

Jellyfish: A Large Language Model for Data Preprocessing

¹Haochen Zhang, ²Yuyang Dong, ^{1,3}Chuan Xiao, ²Masafumi Oyamada

¹Osaka University, ²NEC Corporation, ³Nagoya University
 {chou.koushin, chuanx}@ist.osaka-u.ac.jp, dongyuyang, oyamada@nec.com

Abstract

This paper explores the utilization of LLMs for data preprocessing (DP), a crucial step in the data mining pipeline that transforms raw data into a clean format conducive to easy processing. Whereas the use of LLMs has sparked interest in devising universal solutions to DP, recent initiatives in this domain typically rely on GPT APIs, raising inevitable data breach concerns. Unlike these approaches, we consider instruction-tuning local LLMs (7 – 13B models) as universal DP task solvers that operate on a local, single, and low-priced GPU, ensuring data security and enabling further customization. We select a collection of datasets across four representative DP tasks and construct instruction tuning data using data configuration, knowledge injection, and reasoning data distillation techniques tailored to DP. By tuning Mistral-7B, Llama 3-8B, and OpenOrca-Platypus2-13B, our models, namely, Jellyfish-7B/8B/13B, deliver competitiveness compared to GPT-3.5/4 models and strong generalizability to unseen tasks while barely compromising the base models’ abilities in NLP tasks. Meanwhile, Jellyfish offers enhanced reasoning capabilities compared to GPT-3.5.

Our models are available at:

<https://huggingface.co/NECOUDBFM/Jellyfish> .

Our instruction dataset is available at:

<https://huggingface.co/datasets/NECOUDBFM/Jellyfish-Instruct> .

1 Introduction

The proliferation of large language models (LLMs) has catalyzed a diverse array of applications, extending beyond the domain of NLP to encompass a wide range of fields that require the processing of natural language data. Notably, LLMs have been applied in areas such as software engineering [77, 90], computer simulation [103, 24], data analytics [8, 85], and tabular data processing [53, 62, 111].

This paper focuses on the utilization of LLMs for data preprocessing (DP), a critical step in the data mining pipeline that involves transforming raw data into a manageable and processable format ready for use. Over the past decades, significant strides have been made in various DP tasks. Until 2021, most efforts were concentrated on one or two specific tasks such as error detection (ED) [28, 65], data imputation (DI) [80, 64, 66], schema matching (SM) [109], and entity matching (EM) [43, 56]. A key challenge in developing generic solutions to DP is that these tasks differ in nature: they deal with errors, anomalies, matches, etc. and require different actions such as detection, repairing, and alignment.

With the advent of LLMs like GPT-3 and subsequent versions, researchers have found a key to address this challenge, spurring the development of generic solutions for a wider array of DP tasks [70, 108]. The application of LLMs in DP has the following strengths: (1) The primary strengths of using LLMs in DP lie in their ability to process natural language. Most LLMs provide a prompting interface with which users can interact and assign tasks in natural language, contrasting with existing DP solutions that require computer programming or specific tools (e.g., HoloClean [80] and Magellan [43]). (2) With the knowledge acquired through training on vast amounts of data, LLMs are universal problem

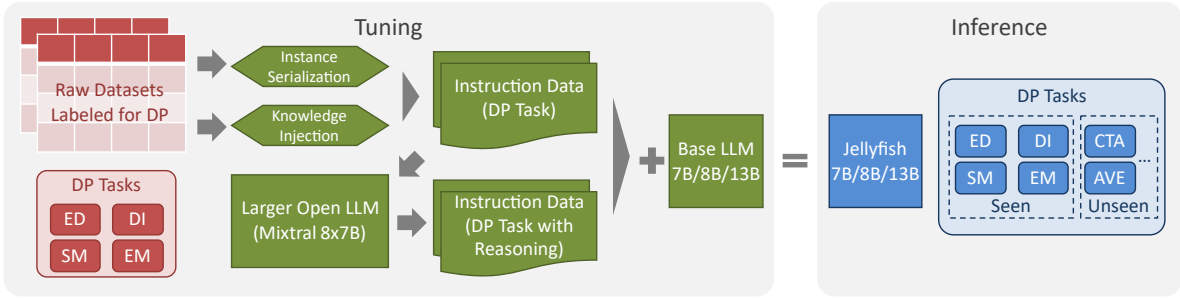


Figure 1: Overview of instruction tuning for data preprocessing.

solvers capable of identifying errors, anomalies, and matches in data (and particularly unseen datasets in unseen tasks), aligning with the aims of DP tasks without needing human-engineered rules [79]. (3) LLMs are excellent reasoners [42], enabling them to not only return DP results but also provide the reasons for these results. In this sense, their answers are more interpretable than those of other deep learning approaches. (4) LLMs can be conditioned by few- [5] or zero-shot [42] prompting. As such, we can condition the criteria for DP tasks (e.g., the degree of matching) using few-shot examples or zero-shot prompts, contrasting with traditional solutions based on a threshold [83, 43] or a time-consuming training process to fit to the data [66].

Despite these strengths, existing LLM-based solutions to DP [70, 108, 44], with reliance on GPT APIs, have raised concerns about data breaches, as evidenced by OpenAI’s first confirmed data breach involving ChatGPT [73]. Another limitation is the difficulty in domain specification [70]. When dealing with data from highly specialized domains, training the LLMs used in these solutions can be costly (e.g., GPT-3.5) and even unavailable due to frozen parameters (e.g., GPT-4), posing difficulty in customizing the model.

In response to these challenges, we propose to construct instruction data and tune LLMs for various DP tasks. The tuned model, namely Jellyfish, distinguish itself with several key features:

- **Versatility:** Jellyfish is a universal DP task solver tuned to the following tasks: ED and DI for data cleaning, and SM and EM for data integration.
- **Cost-Efficiency and Security:** Varying from 7B to 13B, Jellyfish can operate on a local, single, and low-priced GPU, ensuring data security and allowing further tuning.
- **Customizability:** Capable of understanding natural language, Jellyfish allows users to manually craft instructions for DP tasks (or simply use our prompts in this paper) and apply prompt engineering techniques to tailor it to specific tasks and datasets.
- **Domain Knowledge:** Unlike many existing methods that rely heavily on handcrafted knowledge during inference [80, 78], Jellyfish features domain knowledge in its instruction tuning and enables optional knowledge injection during inference.
- **Interpretability:** By employing reasoning data in its instruction tuning, Jellyfish’s interpretation ability provides natural language explanations of its outputs.

Whereas instruction tuning of LLMs has been largely used for unstructured text [110], the construction of Jellyfish is non-trivial in the sense that (1) it tunes for structured data, and (2) we need to find a good data configuration for various DP tasks and (3) specify domain knowledge that can be applied to unseen datasets. Besides, it is expected that the model’s performance in NLP tasks can be preserved for generalizability and further customization. To the best of our knowledge, this is the first study that investigates instruction tuning for DP with LLMs as universal solutions.

As depicted in Figure 1, Jellyfish is constructed by carefully selecting data from several public datasets widely used for DP evaluation, considering their impacts on the overall performance. By

instance serialization, raw data is serialized into instruction tuning prompts. By knowledge injection, task- and dataset-specific knowledge – particularly domain knowledge that can be extended to unseen datasets – is infused to the prompts. Moreover, we resort to Mixtral-8x7B-Instruct-v0.1 to generate reasoning data. As such, Jellyfish distills Mixtral’s knowledge in reasoning DP results.

