtain good performance, all without access to training data, thus addressing **Challenge 1**.

CHAPTER 4

LLMS IN DATA INTEGRATION PIPELINES

In this section, we explain strategies to leverage LLMs for entity resolution, joinability, and unionability. The integration of LLMs into software pipelines addresses **Challenge 2**. Addressing this challenge consists of solving two problems:

- 1. Integrating LLMs with Integration Pipelines.** Solving entity resolution, joinability, and unionability on large table collections is a compute intensive task, for which many techniques have been proposed before. A typical architecture finds *candidates* approximately, e.g., blocking in entity resolution, and then applies more expensive *matchers* to only a subset of the data. For each task, we integrate the LLM in the place that minimizes the number of queries (and thus cost in money, time, and energy impact) while maintaining performance.
- 2. LLMs I/O.** First, expanding the crowd template requires carefully selecting what context to include, subject to context window length limitations of today’s LLMs. For example, giving an LLM an entire table is infeasible with today’s limits (e.g. 4096 tokens for ChatGPT). Second, obtaining the answer from the output of LLMs requires parsing text into query answers, e.g., booleans. This imposes constraints on the prompt to solicit easy-to-parse responses.

Next, we explain how to leverage LLMs for the data integration tasks, including how we address problem 1 and problem 2.

4.1 Entity Resolution

Entity resolution is the problem of identifying pairs of rows between two tables that represent the same real-world entity [40]. Whether two entities are truly the same can be subjective. However, entity resolution typically takes place on table collections with a closed world of concepts, meaning it can be solved computationally. It is used to combine tables, to clean

data and many other tasks.

More formally, let \mathcal{E} be a set of entities (from one table or multiple tables) with n tuples and m attributes. Given all distinct tuple pairs $(r, r') : r \neq r'$, where r, r' are in \mathcal{E} , the problem of entity resolution consists of determining which of the pairs represent the same real-world entity.

Traditional software pipelines apply a “divide-and-conquer” strategy to avoid expensive all-pairs comparisons. Concretely, entity resolution systems employ cheap, approximate *blockers* that return candidate row pairs with high recall. Then, they apply more expensive *matchers* downstream to improve the precision.

Integrating LLMs (problem 1). Using LLMs as blockers would be too expensive due to the large number of pairs of rows that need to be read and hard-to-implement due to the length limitation of context windows. Expensive here means to three resources: money, time, and energy. Instead, we propose to reuse traditional blockers to identify candidates efficiently and then use LLMs as a matcher, thus reducing LLMs intervention to only the space of candidates.

Input/Output (Problem 2). Each candidate row pair constitutes an instance of the entity resolution task which is used to expand a crowd template.

Our implementation of MAKEPROMPT() is straightforward - we simply fill in the prompt template with the specified elements of the input instance. See Fig. 2.2 for an example.

The implementation of PARSERESPONSE() requires careful wording to elicit answers easy to parse by traditional software. We used three strategies that proved effective: i) providing context before prompting [47] by using system messages (e.g., `You are a helpful assistant who can only answer YES or NO and then provide your reasoning.`) ii) few-shot prompting: past works [58, 44], observe that adding training examples within a prompt can be effective in generating answers matching the format required by the task. iii) asking again: if the software receives an unparseable answer, it queries again for a maximum of k times.

Example Prompt + LLM Response. See Fig 2.1.

user: Consider samples of the following two tables:

Table 1: assay_id,doc_id,description,assay_type, [...]

Table 2: [...] 0,1,11087,The compound was tested for [...]

Is it likely that column assay_id from Table 1 and column active_molregno from Table 2 capture the same type of information? Begin your answer with YES or NO.

model: NO. Based on the provided samples, it is likely that the column "assay_id" from Table 1 and the column "active_molregno" from Table 2 do not capture the same type of information. In Table 1, the "assay_id" column represents the unique identifier for each assay conducted. [...] On the other hand, in Table 2, the "active_molregno" column appears to represent an identifier related to the molecular structure of a compound. [...] These columns capture different types of information and serve distinct purposes within their respective tables.

Figure 4.1: ChatGPT Joinability Example

User: I'm a machine learning engineer looking for training data in our company's data lake. I have the following training dataset so far:
Address3,Postcode,LCG,Registered_Patients,[...]

I'm looking for data containing rows I can add to this. So far, I found the following dataset:
COUNTY,TOWN,API WELL NUMBER,WELL NAME, [...]
Modulo some data transformation, is it possible to add rows from this dataset to my current training dataset?

Model: NO. Based on the provided datasets, it seems that the two datasets have different structures and purposes. The first dataset appears to be related to medical practices, with information such as practice names, addresses, and registered patients. On the other hand, the second dataset seems to be related to oil and gas wells, with information like well names, locations, and company names. [...]

Figure 4.2: ChatGPT Unionability Example
4.2 Joinability and Unionability

Joinability and Unionability are two important problems in data discovery, which help data integration, which are defined as identifying all pairs of tables that join/union within a table collection [65, 64, 7, 11, 19, 21, 26, 46, 32, 20, 3].

Similar to entity resolution, the computational complexity of identifying whether two tables join or union with each other is high, as it requires comparing all pairs. For that reason, state of the art systems have implemented strategies that resemble those of the blocker-matcher pair in entity resolution. As a consequence, joinability and unionability systems have a similar architecture that consists of a cheaper algorithm to find candidates with high recall followed by a more expensive matcher/filter to improve precision.

We integrate LLMs similarly to entity resolution, as filters for join/union candidates (**problem 1**). Similarly, while the implementation of `makePrompt` and `parseResponse()` differs across tasks, the general architecture remains the same (**problem 2**), suggesting a straightforward, generic strategy to leverage LLMs for data integration tasks.

Example Prompt + LLM Response. See Fig. 4.1 and Fig. 4.2.

CHAPTER 5

EVALUATION

In this chapter, we answer the central research question of this paper: *can LLMs aid in data integration tasks?* To do so, we structure the chapter around specific research questions:

- **RQ1:** *How do we use LLMs?* Using well-known benchmarks for entity resolution, we measure the performance of different LLM-based methods, including but not limited to those we proposed in Chapter 3, to identify which one performs better.
- **RQ2:** *How do LLMs and the state-of-the-art compare? How does this vary with the availability, quality, and quantity of training data?*

After offering the main results on entity resolution, we study the performance of LLMs on other common data integration tasks:

- **RQ3:** *Do LLMs help joinability and unionability?*

Setup. We use ChatGPT (version `gpt-3.5-turbo-0301`) as it is a recent LLM which has shown good performance across many known tasks and is widely available to the public. We present more specific setup information along with each experiment.

5.1 RQ1. How do we use LLMs?

We study the ability of LLMs to solve entity resolution.

Benchmark. We use the established WDC Large-Scale Product Matching (LSPM) benchmark [37] and the newer, more challenging WDC Products benchmark [51]. Both benchmarks consist of pairs of rows from different product offers scraped from the internet. Some pairs refer to the same real-world entity and some pairs do not. Both benchmarks have train-validation-test splits, and the training splits are further divided by size, which is convenient to evaluate models when different amounts of training data are available. LSPM is split by product: cameras, computers, shoes, and watches, which permits studying the performance

Task	Benchmark	Training size	Test size	Topics	Source
Entity resolution	LSPM	small: 1408-2231 pairs medium: 4080-6331 pairs large: 15091-25891 pairs xlarge: 31293-53258 pairs	1011-1097 pairs	Product offers: cameras, computers, shoes, watches.	2017 Common Crawl[1]
	Products	small: 2500 pairs medium: 6000 pairs large: 19607 pairs	4500 pairs	Product offers across multiple domains	2020 Common Crawl corpus[1]
Joinability	ChemBL	N/A	140450 table-col pairs	chemicals, bioactivity, genomics	ChemBL database
Unionability	SANTOS small	N/A	27500 table pairs	schools, infra- -structure, government, commerce	CA, UK, US, AU Open Data
	TUS	N/A	614875 table pairs	schools, infrastructure, government, commerce	CA, UK Open Data

Table 5.1: Overview of benchmarks for each data integration task. of entity resolution on different datasets, i.e., understand the effect of using in-distribution vs out-distribution data. The Products benchmark comes in multiple variants. We use the variant where 50% of the pairs are "corner cases," defined as pairs selected to have low (for positive pairs) or high (for negative pairs) similarity according to various similarity metrics. Each variant includes three test sets where all, half, or none of the products in the test set pairs appear in the training set (denoted as "seen," "half-seen," and "unseen," respectively). Except where specifically noted, we use the "unseen" test set, as we believe this to be most representative of real-world entity resolution.

Metrics. The LLM method works as a matcher (see Chapter 4), processing candidate pairs from the preblocked set provided in the WDC LSPM and Product benchmarks. We choose F1 as the performance metric because this is a commonly used metric in entity resolution works [39, 45, 6]. In particular, it is used to evaluate Ditto so it helps us present our results in context.

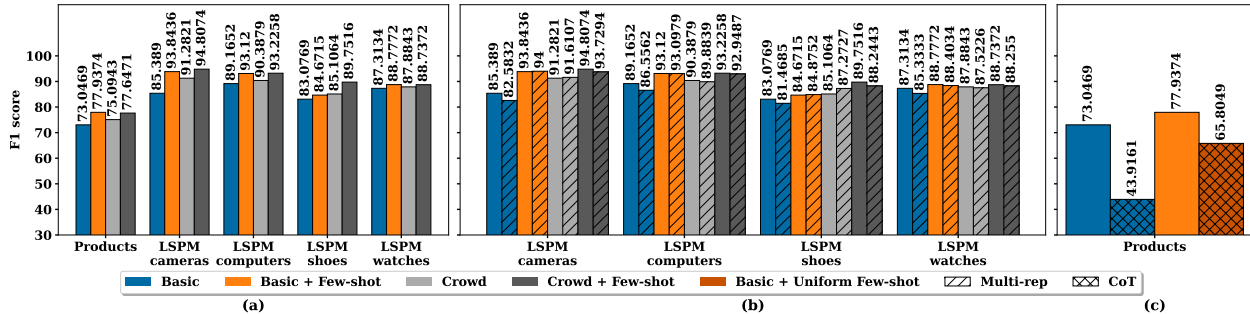


Figure 5.1: Comparison of LLM results with various prompt engineering strategies. X-axis is the benchmark. Y-axis is the F1 score (starts at 30). (a) Comparison of LLM results with and without few-shot learning and crowd truth inference on all datasets. (b) Comparison of LLM results with and without multiple repetitions at high temperature on LSPM benchmarks. (c) Comparison of LLM results with and without chain-of-thought prompting on Products benchmark.

Methods. All methods use ChatGPT (gpt-3.5-turbo-0301) as the LLM. We compare the performance of the following seven prompt engineering strategies, used in combination:

- **Basic.** We use a single prompt template that directly asks ChatGPT to perform entity resolution with temperature 0. This helps us understand the performance of ChatGPT without any additional intervention.
- **Few-shot.** As above, but we add few-shot examples to each prompt. This is a widely-used prompt engineering technique that works well in many different domains [4, 58, 44] We add to each prompt two positive and two negative example instances, sampled uniformly from the benchmark’s "small" training set. An example consists of a prompt (using the same template as the main prompt) along with the desired response.
- **Uniform Few-shot.** As above, but we use the same few-shot examples for the entire dataset.
- **Multi-rep.** We repeat each prompt multiple times at temperature 2 and take the majority response as the answer, similar to the prompt engineering method known as self-consistency [55].
- **CoT.** We use chain-of-thought prompting, another widely used prompt engineering technique [56]. We use the prompt described in **Basic** with an additional request to think step-by-step before producing the answer.

- **Crowd.** We run our LLM-as-a-crowd method. Our default crowd template, used unless otherwise stated, is formed from the **Basic** prompt described above, plus seven other prompts with varying phrasings. Our default crowd method is Dawid-Skene—we will justify this decision with an experiment.
- **OptimalCrowd.** As in our LLM-as-a-crowd method, we run a crowd on the benchmark. Then, we use the responses and the true labels to identify the crowd method that maximizes accuracy on the benchmark using only the prompt responses. Since this method uses the ground-truth labels to identify the optimal method, it is not a true crowd truth inference method; rather, we use it as an upper bound on what is achievable via crowd truth inference given the particular crowd and its responses.¹

When combining the techniques above, we note:

- When combining **Few-shot** with techniques like **Crowd** and **Multi-rep** that query ChatGPT multiple times per instance, we use the same few-shot examples for all queries of a given instance.
- When combining **Crowd** and **Multi-rep**, we use the method described in Section 3.2.
- Few-shot examples with **CoT** need manually-written explanations. For this reason, we use **CoT** with **Few-shot**, as that requires writing explanations for every training set example. We write manual explanations for four curated examples from the training set when using **CoT** and **Uniform Few-shot** together.

Next, we proceed to ask a set of questions and present the results progressively, towards answering *RQ1: How do we use LLMs?*

¹However, in some cases, this method underperforms relative to true crowd methods since it is optimized for accuracy and we use F1 as the metric. We chose to have this method optimize for accuracy regardless as it is more computationally feasible, and the true crowd methods are also attempting to maximize accuracy.

Which traditional prompt engineering techniques are effective?

Fig. 5.1(a) shows the performance of **Basic** and **Crowd** with and without **Few-shot** over all benchmarks. We observe trends:

- Using **Crowd** always improves performance.
- Using **Few-shot** always improves performance.
- Using **Crowd** and **Few-shot** together achieve the highest performance suggesting their benefits are complementary.