Our evaluation focuses on tuning a set of prevalent open LLMs, including Mistral-7B-Instruct-v0.2 (as Jellyfish-7B), Llama 3-8B (as Jellyfish-8B), and OpenOrca-Platypus2-13B (as Jellyfish-13B). The results show that our instruction data applies to all these base models, substantially improving the DP performance. Compared to two categories of baseline methods, (1) non-LLM methods – typically solutions based on machine learning (ML) or pre-trained language models (PLMs) – and (2) LLM methods – typically GPT series methods, Jellyfish-13B consistently outperforms non-LLM methods on its seen datasets, and its effectiveness on unseen datasets even surpasses non-LLM methods on their respective seen datasets. Meanwhile, Jellyfish-7B/8B also exhibit competitiveness, especially on DI and EM tasks. For unseen tasks, Jellyfish models also deliver strong performance, rivaling GPT-3.5/4 models and showcasing generalizability to a wider range of DP tasks beyond the four tasks used for tuning. Our evaluation reveals the impacts of data configuration and the use of reasoning data in building Jellyfish, and discovers that Jellyfish barely compromises the base model’s NLP performance. Furthermore, experiments demonstrate the advantage of Jellyfish’s interpretation over GPT-3.5 in reasoning capabilities as well as the effectiveness of knowledge injection.

Our contributions are summarized as follows.

- We develop Jellyfish, an instruction-tuned LLM as a universal DP task solver.
- Jellyfish showcases several notable features: universal model design, moderate model size, assurance of data security, feasibility for further tuning, natural language instruction handling, optional specification of prior knowledge, and model interpretability.
- Our experiments demonstrate Jellyfish-7B, 8B, and 13B models’ effectiveness in DP task solving, generalizability to new tasks beyond what they are tuned for, and superior reasoning abilities.

The rest of the paper is organized as follows: Section 2 introduces the DP tasks targeted by our model and briefly reviews LLMs. Section 3 describes the instruction data for tuning Jellyfish. Section 4 introduces how to use Jellyfish for solving DP tasks. Section 5 discusses the extensions to unseen tasks. Section 6 reports experimental results and analysis. Section 7 reviews related works on DP. Section 8 concludes this paper.

2 Preliminaries

2.1 Data Preprocessing

In data mining, DP is a crucial step that deals with noise, missing values, inconsistencies, and heterogeneity in data. Major DP procedures include data cleaning, data integration, data transformation, and data reduction [26]. In this initial exploration of LLMs for DP, we concentrate on tabular data, one of the most common data types.

Our data model operates on relational tables specified by schemas. We assume all attributes are either numerical (including binary) or textual (including categorical) values. Diverging from the traditional definition that presents the entire dataset and finds or fixes all the errors (or matches, etc.) within, we define the problem by handling one record (or a pair, depending on the task) at a time, so the prompt can be easily written and its length is within LLMs’ token limitation. Next, we outline the DP tasks involved in this study:

- **Error Detection (ED):** Given a record (i.e., a tuple in a relational table) and an attribute, our task is to detect whether there is an error in the cell value of this attribute.

- **Data Imputation (DI)**: Given a record and an attribute such that cell value for this attribute is missing, our task is to infer its correct value.
- **Schema Matching (SM)**: Given a pair of attributes represented in the form of (name, description), our task is to find whether they refer to the same attribute.
- **Entity Matching (EM)**: Given a pair of records, our task is to infer whether they refer to the same entity.

These four tasks form the most critical part of DP [70, 108] and are extensively discussed in the context of data mining [26]. We use them for instruction tuning. Besides, we consider two unseen tasks:

- **Column Type Annotation (CTA)**: Given a table with no header, our task is to infer the type of each column from a set of predefined types (e.g., name, time, location).
- **Attribute Value Extraction (AVE)**: Given a text description of an entity and a set of predefined attributes, the task is to extract attribute values from the text description.

We term each input object an *instance*, i.e., a record for ED and DI, a pair of attributes for SM, a pair of records for EM, a table or a column for CTA, and a text description for AVE.

2.2 Large Language Models

With advancements in the field of natural language processing (NLP), LLMs have become one of the hottest topics in the AI research community. Representative LLMs include OpenAI’s GPT series (in particular, GPT-3, 3.5, and 4), Anthropic’s Claude, Google’s Gemini, Mistral AI’s Mistral [37], Meta’s Llama [93] series, as well as their variants that can be found at Hugging Face [35]. Due to their superb ability to process natural language, LLMs have not only been used in NLP applications (e.g., ChatGPT and Claude), but also catalyzed the rise of LLM-powered autonomous agents [96] as AI assistants (e.g., by GPTs) or tools for engineering [77, 31] or simulation [104, 103] purposes. Another popular LLM-centric research direction is retrieval-augmented generation (RAG) [51, 52], which gives LLMs access to external information to improve generation performance. We refer readers to [114] for a survey on LLMs. Some LLMs are open-source (e.g., Llama and Llama 2), and they can be fine-tuned with additional tasks to improve their abilities in logical reasoning, question answering, and so on. Among these fine-tuning approaches, instruction tuning [110] has become a prevalent one which further trains LLMs on a dataset consisting of (instruction, output) pairs in a supervised fashion, hence bridging the gap between the next-word prediction objective of LLMs and the users’ objective of having LLMs adhere to human instructions. For efficiency of fine-tuning, parameter-efficient fine-tuning (PEFT) approaches enable adaptation of LLMs to downstream applications without fine-tuning all the parameters. Notable methods are adapter tuning [32], prefix-tuning [55], and low-rank adaptation (LoRA) [33]. In particular, LoRA achieves significantly fewer trainable parameters and no additional inference latency, and has become a prevalent PEFT approach.

In addition to the strengths outlined in Section 1, we discuss the limitations of LLMs in the context of DP: (1) LLMs often require substantial computational resources, thereby increasing the cost of use and compromising the efficiency and scalability of DP on large-scale data. (2) Due to token limitation (the maximum input length, e.g., 4k tokens for GPT-3.5) and lack of memory for keeping historical information, the input to the LLM is often instance-by-instance, and the DP results may exhibit inconsistency across different instances. Simply raising the token limitation (e.g., 128k tokens for GPT-4-turbo) does not solve the problem, because performance may degrade due to increased lengths of input [58]. (3) LLMs sometimes exhibit hallucination [113], i.e., they generate text that is plausible-sounding but factually incorrect or non-sensical, as they lack a fundamental understanding of the world and rely solely on the patterns they learned during training.

Table 1: DP task data statistics. #Positives denotes the number of instances having an error (for ED) or matching objects (for SM and EM). $\times 2$ denotes duplication of instances for treating missing values as errors or not.

Task	Dataset	#Instances	#Positives
ED	Adult	550 $\times 2$	35 $\times 2$
	Hospital	1710 $\times 2$	44 $\times 2$
DI	Buy	586	N/A
	Restaurant	778	N/A
SM	MIMIC-III	7000	11
	Synthea	5000	18
EM	Amazon-Google	6874	699
	Beer	359	54
	DBLP-ACM	5000	885
	DBLP-GoogleScholar	5000	924
	Fodors-Zagats	757	88
	iTunes-Amazon	430	105

3 Instruction Tuning for Data Preprocessing

3.1 Dataset Preparation

For the four seen tasks, we choose a series of datasets that have been widely used in previous studies and cover a variety of application domains. (1) ED: Adult and Hospital [28]; (2) DI: Buy and Restaurant [66]; (3) SM: MIMIC-III and Synthea [109]; (4) EM: Amazon-Google, Beer, DBLP-ACM, DBLP-GoogleScholar, Fodors-Zagats, and iTunes-Amazon from the Magellan data repository [16]. We use the publicly available version of these datasets [70], where errors and missing values are already injected to the datasets of ED and DI, respectively.

To determine the data size for each task, we first consider a constraint that for fair comparison with non-LLM methods [66, 109, 56], the training data in building Jellyfish does not exceed those used for building these methods, which serve as a pool of 115k instances. Then, we have the following observations (Section 6.6): (1) The performance of DI can benefit from the other three tasks, but increasing DI data is relatively negative to them. (2) Increasing ED and SM data is generally beneficial to other tasks. (3) Increasing SM data is beneficial to the overall DP performance. (4) Increasing EM data compromises the performance of other tasks, but keeping its size is the key to the EM performance.

Based on these observations, we use all the ED and DI data in the 115k pool as their sizes are small, and then choose a large data size for SM and a moderate data size for EM. Specifically, we control the data used in large EM datasets (e.g., for DBLP-GoogleScholar, 1/3 is chosen from the pool). As such, we determine the data size for the four tasks, as shown in Table 1.

In addition, we undertake the following efforts to prepare data: (1) Given the disproportionately small number of positive instances compared to negative ones, we incorporate all positive instances available in the datasets. (2) For ED, since missing values can be interpreted as either errors or non-errors, depending on the context, we create two versions of each instance: one treating missing values as errors and the other as non-errors. The duplication is guided by knowledge injection, which is to be detailed in Section 3.2.

Next, we transform raw data to (1) DP task data, for DP task-solving ability, and (2) DP task with reasoning data, for interpretation ability. They can be jointly used for tuning a Jellyfish model.

3.2 DP Task Data

To prepare the DP task data for an LLM, we need to serialize (a.k.a. contextualize) each instance in the raw data to a prompt. The prompt contains the task description, the instance content, and any

	DP Task Data	DP Task with Reasoning Data
system message	You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can.	You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can. While answering, provide detailed explanation and justify your answer.
task description	You are tasked with determining whether two Products listed below are the same based on the information provided. Carefully compare all the attributes before making your decision.	
injected knowledge	Note that missing values (N/A or "nan") should not be used as a basis for your decision.	
instance content	Product A: [name: "Sequoia American Amber Ale", factory: "Wig And Pen"] Product B: [name: "Aarhus Cains Triple A American Amber Ale", factory: "Aarhus Bryghus"]	
question	Are Product A and Product B the same Product?	
output format	Choose your answer from: [Yes, No]	After your reasoning, finish your response in a separate line with and ONLY with your final answer. Choose your final answer from [Yes, No].
answer	No	... the names are not identical, with Product B having ... These are different factories ... they are not the same product. Final answer: No

Figure 2: Example prompt in instruction data. The leftmost column is description and not prompted to the model. Response indicates the answer to the prompt. Detailed prompts are provided in Appendix B.

injected knowledge. To describe our techniques for constructing the DP task data for training, we use an example for an instance in the Beer dataset used for EM, as shown in Figure 2.

At the beginning, there is a system message guiding the model behavior. Here, we instruct the model to act as an AI assistant to answer the user’s question, and its response should always respect this constraint. Then, we describe the DP task. The following part refers to injected knowledge. There are two types of injected knowledge: (1) general knowledge that applies to many datasets, and (2) specific knowledge that only applies to the given dataset. In this example, the knowledge belongs to general knowledge and concerns with missing values. Such knowledge injection may prevent the model incorrectly handling certain values in the dataset, especially when training data is noisy. The following part pertain to the instance content. Finally, there is a question presented to the model, and the output format is specified afterwards.

Whereas in the above example we specify knowledge on missing values, there are other forms of general knowledge used in tuning, including error types and terminology. For example, for ED, we inform the model of the fact that errors can include, but are not limited to, spelling errors, inconsistencies, or values that do not make sense for that attribute; for EM, we instruct the model to consider the full name of an attribute and its acronym to determine if the two values are the same. Specific knowledge highly depends on the application domain, mainly including constraints or rules that pertain to the dataset. For example, in publication datasets, authors’ names may occur in different forms and different orders even for the same article. Additionally, the model can be configured to assign greater importance to certain attributes. In the context of product data, for example, the model is directed to prioritize the comparison of product numbers. Specific knowledge can be applicable to datasets within the same domain, thereby enhancing the model’s performance on unseen datasets, particularly in scenarios where prior knowledge about these datasets is absent. Overall, the knowledge injected through tuning becomes the built-in knowledge of the model and can be used even without user-specification during inference.

3.3 DP Task with Reasoning Data

(DP task with) reasoning data, not only empowers the model to interpret the DP results, but also has the potential in enhancing the DP performance in the sense that the model can learn the rationale behind DP, thereby generalizing to unseen scenarios whose underlying logic resembles the tuned

Table 2: Statistics of reasoning data for instruction tuning. We report the numbers of instances for each task. For ED and DI, the numbers refer to the amount after duplicating the instances having missing values.

Dataset	Task				
	ED	DI	SM	EM	Total
reasoning-8k	3056	1364	2000	2000	8420
reasoning-11k	3056	1364	3500	3500	11420
reasoning-14k	3056	1364	5000	5000	14420
reasoning-20k	3056	1364	8600	7000	20020

tasks/datasets. On the other hand, due to the small size of local LLMs, tuning the model with excessive reasoning data may compromise its ability to conduct the tuned DP tasks. Thus, we need to strike a balance between DP performance and generalizability. In general, we observe that native models (Mistral and Llama 3) are more likely to benefit from the use of reasoning data (Section 6.7).

Another key feature in our reasoning data is that we resort to a larger open LLM, Mixtral-8x7B-Instruct-v0.1, to retrieve reasoning answers as ground truths. As such, Jellyfish distills Mixtral’s knowledge in reasoning for DP. Since this does not involve external APIs like GPT-4, data security can be ensured, in case users want to include confidential information in the reasoning data.

We use the same set of datasets as DP task data to construct the reasoning data. The prompt in reasoning data only differs from DP task data in the reasoning instructions (Figure 2, system message and output format). To retrieve reasoning answers from Mixtral, we add a hint at the end of the prompt for the correct DP result (e.g., “yes/no” for matching tasks), hence to instruct Mixtral to reason in the right direction (Appendix B.2). Note that such hint does not appear in the prompt given to Jellyfish.

To control the size and quality of reasoning data, we select data as follows: (1) For ED and SM, we keep all positive instances due to their small numbers, and then sample negative instances. (2) For DI, we keep all instances due to the small data size. (3) For EM, we sample instances. From the 115k pool, we tune the numbers in the sample to make four sets of reasoning data with 8k, 11k, 14k, and 20k instances, respectively (Table 2). Moreover, from the answers returned by Mixtral, we remove low-quality ones that simply rephrase instance contents, as they barely refers to reasoning.

4 Data Preprocessing with Jellyfish

For inference, the prompt is same as the instruction data shown in Figure 2. Users can craft dataset-specific knowledge into the prompt, such as the domain knowledge (e.g., constraints) outlined in the previous section. Such user-specified knowledge is optional.

Feature Engineering. Users can optionally select a subset of features to improve performance. For instance, for EM in the Beer dataset, **name** and **factory** are more relevant features, while **style** and **ABV** are less relevant. Hence users may choose to use only **name** and **factory** as attributes. Such feature engineering can be also implemented in the prompt as specific knowledge, e.g., *you should only consider name and factory and ignore other attributes*.

Prompt Engineering. Prompt engineering [100] is the process of structuring text to enhance the model performance. We incorporate few-shot prompting [5], which conditions the Jellyfish models to learn from a small selection of examples drawn from the dataset. The prompts for few-shot examples are reported in Appendix D.

Batch prompting [9] is a prompt engineering technique designed to enable models to perform inference in batches, rather than processing single instances individually. This approach involves presenting multiple instances within a single prompt, with the model instructed to respond to all of them concurrently. Though proven effective for GPT-3.5 and GPT-4 in reducing token consumption

and execution time [108], we have opted not to employ this method in the current version of Jellyfish. Our concern is that overburdening a 7B or 13B model with an excessive number of tokens, even when staying within its token limitation, could lead to diminished attention. This might significantly impair performance, potentially resulting in the model overlooking responses to some instances. We anticipate that future research, utilizing larger models or increasing the token input capacity, could effectively address this limitation.