Fig. 5.1(b) shows the performance of **Multi-rep** combined with other techniques and crowd truth inference on the LSPM benchmarks. Although self-consistency shows improvement for other tasks it does not help for data integration tasks. We no longer use it in the remainder of this evaluation.

Fig. 5.1(c) shows the performance of **CoT** relative to **Basic** with and without **Few-shot**. We find that adding chain-of-thought prompting drops performance significantly, particularly in the zero-shot case, where the F1 score decreases by 29.131, due to a high false negative rate (the precision of **CoT** on this benchmark is just 31.4). See Fig. 5.2 for a false negative example.

In summary, of all techniques we tried, **Crowd** and **Few-shot** are effective and complementary to each other, while **Multi-rep** and **CoT** do not boost the performance for this task; in fact they affect it negatively. We now study in more depth the performance improvements of crowd methods.

Do crowd methods improve performance?

A user can construct multiple prompt templates, using different wording and/or prompt engineering techniques, but it is difficult to know in advance which will be most performant for a learning task. Table 5.2 shows that the *individual* template performance varies significantly

user: We are trying to integrate product data from two different databases. [...] Since the databases are different, there will still be some differences between entries that refer to the same product.

Here is an entry from the first database:

- title: Everyday Backpack 30L v2 - Midnight
[other attributes provided are: description, price, currency]

Here is an entry from the second database:

- title: Peak Design Everyday Backpack 20L v2 - Midnight
[other attributes provided are: description, price, currency]

As best as you can tell, do these entries refer to the same product? Let's think step-by-step:

model: Step 1: Compare the titles of the two entries.

The titles are similar but not identical. The first entry is titled [...]

Step 2: Compare the descriptions of the two entries.

The descriptions are similar and both mention the bag's ability to adapt to changing gear needs. However, there are some differences in the details mentioned.

Step 3: Compare the prices of the two entries.

The prices are different, with the first entry priced at HKD 2,599.00 and the second entry priced at USD 259.95.

Based on these differences, it is unlikely that these entries refer to the same product.

Therefore, the answer is no.

Figure 5.2: An example of a common false negative failure mode when using ChatGPT CoT (chain-of-thought): the model is fixated on the many non-meaningful differences.

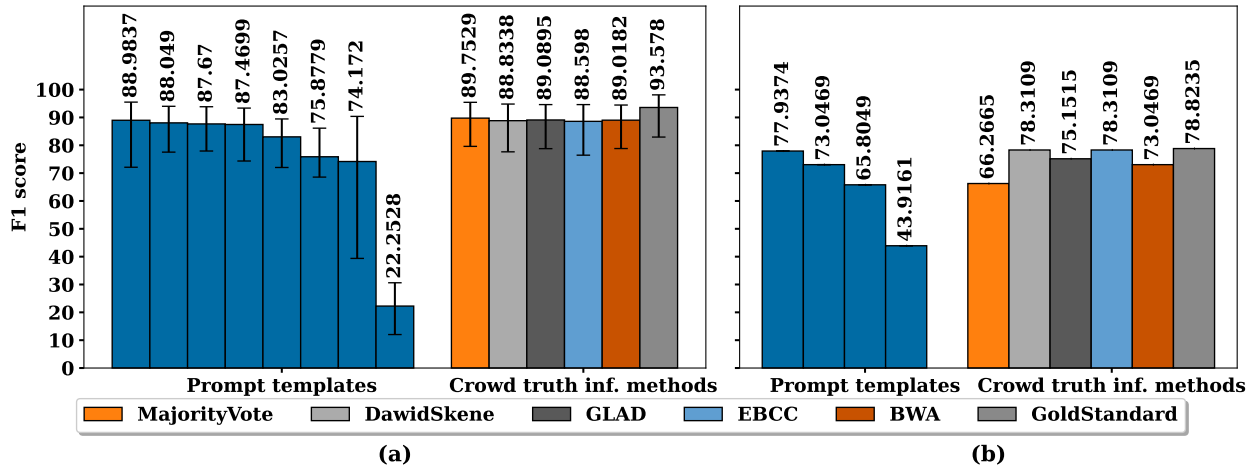


Figure 5.3: (left) Performance of various workers and crowd methods. Y-axis is the average F1 score. (a) Results for our standard crowd used in ChatGPT Crowd + Few-shot, averaged across all datasets. Error bars show the maximum and minimum dataset F1 scores. (b) Results from a crowd formed from various prompt engineering techniques - Basic + Few-shot, Basic, CoT + Few-shot, and CoT.

between datasets, demonstrating that it would be hard to commit to a template ahead of time. This result is true both when using zero- and few-shot, motivating our approach of using crowd truth inference to avoid relying on choosing a *good* prompt template.

In Fig. 5.3(a) we show the F1 score (averaged across all datasets, with the max and min shown in error bars) of the eight individual prompt templates that make up our default crowd (with Few-shot, shown on the cluster of bars on the left side of the plot), as well as the averaged F1 scores of various crowd methods on our default crowd, shown on the right side of the plot. This experiment shows:

- Individual templates’ performance varies widely across datasets, both absolutely and relatively. We cannot reliably predict absolute or relative template performance ahead of time.
- Crowd methods perform much more consistently across datasets, boosting the performance.
- Crowd methods’ perform almost as well as `OptimalCrowd`.

Now we re-analyze the crowd results of Fig. 5.1(a), which show that, while crowd truth

Dataset	Method	Rank		
		1	2	3
Products	Crowd	baseline	veryplain	security
	Crowd + Few-shot	baseline	veryplain	security
LSPM cameras	Crowd	veryplain	layperson	baseline
	Crowd + Few-shot	layperson	veryplain	baseline
LSPM computers	Crowd	layperson	baseline	veryplain
	Crowd + Few-shot	security	baseline	veryplain
LSPM shoes	Crowd	veryplain	baseline	layperson
	Crowd + Few-shot	layperson	security	veryplain
LSPM watches	Crowd	baseline	veryplain	security
	Crowd + Few-shot	layperson	veryplain	baseline

Table 5.2: Top 3 prompt templates (by F1 score) from our default crowd (with and without Few-shot).

inference improves performance with and without few-shot prompting, the benefits are much more modest with few-shot prompting. We want to understand why. We observe that few-shot prompting increases the partial correlations between workers (controlling for the true label as a covariate), as demonstrated in Fig. 5.4. Crowd methods like Dawid-Skene assume conditional independence of workers. Conditional independence is violated for most pairs of prompts in the zero-shot case (the chi-square test statistic is greater than 0.05 for only 4 out of 28 pairs, and less than $1.0e-5$ for 19 pairs), and all pairs in the two-shot case (the largest chi-square test statistic of any pair is $1.25e-11$). It is not generally possible to evaluate whether workers are conditionally independent without access to the true labels. The good news is that empirically, crowd methods perform well even when independence assumptions are violated, as demonstrated in Fig. 5.3(a).