5 Extensions to Unseen Tasks

For unseen tasks, we consider two case studies: CTA and AVE, as outlined in Section 2.1. Jellyfish models can be easily extended to support them by employing the prompt engineering techniques in existing LLM-based solutions, hence simplifying its use in unseen tasks.

Column Type Annotation. As a task in the realm of table understanding, CTA is an essentially DP step for data search [7], knowledge base completion [82], and data integration a data lake [25]. We follow the two-stage pipeline proposed in [44], which was designed for ChatGPT and based on chain-of-thought [99], a technique that enables complex reasoning capabilities through intermediate reasoning steps.

Given a table to be annotated, in the first stage, the model predicts the domain of the table. In the second stage, given a set of predefined types, the model determines the type of column based on sample values extracted from it. The chain-of-thought prompt instructs the model in a step-by-step manner. For example, to predict the domain of the table, there are four steps: (1) look at the input and make a table out of it, (2) look at the cell values in detail, (3) decide if the table describes domain A, domain B ... and (4) answer with the domain. Then, the model follows this prompt to cope with the task. The column type selection in the second stage works in the same way, except that table is replaced by column and domains are replaced by candidate types.

Attribute Value Extraction. Given a text description, AVE is an information extraction task that discovers missing values of attributes and reconstructs a table. For this task, we follow the prompt in [4] designed for GPT-4. The prompt is simple, beginning with the task description. Then, the instance content follows, with the description of the entity and the attribute to be extracted. Finally, an exception rule is mentioned: if the attribute cannot be extracted, the model should answer “N/A”.

We also would like to mention that Jellyfish models enable further fine-tuning. Users may choose to condition the model for specific DP tasks or domains to seek better performance. Moreover, Jellyfish models can be utilized for multiple tasks in a DP pipeline, e.g., data cleaning followed by data integration on the same sets of data. It is likely that the DP tasks within this pipeline belong to the same domain. In this case, Jellyfish models may deliver consistency in handling the data in different tasks due to the built-in domain knowledge acquired through instruction tuning for DP.

6 Experiments

6.1 Experimental Setup

Datasets. Apart from the seen datasets in Jellyfish (Section 3), we use the following datasets as unseen data, where CTA and AVE are used for case studies on unseen tasks. (1) ED: Flights and Rayyan [65]; (2) DI: Flipkart [22] and Phone [81] from Kaggle; (3) SM: CMS [109]; (4) EM: Abt-Buy and Walmart-Amazon from the Magellan data repository [16]; (5) CTA: SOTAB [44]; (6) AVE: AE-110k and OA-Mine [4]. The statistics of the datasets are reported in Table 3. We generate train/valid/test splits following the protocols for Adult and Hospital [28], Flipkart and Phone [66], and MIMIC-III and CMS [109]. The other datasets have already been provided with splits [70, 44, 4]. A subset of the train/valid splits is used in Jellyfish, as reported in Tables 1 and 2.

Table 3: Testing dataset statistics. For Walmart-Amazon, the entities belong to a different category of products from the Amazon dataset used for instruction tuning.

Task	Type	Dataset	#Instances
ED	Seen	Adult	9900
		Hospital	17101
	Unseen	Flights	12832
		Rayyan	8997
DI	Seen	Buy	65
		Restaurant	86
	Unseen	Flipkart	2675
		Phone	1194
SM	Seen	MIMIC-III	6408
		Synthea	2964
	Unseen	CMS	2564
EM	Seen	Amazon-Google	2293
		Beer	91
		DBLP-ACM	2473
		DBLP-GoogleScholar	5742
		Fodors-Zagats	189
		iTunes-Amazon	109
	Unseen	Abt-Buy	1946
		Walmart-Amazon	2049
CTA	Unseen	SOTAB	250
AVE	Unseen	AE-110K	1482
		OA-Mine	2451

Jellyfish Models. We instruction-tune three base models: (1) Mistral-7B (Mistral-7B-Instruct-v0.2 [37]), (2) Llama 3-8B (Llama-3-8B-Instruct [67]), and (3) OOP2-13B (OpenOrca-Platypus2-13B [49]), a Llama 2-13B variant with enhanced reasoning capabilities and logic proficiency. The tuned models are dubbed Jellyfish-7B, Jellyfish-8B, and Jellyfish-13B, respectively. The 7B and 8B models are tuned with both DP task and reasoning data (15k reasoning instances for the 7B model and 8k for the 8B model). The 13B model is tuned with only DP task data. As such, Jellyfish-7B and Jellyfish-8B are interpretation models while Jellyfish-13B is a task solver dedicated to the tuned tasks.

We report hyperparameter setup in Appendix A and injected knowledge in Appendix C. For inference, the (zero-shot) prompts are the same as DP task data and reasoning data, respectively. We apply general knowledge in the prompts, e.g., missing values in matching tasks and error types in ED. Dataset-specific knowledge is not used. When few-shot prompting is enabled, we equip LLMs with three examples for each dataset, covering both positives and negatives (Appendix D).

Baselines. We categorize existing methods into non-LLM methods and LLM methods. For non-LLM methods, we select the following baselines, in line with [70]: (1) ED: HoloDetect [28] and Raha [65]; (2) DI: IPM [66]; (3) SM: SMAT [109]; (4) EM: Ditto [56] and Unicorn [94]; (5) CTA: RoBERTa [61]. For their performance, we follow the best numbers reported in prior works [70, 44, 94]. Other methods such as Baran [64], HoloClean [80], and DODUO [89], have been shown to be outperformed by the above competitors [66, 70, 44], and hence are not compared here.

LLM methods are GPT-3 (text-davinci-002), GPT-3.5 (gpt-3.5-turbo-0301), Table-GPT [53] (GPT-3.5 fine-tuned for tables), GPT-4 (gpt-4-0314), GPT-4o (gpt-4o-2024-05-13), Stable Beluga 2 70B [63], and SOLAR 70B [95]. We follow the numbers reported in previous works [70, 108, 4]. Few-shots are used in line with Jellyfish models for fair comparison. TableLlama [111], which can handle CTA, is not compared because it is tuned for CTA, whereas our purpose is to evaluate the performance of LLMs on CTA as an unseen task.

Metrics. For DP task solving, we measure accuracy for DI, F1 score for ED, DI, EM, and AVE, and

Table 4: DP performance on seen tasks, accuracy for DI and F1 score for the other three tasks, with winners in boldface and runners-up underlined. All datasets are seen for non-LLM methods and Table-GPT. All datasets are unseen for GPT-3/3.5/4/4o. For LLM methods, zero-shot is used on seen datasets and few-shot is used on unseen datasets. “–” indicates numbers not reported in prior works for this dataset.