Which crowd method performs best?

In Fig. 5.3(a), all crowd methods have similar overall performance. However, this is not true across all datasets and crowds. In 5.3(b), we form a crowd by using prompts that incorporate different prompt engineering techniques, rather than different prompt phrasings. Here, we find that the simplest method, MajorityVote, significantly under-performs relative to the others. What this result reveals is that in scenarios where the relative performance of

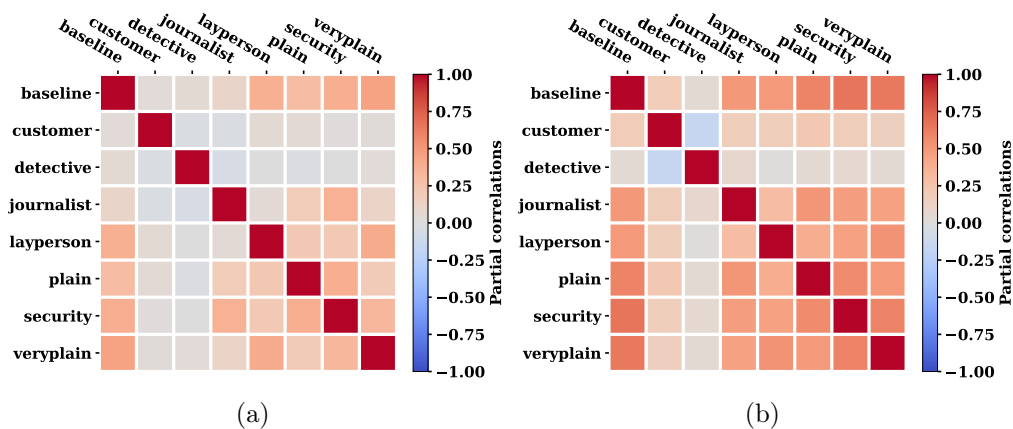


Figure 5.4: Partial correlations (controlling for the true label as a covariate) between the prompt templates of our default crowd across all datasets. (a) ChatGPT. (b) ChatGPT + Few-shot.

individual prompts lead to a deterioration of the quality of MajorityVote, more sophisticated techniques can still outperform. Of all crowd methods we choose Dawid-Skene because it is a simple model with relatively few parameters that nonetheless avoids the deterioration exhibited by Majority Vote. As shown in the figure, its performance is also close to the reference OptimalCrowd.

How much does the specific choice of few-shot examples matter? Is it important that they are sampled in-distribution?

We observed that few-shot is another effective technique to boost performance. We now study this in more depth, in particular paying attention to the effect of the examples chosen. We sample few-shot examples uniformly from the training set to get a more precise estimate of the benefit in expectation from adding few-shot examples. However, in a more realistic scenario, there may only be a few labeled examples that must be re-used for every prompt. To estimate the variability in performance due to choice of few-shot examples, we selected 10 different sets of few-shot examples uniformly at random from the LSPM computers training set, and using each set ran Crowd + Uniform Few-shot on the entire LSPM computers benchmark. The average F1 score for these 10 runs is 92.08 (an average improvement of 2.91 over zero-

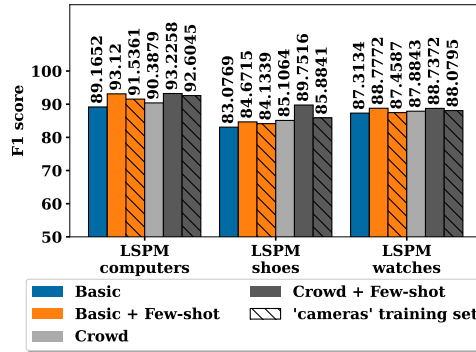


Figure 5.5: Comparison of LLM results with in- and out-of-distribution few-shot examples. X-axis is the benchmark. Y-axis is the F1 score (starts at 50). shot), and the sample standard deviation is 2.14. This indicates that the choice of few-shot examples does matter.

Fig. 5.5 compares the performance on LSPM computers, shoes, and watches when the few-shot examples are drawn from the corresponding training set vs when the few-shot examples are drawn from the cameras dataset. The out-of-distribution few-shot examples are still helpful, increasing scores by 1.06 with crowd truth inference and 1.19 without, but generally not as helpful as the in-distribution few-shot examples (which increase scores by 2.78 and 2.34 respectively).

Takeaways

Of the standard prompt engineering techniques we tried, we find that only FEW-SHOT is consistently helpful. Furthermore, we find crowd truth inference is helpful when there is uncertainty as to which prompt might have the highest performance, and even improves performance over the best prompt. We empirically find that Crowd + Few-shot works the best, so we use it as our main point of comparison in the following section.

5.2 RQ2. How do LLMs and the state-of-the-art compare?

In this section, we put our LLM results in context by comparing the LLM methods Crowd and Crowd + Few-shot with Ditto, the state of the art method for entity resolution. Because Ditto uses training data, we also ask how can training data—when available—be leveraged to boost LLMs. Concretely, we fine-tune the smallest OpenAI GPT-3 model, Ada, and include the results for context.

Benchmark. We use the WDC LSPM and Product benchmarks described before. In addition to using the LSPM benchmarks as before, we also consider the results when the various models are trained using the cameras training set (or, in ChatGPT’s case, the few-shot examples are drawn from the cameras training set) but tested on the other LSPM benchmarks; this is to understand whether Ditto trained on dataset X also performs well on dataset Y or whether its performance deteriorates, which would suggest the need for collecting training datasets for each scenario. For the Product benchmark, we additionally consider variants of the test set that contain various overlaps of products with the training set.

Metrics. All methods are Matchers, so we use F1. We compare the performance of the following four methods:

- ChatGPT Crowd: as before.
- ChatGPT Crowd + Few-shot: as before.
- Ditto. Ditto is trained and validated on various sizes (small, medium, large, or xlarge [LSPM only]) of the benchmark training sets. The number of examples in each size category vary by benchmark, but have similar magnitude. We list the number of examples in each size category in Table5.1.
- Fine-tuned Ada. We fine-tune OpenAI’s smallest GPT-3 model, Ada, on various sizes of the benchmark training sets, as above. This LLM method is more similar to Ditto in that

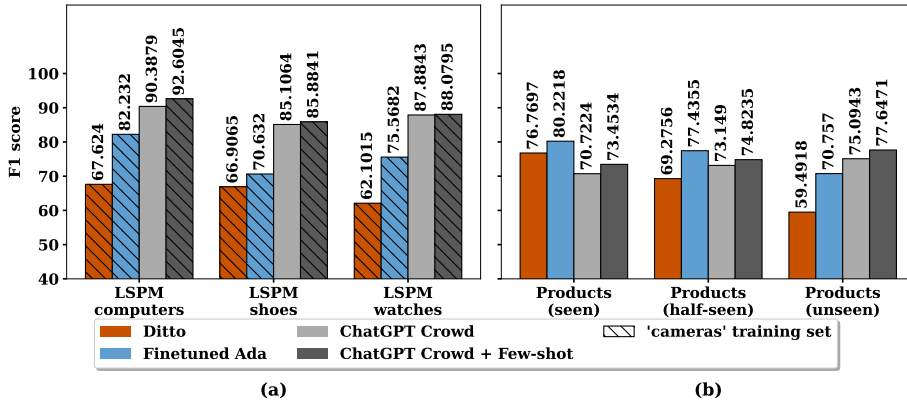


Figure 5.6: (left) Comparisons of Ditto and LLM techniques. X-axis is the benchmark. Y-axis is the F1 score (starts at 40). (a) Comparison of techniques when available training data are out-of-distribution (the training data is from LSPM cameras). (b) Comparison of techniques on variants of the Products dataset that require more or less generalization.

both fine-tune a language model using training data.

Training data availability. We evaluate our various methods under four scenarios of increasing access to high-quality labeled training data. This is to study how the methods perform in what is the hardest but most impactful scenario, i.e., no training data, while reproducing the environment in which state of the art methods were designed, i.e., with access to training data:

1. No training data relevant to the task is available.
2. Training data is available, but it is drawn from a different distribution than the task’s examples.
3. A limited amount of in-distribution training data is available.
4. A large amount of in-distribution training data is available.

Scenarios 1 and 2 are common in real-world contexts. Scenario 3 is relevant as labeling large amounts of data is often infeasible. Scenario 4 is the scenario that is implicitly assumed in many data integration papers, including Ditto.

Scenario 1: No relevant training data. Ditto requires training data, thus it does not work in this scenario. ChatGPT Crowd works in this scenario (it is unaffected), and few-shot

learning can still be used as long as a few test set examples can be manually labeled.

Scenario 2: Out-of-distribution training data. In our second scenario, we explore the case where the training and test datasets are drawn from different distributions. We train Ditto and fine-tune Ada on the LSPM cameras dataset and evaluate their performance on the other LSPM datasets. ChatGPT Crowd + Few-shot draws its few-shot examples from the cameras training set. We report the results in Fig. 5.6(a). We report results for the training set size with the highest performance - "xlarge" for Ditto and "large" for Ada, which are the most favorable scenarios for those methods. Ada significantly outperforms Ditto and ChatGPT significantly outperforms Ada, indicating that LLMs are the best option when there is not high-quality training data that is specific to the relevant task. We consider this a key result of our paper: LLMs achieve high-performance in difficult data integration tasks compared to existing baselines without requiring access to training data.

Is there information leakage? If the LLM performs well on these tasks because the benchmark data was used to build the LLM, the results may be misleading, and not transfer to new datasets. We ameliorate this concern by using a benchmark (Products) published in 2022 on a model (gpt-3.5-turbo-0301) trained up to 2021.

Scenario 3: Limited amounts of in-distribution training data. In the third scenario, we assume some existing training data is available, and we explore what happens when the amount of training data varies. We train Ditto and Ada on different sizes of the training datasets, and report the results in Table 5.3. Ditto relies on having a relatively large amount of training data - ChatGPT Crowd + Few-shot overall outperforms Ditto on LSPM when 5000 training examples are available (medium) and Finetuned Ada outperforms Ditto even when 20000 training examples are available (large), indicating that LLMs are the best option absent a large amount of training data.

Scenario 4: Large amount of in-distribution training data. In our final scenario, we assume that we have access to a large amount of high-quality training data, and that

Dataset	Model	Training set size			
		small	medium	large	xlarge
Products	Ditto	50.03	58.09	59.49	N/A
	Finetuned Ada	71.25	68.07	70.76	N/A
	ChatGPT Crowd + Few-shot	77.65	77.65	77.65	N/A
LSPM cameras	Ditto	75.18	84.98	91.94	94.90
	Finetuned Ada	87.78	92.62	89.98	89.77
	ChatGPT Crowd + Few-shot	94.81	94.81	94.81	94.81
LSPM computers	Ditto	78.62	83.68	92.36	94.79
	Finetuned Ada	91.48	93.49	92.23	94.74
	ChatGPT Crowd + Few-shot	93.23	93.23	93.23	93.23
LSPM shoes	Ditto	75.18	80.06	86.41	88.97
	Finetuned Ada	80.07	88.89	90.32	90.63
	ChatGPT Crowd + Few-shot	89.75	89.75	89.75	89.75
LSPM watches	Ditto	87.12	89.89	97.08	96.50
	Finetuned Ada	93.17	95.39	96.15	96.17
	ChatGPT Crowd + Few-shot	88.74	88.74	88.74	88.74
LSPM (mean)	Ditto	79.02	84.65	91.95	93.79
	Finetuned Ada	88.12	92.60	92.17	92.83
	ChatGPT Crowd + Few-shot	91.63	91.63	91.63	91.63

Table 5.3: Performance (F1 score) of Ditto, Finetuned Ada, and ChatGPT on the entity resolution benchmarks, with access to the training sets of various sizes. Numbers for ChatGPT are repeated as ChatGPT always draws its few-shot examples from the small training set. Bold numbers are the highest F1 score for the given size and benchmark. We highlight Products as we believe it to be the most informative benchmark for entity resolution.

the integration queries come from the same distribution as that training data—this is the most favorable scenario for Ditto, but the one least favorable to an analyst who needs to collect data for a target task. The results are reported in the rightmost columns of Table 5.3. We allow Ditto and Ada to train on the largest version of each training dataset. ChatGPT Crowd + Few-shot still draws its few-shot examples from the "small" version of each training dataset. LLMs significantly outperform Ditto on the Product dataset, achieving a state-of-the-art result. However, Ditto generally performs better on the various LSPM datasets. The Product benchmark was designed to be more challenging than the older LSPM benchmarks, so LLMs’ superior ability to handle semantic ambiguity may be better used on the former.

Generalization on Products. Digging deeper into the previous results, Fig. 5.6(b) shows the performance of the various methods on the variants of the Products test sets with differing levels of product overlap with the training set. Ditto’s performance drops significantly when the test set contains products that did not appear in the training data. Ada overall outperforms Ditto in all variants but also drops in performance somewhat. ChatGPT main-

tains similar performance in all variants and achieves the highest in the most challenging unseen category (right-most part of the plot).

Takeaway. We conclude LLMs have the highest performance in scenarios 1, 2, and 3, and may generalize better in scenario 4.