Task	Type	Dataset	Model								
			Best of non-LLM	GPT-3	GPT-3.5	GPT-4	GPT-4o	Table-GPT	Jellyfish-7B	Jellyfish-8B	Jellyfish-13B
ED	Seen	Adult	<u>99.10</u>	<u>99.10</u>	92.01	92.01	83.58	–	77.40	73.74	99.33
		Hospital	94.40	97.80	90.74	90.74	44.76	–	94.51	93.40	<u>95.59</u>
	Unseen	Flights	81.00	–	–	83.48	66.01	–	69.15	66.21	<u>82.52</u>
		Rayyan	79.00	–	–	<u>81.95</u>	68.53	–	75.07	81.06	90.65
DI	Seen	Buy	96.50	98.50	98.46	100	100	–	98.46	98.46	100
		Restaurant	77.20	88.40	<u>94.19</u>	97.67	90.70	–	89.53	87.21	89.53
	Unseen	Flipkart	68.00	–	–	89.94	83.20	–	87.14	<u>87.48</u>	81.68
		Phone	86.70	–	–	90.79	86.78	–	86.52	85.68	<u>87.21</u>
SM	Seen	MIMIC-III	20.00	–	–	40.00	29.41	–	53.33	<u>45.45</u>	40.00
		Synthea	38.50	45.20	<u>57.14</u>	66.67	6.56	–	55.56	47.06	56.00
	Unseen	CMS	<u>50.00</u>	–	–	19.35	22.22	–	42.86	38.10	59.29
EM	Seen	Amazon-Google	75.58	63.50	66.50	74.21	70.91	70.10	81.69	<u>81.42</u>	81.34
		Beer	94.37	100	96.30	100	90.32	96.30	100.00	100.00	96.77
		DBLP-ACM	98.99	96.60	96.99	97.44	95.87	93.80	98.65	98.77	<u>98.98</u>
		DBLP-GoogleScholar	<u>95.70</u>	83.80	76.12	91.87	90.45	92.40	94.88	95.03	98.51
		Fodors-Zagats	100	100	100	100	93.62	100	100	100	100
		iTunes-Amazon	97.06	<u>98.20</u>	96.40	100	98.18	94.30	96.30	96.30	98.11
	Unseen	Abt-Buy	89.33	–	–	92.77	78.73	–	86.06	88.84	<u>89.58</u>
		Walmart-Amazon	86.89	87.00	86.17	90.27	79.19	82.40	84.91	85.24	<u>89.42</u>
	Average		80.44	–	–	<u>84.17</u>	72.58	–	82.74	81.55	86.02

Table 5: Precision (P), recall (R), and F1 score on SM.

Type	Dataset	Model								
		SMAT			GPT-4			Jellyfish-13B		
		P	R	F1	P	R	F1	P	R	F1
Seen	MIMIC-III	11.5	84.6	20.2	33.33	50.0	40.0	45.45	35.71	40.0
	Synthea	24.4	90.9	38.5	71.42	62.5	66.67	41.18	87.50	56.00
Unseen	CMS	33.9	95.0	50.0	60.0	11.5	19.35	57.14	61.54	59.26

micro-F1 for CTA, all reported on a 100-scale.

Environment. Training and inference of LLMs are conducted on NVIDIA A100 80GB GPUs. We employ LoRA [33] and FlashAttention-2 [15] for tuning and vLLM with PageAttention [47] for inference.

6.2 DP Performance

Seen Tasks. Table 4 reports the performance on the seen tasks. GPT-4 performs the best in most cases (11 out of 19). However, its score on the CMS dataset of SM is mediocre. Jellyfish-13B wins the second most (7 out of 19) and reports the best average score due to advantage over GPT-4 on the CMS dataset. Comparing Jellyfish-13B with GPT-3, GPT-3.5, GPT-4o, and Table-GPT, Jellyfish-13B wins in more cases. Jellyfish-13B also outperforms best of non-LLMs on all unseen datasets and all but one seen datasets. Note that for non-LLM methods, all the datasets are seen because they need to be fine-tuned on them. Meanwhile, the 7B and 8B Jellyfish models also exhibit competitiveness, especially for DI and EM, and their average scores surpass best of non-LLMs and GPT-4o.

Among the four tasks, SM is the hardest task, and all the competitors report relatively low F1 score. Looking into the datasets, we find that even humans have difficulties in telling whether the two attributes match, given only name and description. To compare the methods in more detail, we report precision and recall in Table 5. The non-LLM method, SMAT, reports the highest recall, yet with a very low precision. Among its results, only 1 out of 3 – 9 is true positive. This is because

Table 6: DP performance on unseen tasks, micro-F1 for CTA and F1 for AVE. CTA is a seen task for RoBERTa. Zero-shot is used for LLMs. “–” indicates numbers not reported in prior works for this dataset.

Task	Dataset	Model									
		RoBERTa (159 shots)	RoBERTa (356 shots)	Stable Beluga 2 70B	SOLAR 70B	GPT-3.5	GPT-4	GPT-4o	Jellyfish- 7B	Jellyfish- 8B	Jellyfish- 13B
CTA	SOTAB	79.20	89.73	–	–	<u>89.47</u>	91.55	65.06	83.00	76.33	82.00
AVE	AE-110k	–	–	52.10	49.20	61.30	55.50	55.77	56.09	<u>59.55</u>	58.12
	OA-Mine	–	–	50.80	55.20	<u>62.70</u>	68.90	60.20	51.98	59.22	55.96

Table 7: Impact of instruction tuning for DP on the unseen task of CTA. “+ task” denotes the model tuned for the task.

OOP2-13B	+ ED	+ DI	+ SM	+ EM	Jellyfish-13B
56.40	74.20	79.20	76.70	71.50	82.00

Table 8: Impact of prompt engineering on the unseen task of CTA, varying options in stages and chain-of-thought (CoT) over Jellyfish-13B.

One-stage, w/o CoT	One-stage, w/ CoT	Two-stage, w/o CoT	Two-stage, w/ CoT
51.50	58.00	67.00	82.00

many SM-tailored methods seek high recall, in order to find more candidates for further verification. Jellyfish-13B exhibits relatively high precision (41% – 57%), and is close to GPT-4 on the unseen dataset of CMS. This suggests that Jellyfish-13B can be used as a verification method (1 out of 2 is true positive) on top of a filtering approach (e.g., SMAT).

Unseen Tasks. Table 6 reports the performance comparison on the unseen tasks. For CTA, GPT-4 performs the best. Jellyfish models also exhibit competitiveness, especially for the 7B and 13B models. For AVE, all the Jellyfish models showcase strong generalizability. In particular, Jellyfish-8B and Jellyfish-13B surpass the two 70B models on both datasets, and outperform GPT-4 on the AE-110k dataset.

To drill down to the impact of tuning on unseen tasks, we investigate the case of CTA with Jellyfish-13B as an example. Table 7 helps us find out which task contributes the most to this unseen task. When tuning with only one task, the model reports a micro-F1 in the range of 71% – 79%, with DI being the highest. We suppose this is because DI is exactly the inverse operation of CTA, i.e., DI fills the value of an attribute, and meanwhile CTA infers the type of an attribute given a set of sample values. Moreover, the four tasks jointly contributes to an overall micro-F1 of 82% and it surpasses the performance of tuning with only DI, showcasing the usefulness of other tasks as well.

Further, we conduct an ablation study to study the impact of prompting and report the results in Table 8. The two-stage pipeline performs better than the one-stage pipeline, and chain-of-thought, which splits the inference of column types into four steps, is also useful, in line with the observation in a previous work [44]. This demonstrates that the prompt engineering techniques developed for existing LLM-based solutions also work for Jellyfish-13B. In doing so, the design of prompts for Jellyfish-13B on unseen tasks is rendered much easier, as users may directly follow those used in existing works.

6.3 Improvement of Jellyfish over Base Models

Table 9 compares Jellyfish models and their base models on DP tasks. Consistent performance improvement is observed on all datasets for the 7B and 13B models, and on all but one dataset for the 8B model. The improvement of the 7B model is the most significant, with an average score of 35. For the 8B and 13B models, the improvement is also remarkable, with an average of 18 and 21, respectively. We also observe that the tuning benefits the performance on unseen datasets and unseen tasks. Such

Table 9: Improvement of Jellyfish over base models on DP. Zero-shot is used on seen datasets and few-shot is used on unseen datasets. All datasets are unseen for base models.