5.3 RQ3. Generalization to other Tasks

In this section, we explore the performance of LLMs for the other two data integration tasks we consider, joinability and unionability. Since these integration tasks do not explicitly use training data, we compare against the best zero-shot and few-shot LLM methods identified in RQ1: ChatGPT Crowd and ChatGPT Crowd Few-shot. We use Dawid-Skene as the crowd method. Our results align with those of entity resolution: ChatGPT Crowd + Few-shot boosts the precision of current joinable/unionable table search systems.

Joinable Table Search: Quantitative Analysis

In this experiment, we seek to understand the ability of LLMs to determine whether tables join with each other. We conduct two experiments.

Dataset. We use the ChemBL database, which is a database containing details of chemicals, their properties, experiments concerning them, etc. The database contains a well-designed schema that incorporates 100 PK/FK dependencies between attributes in the tables. We use these dependencies as the ground truth for this experiment.

Does the LLM correctly identify joinable tables? In this first experiment, we select the 100 PK/FK dependencies of ChEMBL (true positives) and 100 additional ones false PK/FK, which we choose at random from all other pairs. We then feed each of the 200 instances of this joinability problem to ChatGPT Crowd + Few-shot and measure its performance in terms of precision (how many of the returned answers are correct) and recall (how many of

the correct answers are identified). This results in a precision of 0.75 and a recall of 0.95, i.e., the LLM correctly identifies 95/100 PK/FKs, correctly dismisses 68/100 false PK/FKs. ChEMBL is a domain-specific database with concepts that are not hard to grasp and yet the LLM returns impressive results (see Fig. 4.1 for a qualitative example).

How do LLMs perform in joinability pipelines? While the previous experiment highlights the potential of LLMs, in a real joinability task, one must compare all pairs of tables to determine which ones join. This is a computationally expensive problem for which many approaches exist. Next, we use one such approach to identify candidates and use the LLM to *curate* the output results. The potential contribution of LLMs in this case is that they improve the precision of the system. We compare the following methods:

- **Aurum.** We use Aurum’s *pkfk_table* API, which returns the likely PK/FKs of a table from a table repository. We do this by inputting each table in the table repository to *pkfk_table*.
- **ChatGPT Basic.** As before, but with a prompt geared towards joinability.
- **ChatGPT Crowd.** This works as described previously, except we use a different set of seven prompt templates for joinability.
- **ChatGPT Crowd + Uniform Few-shot.** We use four curated examples - an obvious negative and positive, and a tricky negative and positive.

On ChEMBL, Aurum identifies 1150 joins with a precision of 0.05. When we apply ChatGPT Crowd + Uniform Few-shot on the candidates, the precision improves significantly, by 3x (to 0.137). This means the LLM correctly dismisses false positives identified by Aurum. Since Aurum aims for high recall, the LLM helps by curating the output and improving the precision. Aurum finds 1150 joins of which 57 are correct. ChatGPT Crowd + Uniform Few-Shot correctly filter out the joins to 408, while only incorrectly dismissing 1 FK, thus finding 56.

Compared to ChatGPT Crowd + Uniform Few-shot, ChatGPT Basic does much worse, even though it still improves the precision (to 0.09). This result confirms the benefits of using

Dataset	SANTOS	ChatGPT Basic	ChatGPT Crowd	ChatGPT Crowd + Uniform Few-shot
SANTOS	0.969	0.903	1.000	1.000
TUS	0.709	0.683	0.712	0.719

Table 5.4: Precision of LLM Methods vs SANTOS, by Dataset. Crowd for boosting the quality of LLMs on data integration tasks.

Unionable Table Search: Quantitative Analysis

In our last experiment, we consider the benefits of LLMs on unionability. We measure the benefits of using LLMs alongside state of the art systems. We use two datasets for which the ground truth is available: (i) the SANTOS’s "small" benchmark, used in SANTOS’s experiments, (ii) the table union search (TUS) dataset.

Similar to the previous experiments, we use the LLM to postprocess candidates returned by SANTOS, aiming to boost SANTOS’s precision. Therefore, we compare SANTOS’s precision to the precision of postprocessing SANTOS candidates. We report the original precision of SANTOS and include the effect of using LLM methods to curate the candidates identified by SANTOS.

The precision results are shown in Table 5.4. Similar to the joinability task, Crowd + Uniform Few-shot (with a crowd of five prompt templates) helps the upstream system, in this case SANTOS, by curating its output, improving the precision to 1. On the TUS dataset, using Crowd + Uniform Few-shot achieves a new state of the art performance number (see Fig. 4.2 for a qualitative example).

In summary, LLMs significantly help data integration tasks when used alongside a blocker (in entity resolution) or a candidate generator (in joinability or unionability).

CHAPTER 6

RELATED WORK

Incorporating Semantics into Data Integration. As mentioned in Section 2.1, there have been many efforts to resolve semantic ambiguity by incorporating semantic knowledge into data integration pipelines, ranging from the use of ontologies to the use of word embeddings.

Some works consider the problem of mapping tables to existing ontologies or knowledge bases as a precursor to integration tasks. For example, *semantic type annotation* is helpful to incorporate knowledge into the columns of a table that can then be leveraged by entity resolution, joinability, etc. Many semantic annotation works primarily label columns with labels from existing ontologies (e.g., DBpedia, Wikidata, an enterprise’s knowledge base, etc.), either via classification to given labels (e.g., *Sherlock* [27] and *Sato* [60]) or via mapping to several ontologies (e.g., C2 [33]—notably, C2 also uses curated webtables as an ontology). Several systems for semantic annotation from ontologies have also been developed for the SemTab Competition [2].

Other works more directly apply ontologies for data integration problems. For example, SemProp [8] labels tables with ontology classes, and determines table relatedness via links between these ontology classes. SANTOS [32] labels columns with semantic labels from YAGO and assesses table unionability using both schema similarity based on column labels and similarity of links among column labels in the same table.

Some works have also used word embedding similarity to solve data integration problems. TUS [46] and D3L [3] leveraged open-source word embeddings, such as word2vec and fastText to incorporate schema similarity as one of many metrics used to determine unionability. Warpgate [12] and PEXESO [18] use embeddings of pretrained models to determine whether columns contain overlapping values, where values are considered equal based on semantics rather than equality.

From the examples in this paper (see Fig. 4.1 and Fig. 4.2), we have qualitatively shown that LLMs indicate semantic types in their output, successfully incorporating much valuable context based on external knowledge. Overall, we see LLMs as a new tool that aids in the general effort of addressing data integration and we showed in the paper the promise of such technology.

LLMs for Integration & Table Understanding. Several papers have studied GPT-3 and ChatGPT’s performance on various data integration problems. [50] has studied ChatGPT’s performance on entity matching tasks using zero-shot and few-shot prompting, as well as specifying rules that may help. CHORUS [31] is a system that uses LLMs to perform semantic annotation while avoiding hallucination of labels. [52] has studied GPT-3’s performance at table understanding (meanings of cells, columns, rows), and tabular question-answering. StructGPT has studied GPT-3’s performance on tabular question-answering and semantic parsing of text to SQL. [10] studied GPT-4’s performance at various data analysis tasks, including sql query generation and plotting. To the best of our knowledge, this is the first paper to systematically study the effect of different prompting techniques in the application of LLMs to data integration tasks.