Task	Type	Dataset	Model					
			Mistral-7B	Jellyfish-7B	Llama 3-8B	Jellyfish-8B	OOP2-13B	Jellyfish-13B
ED	Seen	Adult	20.66	77.40 (+56.74)	47.42	73.74 (+26.32)	61.53	99.33 (+37.80)
		Hospital	37.09	94.51 (+57.42)	52.51	93.40 (+40.89)	63.24	95.59 (+32.35)
	Unseen	Flights	28.07	69.15 (+41.08)	67.71	66.21 (-1.50)	73.01	82.52 (+9.51)
		Rayyan	22.86	75.07 (+52.21)	62.46	81.06 (+18.64)	89.37	90.65 (+1.28)
DI	Seen	Buy	76.92	98.46 (+21.54)	86.15	98.46 (+12.31)	89.23	100 (+10.77)
		Restaurant	18.75	89.53 (+70.78)	43.02	87.21 (+44.19)	81.40	89.53 (+8.13)
	Unseen	Flipkart	79.52	87.14 (+7.62)	66.50	87.48 (+20.98)	78.49	81.68 (+3.19)
		Phone	76.72	86.52 (+9.80)	82.16	85.68 (+3.52)	84.33	87.21 (+2.88)
SM	Seen	MIMIC-III	6.90	53.33 (+46.43)	14.81	45.45 (+30.64)	36.36	40 (+3.64)
		Synthea	26.67	55.56 (+28.89)	23.52	47.06 (+23.54)	22.22	56 (+33.78)
	Unseen	CMS	0	42.86 (+42.86)	27.02	38.10 (+11.08)	13.33	59.29 (+45.96)
EM	Seen	Amazon-Google	36.51	81.69 (+45.15)	60.67	81.42 (+20.75)	36.70	81.34 (+44.64)
		Beer	69.57	100 (+30.43)	88	100 (+12)	85.71	96.77 (+11.06)
		DBLP-ACM	85.30	98.65 (+13.35)	82.14	98.77 (+16.63)	78.86	98.98 (+20.12)
		DBLP-GoogleScholar	59.54	94.88 (+35.34)	76.15	95.03 (+18.88)	59.48	98.51 (+39.03)
		Fodors-Zagats	66.67	100 (+33.33)	95.23	100 (+4.77)	92.68	100 (+7.32)
		iTunes-Amazon	70.97	96.30 (+25.33)	79.36	96.30 (+16.94)	57.45	98.11 (+40.66)
	Unseen	Abt-Buy	36.99	86.06 (+49.07)	44.60	88.84 (+44.24)	31.51	89.58 (+58.07)
		Walmart-Amazon	63.14	84.91 (+21.77)	59.69	85.24 (+25.55)	65.21	89.42 (+24.21)
CTA	Unseen	SOTAB	23.49	83.00 (+59.1)	64.25	76.33 (+12.08)	56.36	82.00 (+25.64)
AVE	Unseen	AE-110k	32.92	56.09 (+23.17)	56.33	59.55 (+3.22)	43.87	58.12 (+14.25)
		OA-Mine	32.44	51.98 (+19.54)	55.57	59.22 (+3.65)	54.81	55.96 (+1.15)
Average			44.17	80.14 (+35.97)	60.69	79.30 (+18.60)	61.60	83.21 (+21.61)

impact is the most significant on EM’s unseen data, showcasing that the knowledge injected through tuning applies well to unseen scenarios.

6.4 Impact of Knowledge Injection

To evaluate the impact of knowledge injection, we report in Table 10 the results for OOP2-13B and its tuned version with knowledge either injected or not. Comparing OOP2-13B and the one without injected knowledge, the performance is significantly raised on seen datasets but drops on a few unseen datasets. When we turn on knowledge injection, the performance further improves the performance on seen datasets and the improvement is consistent on all but two datasets. Furthermore, the improvement is also observed and more significant on unseen datasets, because like seen datasets of Amazon-Google and Beer, they are also product data. This observation suggests that the domain knowledge learned through tuning indeed enhances the model’s generalizability to unseen datasets. In addition, the impact is the most remarkable on CMS, the unseen dataset of SM, remedying the model’s performance on this dataset and making it highly competitive.

6.5 NLP Performance

Table 11 compares Jellyfish models and their original models on various NLP benchmarks [30, 84, 57, 11, 14, 107] of the Open LLM Leaderboard [20]. For the 8B and 13B models, their NLP performance roughly retains after tuning for DP, with very slight change (0.56 and 0.36 on average, respectively), and even improves on two benchmarks. The 7B model sacrifices more NLP performance (1.32 on average) for DP performance. We think this reflects the no free lunch theorem [101], considering its smallest size among the three.

6.6 Impact of Instruction Data Configuration

We study the impact of the data configuration in the instruction data. For this set of experiments, we randomly sample data from the datasets in Table 1 and disable the data preparation techniques

Table 10: Impact of knowledge injection, zero-shot. “w/o KNL” denotes the model tuned without injected knowledge in the DP task data.

Task	Type	Dataset	Model		
			OOP2-13B	Jellyfish-13B (w/o KNL)	Jellyfish-13B
ED	Seen	Adult	61.53	72	99.33
		Hospital	63.24	69.81	95.59
	Unseen	Flights	73.01	65.44	82.52
		Rayyan	89.37	76.14	90.65
DI	Seen	Buy	89.23	93.85	100
		Restaurant	81.40	88.37	89.53
	Unseen	Flipkart	78.49	82.80	81.68
		Phone	84.33	83.58	87.21
SM	Seen	MIMIC-III	36.36	46.15	40
		Synthea	22.22	53.33	56
	Unseen	CMS	13.33	7.14	59.29
EM	Seen	Amazon-Google	36.70	77.78	81.34
		Beer	85.71	93.33	96.77
		DBLP-ACM	78.86	97.36	98.98
		DBLP-GoogleScholar	59.48	93.10	98.51
		Fodors-Zagats	92.68	100	100
		iTunes-Amazon	57.45	93.10	94.55
	Unseen	Abt-Buy	31.51	86.29	89.58
		Walmart-Amazon	65.21	74.15	89.42

Table 11: NLP performance on the Open LLM Leaderboard.

Size	Model	MMLU (5-shot)	WinoGrande (0-shot)	ARC (25-shot)	TruthfulQA (0-shot)	GSM8K (8-shot)	HellaSwag (10-shot)	Average
7B	Mistral-7B	62.91	73.88	63.48	66.91	41.32	84.79	65.55
	Jellyfish-7B	62.08 (-0.83)	72.69 (-1.19)	63.48 (+0.00)	64.76 (-2.15)	37.91 (-3.41)	84.48 (-0.31)	64.23 (-1.32)
8B	Llama 3-8B	64.51	71.74	61.01	51.63	70.36	78.61	66.31
	Jellyfish-8B	64.23 (-0.28)	72.06 (+0.32)	60.15 (-0.14)	51.83 (+0.20)	69.29 (-1.07)	77.92 (-0.69)	65.76 (-0.56)
13B	OOP2-13B	54.49	74.03	62.63	52.56	25.32	83.24	58.71
	Jellyfish-13B	53.04 (-1.45)	74.19 (+0.16)	62.88 (+0.25)	52.56 (+0.00)	24.26 (-1.06)	83.16 (-0.08)	58.35 (-0.36)

regarding positives and missing values (Section 3.1) to see the impact of dataset size clearly.

To simplify the evaluation, we tune the 13B model with data for a single DP task and evaluate its effect. By varying the amount of data, Figure 3 displays how the tuning data for a specific task affects the DP performance. In general, the four tasks are all useful in improving the overall performance. For intra-task performance (e.g., ED to ED), as expected, the tuning data has a significantly positive impact. For inter-task performance, ED and SM are generally positive to other tasks, while DI and EM report negative effects. Such impact on the overall DP performance is also observed when we increase the amount of tuning data (e.g., doubling EM from 21k to 43k). We also find that DI can benefit from all the other three tasks. We think this is because the other three tasks all contain correct values for the attributes, thereby enhancing the model’s ability in filling missing values. In addition, the benefit of increasing tuning data for SM is obvious. Overall, these observations results in the data configuration in constructing Jellyfish (Section 3.1).