Crowdsourcing for Data Integration. Previous work has used human crowds to address data integration [17, 14, 49], showing the promise of aggregating the *opinions* of large samples of humans. A long line of work in the database community has focused on efficient crowdsourcing algorithms for data integration [54, 9], including decomposing larger tasks into subtasks more likely to elicit correct responses from human users [23]. In this paper, we elicit the crowd from large language models, which requires rethinking the prompting strategy—similar to preparing tasks for a human crowd—and then aggregating the results, in this case identical to aggregating human answers. We have shown that these techniques are effective when the crowds for data integration are elicited from LLMs.

CHAPTER 7

CONCLUSIONS

Large language models hold tremendous promise to aid in important data integration tasks such as entity resolution, joinability, and unionability. Despite challenges related to prompt engineering and their integration with software pipelines, this paper shows that thinking of LLM-as-a-crowd and leveraging the vast advances in data integration tasks in data management are effective techniques. The results show that LLMs produce high-quality results without access to training data and that thinking of them as crowds further reduces the reliance on high-cost prompt engineering.

Our work is a starting point toward using LLMs for data integration reducing reliance on training data or prompt engineering. As future work, we plan to expand the scope of integration tasks explored: for example, we observed that LLMs cannot be used for both blocking and matching of entity resolution, but our method used LLMs for matching. We can explore how LLMs can be used for the blocking phase of entity resolution.

Further, we used ChatGPT in our work. However, a cheaper, reproducible alternative are open LLMs, such as LLaMA2, Vicuna, Longformer, etc. These models hold promise as alternatives to the black-box OpenAI models. However, their performance is markedly worse. This leaves more room for improvement, which our crowd method may provide. Additionally, we can explore the choice of prompts to use in a crowd in more depth. The next-word joint probability distributions trained by these models can be fully explored, unlike with OpenAI LLMs. We can use these distributions to determine probabilities of correct answers on integration tasks for given prompts, allowing us to use training data not only to fine-tune LLMs or extract few-shot examples, but also to estimate the performance of given prompts, and use this as a more accurate prior for crowd truth inference methods. On a more speculative note, this may also help devise crowd truth inference methods that account for the characteristics of prompts using the joint distributions trained by LLMs.

REFERENCES

- [1] Common crawl. <https://commoncrawl.org/tag/2017/>. Accessed 6-22-23.
- [2] N. Abdelmageed, J. Chen, V. Cutrona, V. Efthymiou, O. Hassanzadeh, M. Hulsebos, E. Jiménez-Ruiz, J. Sequeda, and K. Srinivas. Results of semtab 2022. *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching*, 3320, 2022.
- [3] A. Bogatu, A. A. A. Fernandes, N. W. Paton, and N. Konstantinou. Dataset discovery in data lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 709–720, 2020. doi:10.1109/ICDE48307.2020.00067.
- [4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf.
- [5] M. J. Cafarella, A. Halevy, and N. Khousseinova. Data integration for the relational web. *Proc. VLDB Endow.*, 2(1):1090–1101, aug 2009. ISSN 2150-8097. doi:10.14778/1687627.1687750. URL <https://doi.org/10.14778/1687627.1687750>.
- [6] R. Cappuzzo, P. Papotti, and S. Thirumuruganathan. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, SIGMOD '20*, page 1335–1349, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367356. doi:10.1145/3318464.3389742. URL <https://doi.org/10.1145/3318464.3389742>.
- [7] R. Castro Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012, 2018. doi:10.1109/ICDE.2018.00094.
- [8] R. Castro Fernandez, E. Mansour, A. A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 989–1000, 2018. doi:10.1109/ICDE.2018.00093.
- [9] C. Chai, G. Li, J. Li, D. Deng, and J. Feng. Cost-effective crowdsourced entity resolution: A partial-order approach. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD '16*, page 969–984, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450335317. doi:10.1145/2882903.2915252. URL <https://doi.org/10.1145/2882903.2915252>.

- [10] L. Cheng, X. Li, and L. Bing. Is gpt-4 a good data analyst?, 2023.
- [11] N. Chepurko, R. Marcus, E. Zraggen, R. C. Fernandez, T. Kraska, and D. Karger. Arda: Automatic relational data augmentation for machine learning. *Proc. VLDB Endow.*, 13(9):1373–1387, jun 2020. ISSN 2150-8097. doi:10.14778/3397230.3397235. URL <https://doi.org/10.14778/3397230.3397235>.
- [12] T. Cong, J. Gale, J. Frantz, H. Jagadish, and Ç. Demiralp. Warpgate: A semantic join discovery system for cloud data warehouse. *arXiv preprint arXiv:2212.14155*, 2022.
- [13] A. Creswell and M. Shanahan. Faithful reasoning using large language models, 2022.
- [14] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh. Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions. *ACM Computing Surveys (CSUR)*, 51(1):1–40, 2018.
- [15] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28, 1979. ISSN 00359254, 14679876. URL <http://www.jstor.org/stable/2346806>.
- [16] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Zencrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In *Proceedings of the 21st International Conference on World Wide Web, WWW '12*, page 469–478, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312295. doi:10.1145/2187836.2187900. URL <https://doi.org/10.1145/2187836.2187900>.
- [17] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. Large-scale linked data integration using probabilistic reasoning and crowdsourcing. *The VLDB Journal*, 22(5):665–687, 2013.
- [18] Y. Dong, K. Takeoka, C. Xiao, and M. Oyamada. Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 456–467, 2021. doi:10.1109/ICDE51399.2021.00046.
- [19] M. Esmailoghli, J.-A. Quiané-Ruiz, and Z. Abedjan. Cocoa: Correlation coefficient-aware data augmentation. In *EDBT*, 2021.
- [20] G. Fan, J. Wang, Y. Li, D. Zhang, and R. J. Miller. Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning. *Proc. VLDB Endow.*, 16(7):1726–1739, may 2023. ISSN 2150-8097. doi:10.14778/3587136.3587146. URL <https://doi.org/10.14778/3587136.3587146>.
- [21] S. Galhotra, Y. Gong, and R. C. Fernandez. Metam: Goal-oriented data discovery. In *ICDE*. IEEE, 2023.

- [22] Y. Gong, Z. Zhu, S. Galhotra, and S. Castro Fernandez. Ver: View discovery in the wild. In *ICDE 2023*. IEEE, 2023.
- [23] A. Gruenheid, B. Nushi, T. Kraska, W. Gatterbauer, and D. Kossmann. Fault-tolerant entity resolution with the crowd. *arXiv preprint arXiv:1512.00537*, 2015.
- [24] A. Holtzman, J. Buys, M. Forbes, and Y. Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019. URL <http://arxiv.org/abs/1904.09751>.
- [25] A. Holtzman, P. West, V. Shwartz, Y. Choi, and L. Zettlemoyer. Surface form competition: Why the highest probability answer isn’t always right. *CoRR*, abs/2104.08315, 2021. URL <https://arxiv.org/abs/2104.08315>.
- [26] Z. Huang, P. Subramaniam, R. C. Fernandez, and E. Wu. Kitana: Efficient data augmentation search for auttml, 2023.
- [27] M. Hulsebos, K. Hu, M. Bakker, E. Zraggen, A. Satyanarayan, T. Kraska, c. Demiralp, and C. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’19, page 1500–1508, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi:10.1145/3292500.3330993. URL <https://doi.org/10.1145/3292500.3330993>.
- [28] J. Jung, L. Qin, S. Welleck, F. Brahman, C. Bhagavatula, R. Le Bras, and Y. Choi. Maieutic prompting: Logically consistent reasoning with recursive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1266–1279, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.emnlp-main.82>.
- [29] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 2. 02 2008.
- [30] D. R. Karger, S. Oh, and D. Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.
- [31] M. Kayali, A. Lykov, I. Fountalis, N. Vasiloglou, D. Olteanu, and D. Suciuc. Chorus: Foundation models for unified data discovery and exploration, 2023.
- [32] A. Khatiwada, G. Fan, R. Shraga, Z. Chen, W. Gatterbauer, R. J. Miller, and M. Riedewald. Santos: Relationship-based semantic table union search, 2022.
- [33] U. Khurana and S. Galhotra. Semantic annotation for tabular data. *CoRR*, abs/2012.08594, 2020. URL <https://arxiv.org/abs/2012.08594>.
- [34] U. Khurana and S. Galhotra. Semantic concept annotation for tabular data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge*

- Management*, CIKM '21, page 844–853, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384469. doi:10.1145/3459637.3482295. URL <https://doi.org/10.1145/3459637.3482295>.
- [35] H.-C. Kim and Z. Ghahramani. Bayesian classifier combination. In N. D. Lawrence and M. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 619–627, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL <https://proceedings.mlr.press/v22/kim12.html>.
- [36] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [37] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.*, 3(1–2):484–493, sep 2010. ISSN 2150-8097. doi:10.14778/1920841.1920904. URL <https://doi.org/10.14778/1920841.1920904>.
- [38] Y. Li, B. Rubinstein, and T. Cohn. Exploiting worker correlation for label aggregation in crowdsourcing. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3886–3895. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/li19i.html>.
- [39] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, sep 2020. ISSN 2150-8097. doi:10.14778/3421424.3421431. URL <https://doi.org/10.14778/3421424.3421431>.
- [40] Y. Li, J. Li, Y. Suhara, J. Wang, W. Hirota, and W.-C. Tan. Deep entity matching: Challenges and opportunities. *J. Data and Information Quality*, 13(1), jan 2021. ISSN 1936-1955. doi:10.1145/3431816. URL <https://doi.org/10.1145/3431816>.
- [41] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1–2):1338–1347, sep 2010. ISSN 2150-8097. doi:10.14778/1920841.1921005. URL <https://doi.org/10.14778/1920841.1921005>.
- [42] Q. Liu, J. Peng, and A. T. Ihler. Variational inference for crowdsourcing. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/cd00692c3bfe59267d5ecfac5310286c-Paper.pdf.
- [43] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 169–182, New

- York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336314. doi:10.1145/2809695.2809715. URL <https://doi.org/10.1145/2809695.2809715>.
- [44] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.
- [45] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD '18*, page 19–34, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450347037. doi:10.1145/3183713.3196926. URL <https://doi.org/10.1145/3183713.3196926>.
- [46] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller. Table union search on open data. *Proc. VLDB Endow.*, 11(7):813–825, mar 2018. ISSN 2150-8097. doi:10.14778/3192965.3192973. URL <https://doi.org/10.14778/3192965.3192973>.
- [47] OpenAI. Techniques to improve reliability. https://github.com/openai/openai-cookbook/blob/main/techniques_to_improve_reliability.md.
- [48] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.
- [49] N. W. Paton and A. Fernandes. Crowdsourcing feedback for pay-as-you-go data integration. *DBCrowd 2013*, page 32, 2013.
- [50] R. Peeters and C. Bizer. Using chatgpt for entity matching, 2023.
- [51] R. Peeters, R. C. Der, and C. Bizer. Wdc products: A multi-dimensional entity matching benchmark, 2023.
- [52] Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang. Evaluating and enhancing structural understanding capabilities of large language models on tables via input designs, 2023.
- [53] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi. Community-based bayesian aggregation models for crowdsourcing. In *Proceedings of the 23rd International Conference on World Wide Web, WWW '14*, page 155–164, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327442. doi:10.1145/2566486.2567989. URL <https://doi.org/10.1145/2566486.2567989>.

- [54] N. Vesdapunt, K. Bellare, and N. Dalvi. Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment*, 7(12):1071–1082, 2014.
- [55] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=1PL1NIMMrw>.
- [56] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- [57] J. Whitehill, T.-f. Wu, J. Bergsma, J. Movellan, and P. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/f899139df5e1059396431415e770c6dd-Paper.pdf.
- [58] S. M. Xie, A. Raghunathan, P. Liang, and T. Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=RdJVFCHjUMI>.
- [59] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: Entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 97–108, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312479. doi:10.1145/2213836.2213848. URL <https://doi.org/10.1145/2213836.2213848>.
- [60] D. Zhang, Y. Suhara, J. Li, M. Hulsebos, Ç. Demiralp, and W.-C. Tan. Sato: Contextual semantic type detection in tables. *Proc. VLDB Endow.*, 13(12):1835–1848, 2020. URL <https://doi.org/10.14778/3407790.3407793>.
- [61] M. Zhang, O. Press, W. Merrill, A. Liu, and N. A. Smith. How language model hallucinations can snowball, 2023.
- [62] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proc. VLDB Endow.*, 10(5):541–552, jan 2017. ISSN 2150-8097. doi:10.14778/3055540.3055547. URL <https://doi.org/10.14778/3055540.3055547>.
- [63] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, and E. H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=WZH7099tgfM>.
- [64] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller. Lsh ensemble: Internet-scale domain search. *Proc. VLDB Endow.*, 9(12):1185–1196, aug 2016. ISSN 2150-8097. doi:10.14778/2994509.2994534. URL <https://doi.org/10.14778/2994509.2994534>.

- [65] E. Zhu, D. Deng, F. Nargesian, and R. J. Miller. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, page 847–864, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450356435. doi:10.1145/3299869.3300065. URL <https://doi.org/10.1145/3299869.3300065>.