We study the impact of tuning the 13B model with multi-task data and plot the results in Figure 4. By feeding the tuning set with data for more tasks, it is obvious that they jointly contribute to better DP performance, and the improvement is consistent. When fully utilized the data, as indicated by (1, 1, 1, 1), it achieves the best performance. Based on the above results, we construct the Jellyfish

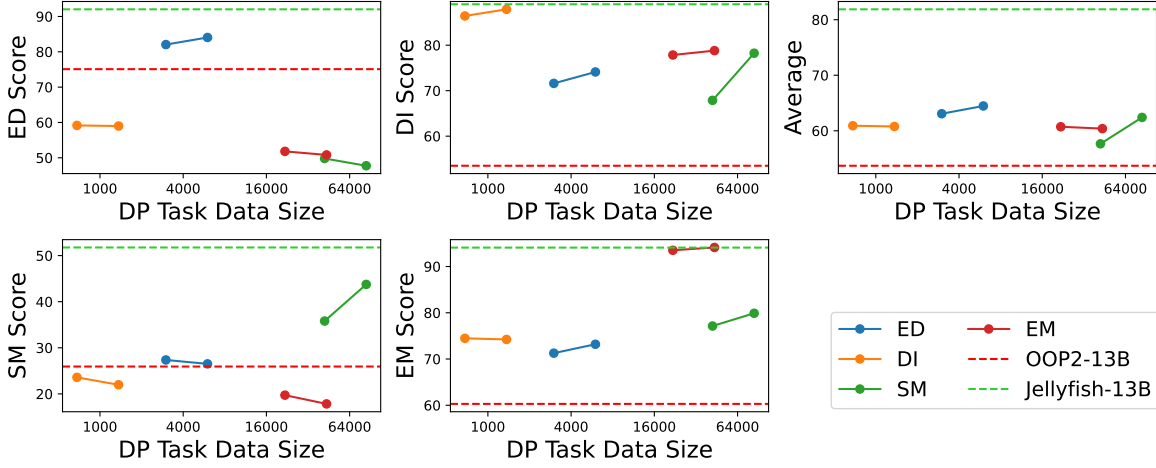


Figure 3: Impact of tuning with single-task data on DP performance, zero-shot. Above red line is positive.

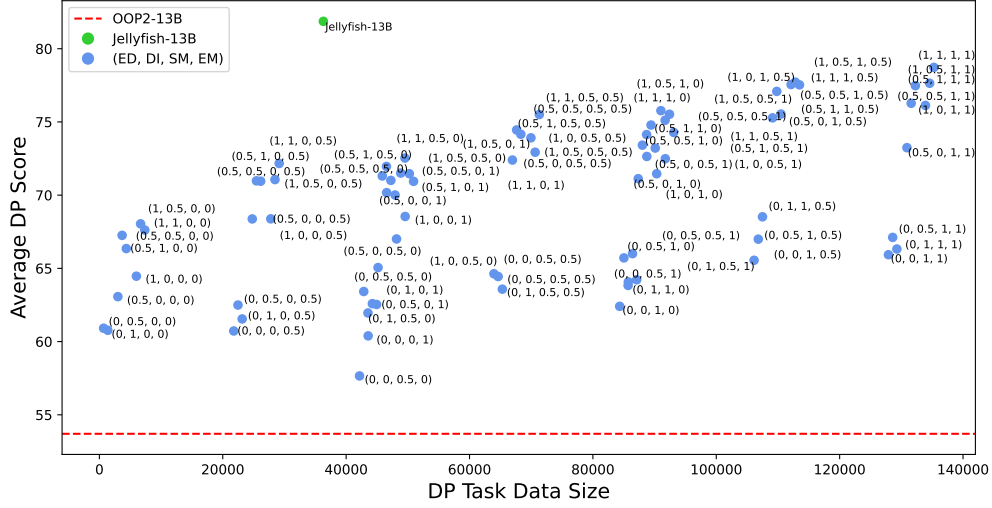


Figure 4: Impact of tuning with multi-task data on DP performance. Numbers in parenthesis indicate the percentage of data used for each task.

data by appropriately choosing the size of data for each task. Moreover, with the data preparation techniques (Section 3.1) applied, Jellyfish-13B, even with less amount of tuning data, performs better than (1, 1, 1, 1) in Figure 4.

Then, we evaluate how the data for a specific DP task affects the NLP performance and report the results in Figure 5. In general, ED and EM exhibit positive impacts on the overall NLP performance. By increasing the amount of tuning data, all the tasks, except DI, are positive to NLP tasks. Specifically, SM turns from negative to positive when the dataset size is doubled, whereas the trend for DI is reversed, resulting in a significant drop. To drill down to each benchmark, all the four tasks are positive to WinoGrande, while they are generally negative to MMLU, and neutral to the other

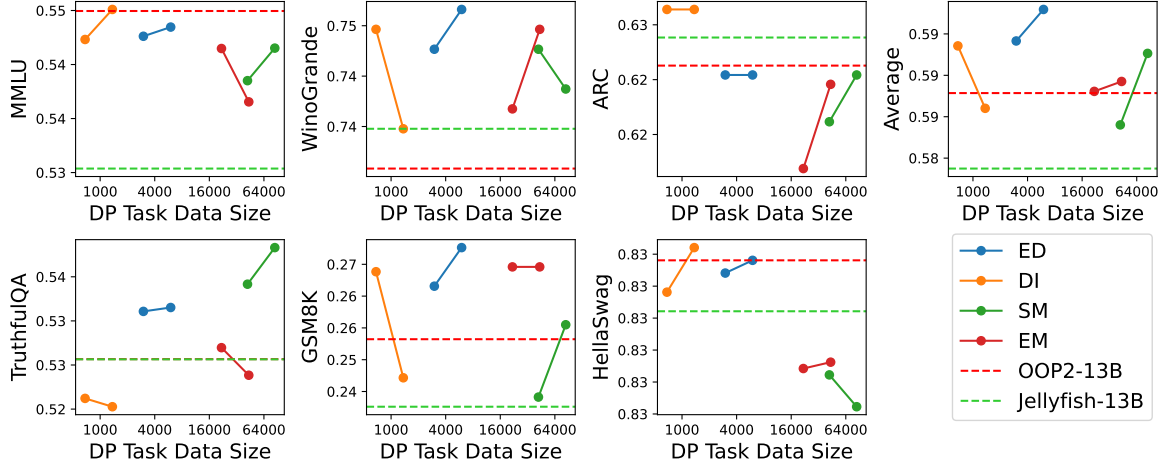


Figure 5: Impact of tuning with single-task data on NLP performance. Above red line is positive.

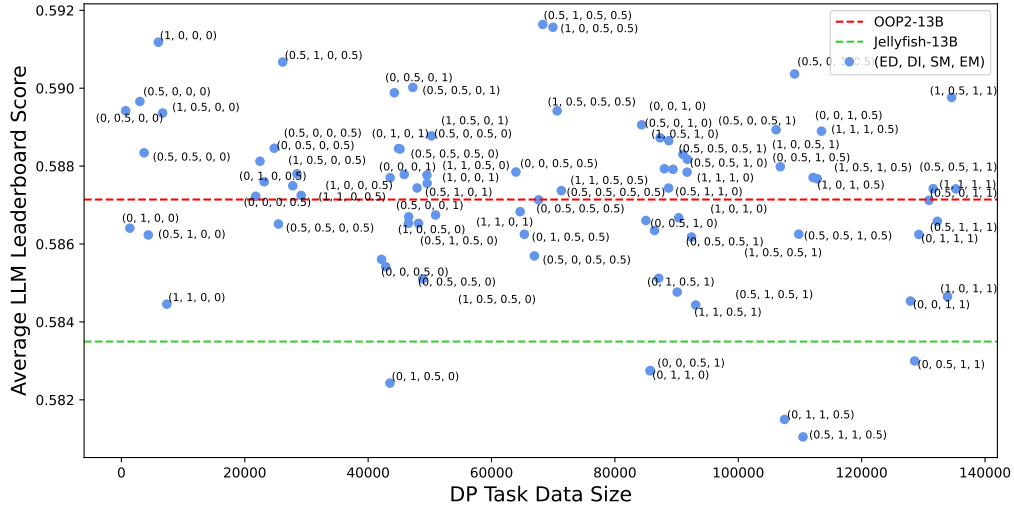


Figure 6: Impact of tuning with multi-task data on NLP performance. Numbers in parenthesis indicate the percentage of data used for each task.

benchmarks, roughly in line with the results in Table 11. This experiment indicates that we need to choose an appropriate data size for each DP task, specifically, with moderately less data for DI, to prevent the model from losing its NLP capability.

We also test the impact of tuning the 13B model with multi-task data on its NLP performance over the six benchmarks used in Table 11. The results are reported in Figure 6. The general trend is that with data for more tasks, the NLP performance has a drop, yet this change, as shown in more sporadic points, is less consistent than what we observed in Figure 4. It is noteworthy that the overall decrease in NLP performance is moderate, with an average of 0.36 (from 58.71 to 58.35) for Jellyfish-13B.

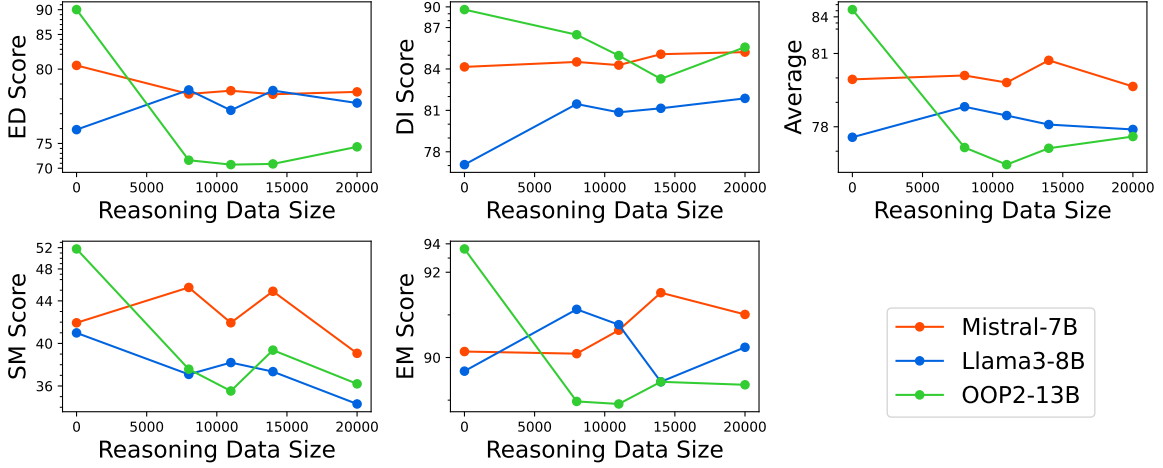


Figure 7: Impact of reasoning data on DP performance, zero-shot, plotted in log scale to show trends clearly.

6.7 Impact of Reasoning Data

Figure 7 shows how reasoning data, varying from 8k, 11k, 14k, to 20k instances, impacts the DP performance. For the 7B and 8B models, the average scores increase first and then drop when more reasoning data is used for tuning, suggesting that small amount of reasoning data – with the rationale behind DP – can enhance the model’s DP performance. Seeing this, we choose 14k and 8k for the two models, respectively, as the reasoning data size for tuning, striking a balance for the overall performance. For the 13B model, the scores drastically reduce and then rebound with more reasoning data. This may be attributed to the reasoning and logic capabilities of OOP2-13B, which are intended to enhance those of Llama 2 but ultimately do not align well with the underlying logic of DP. Only when the DP reasoning data reaches 20k, the model learns to handle DP well with reasoning. Nonetheless, the scores are still below those without reasoning, and thus we choose not to tune the 13B model with reasoning data.

6.8 Evaluation of Interpretation

We evaluate the performance of Jellyfish’s 7B and 8B models and compare them with GPT-3.5 (gpt-3.5-turbo-0613). Given an answer output by Jellyfish, we generate reasons using both Jellyfish and GPT-3.5, and request GPT-4o to decide which one is better. Note that GPT-4o is unaware of the correct answer to the question in DP. As such, it needs to judge by its own analysis of the question as well.

Table 12 reports the head-to-head comparison of the two models. The results show that Jellyfish-7B is almost on a par with GPT-3.5, while Jellyfish-8B is a much better interpreter. In particular, Jellyfish-7B wins over GPT-3.5 on 3 out of 11 datasets, reporting an overall winning rate of 46% across 220 tests. Jellyfish-8B defeats GPT-3.5 on 10 out of 11 datasets, achieving a winning rate of 73%. To delve into detailed comparison of Jellyfish-8B and GPT-3.5, we show an example on the Amazon-Google dataset in EM. The prompt is as follows.

Table 12: Head-to-head comparison of GPT-3.5 and Jellyfish-7B/8B on interpretation, judged by GPT-4o. The two comparisons share the same sets of questions and the same answers from GPT-3.5.

Task	Dataset	Comparison 1		Comparison 2	
		GPT-3.5	Jellyfish-7B	GPT-3.5	Jellyfish-8B
ED	Adult	17	3	4	16
	Hospital	4	16	4	16
DI	Buy	4	16	4	16
	Restaurant	10	10	9	11
SM	Synthea	15	5	3	17
EM	Amazon-Google	3	17	2	18
	Beer	13	7	7	13
	DBLP-ACM	11	9	2	18
	DBLP-GoogleScholar	16	4	9	11
	Fodors-Zagats	13	7	13	7
	iTunes-Amazon	12	8	2	18
Total		118	102	59	161
Winning Rate		53.63%	46.36%	26.81%	73.18%

Head-to-Head Instance

You are an AI assistant that follows instruction extremely well. The user will give you a question. Your task is to answer as faithfully as you can. While answering, provide detailed explanation and justify your answer.

Answer your reason first, then finish your response in a separate line with and ONLY with your final answer.

Product A: [name: "adobe dreamweaver cs3 [mac]", manufacturer: "adobe", price: "399.0"]

Product B: [name: "adobe cs3 web premium", price: "1659.99"]

Are Product A and Product B the same Product?

The reason given by Jellyfish-8B is as follows.

No, Product A and Product B are not the same product. Although both products are manufactured by Adobe, they have different names and prices. Product A is named "adobe dreamweaver cs3 [mac]" and is priced at \$399.00. On the other hand, Product B is named "adobe cs3 web premium" and is priced at \$1659.99. The difference in price and name suggests that these are two distinct products with different features and functionalities.

In addition, the name of Product A includes the specific software name "dreamweaver cs3" and the operating system it is designed for, "mac". This suggests that Product A is a specific version of the Dreamweaver software for Mac users. In contrast, Product B is named "cs3 web premium", which suggests that it is a more comprehensive package that includes multiple Adobe software products, rather than just a single software application.

Therefore, based on the available information, it is clear that Product A and Product B are not the same product.

Final Answer: No, Product A and Product B are not the same product."

The reason given by GPT-3.5 is as follows.

No, Product A and Product B are not the same product. The reason is that although both products are from Adobe and are part of the CS3 suite, they have different names and prices. Product A is specifically named "Adobe Dreamweaver CS3 [Mac]" and is priced at \$399.0, while Product B is named "Adobe CS3 Web Premium" and is priced at \$1659.99. The difference in names and prices indicates that they are different products within the CS3 suite, likely with different features and functionalities.

Therefore, Product A and Product B are not the same product.

Final answer: No.

GPT-4o compares the outputs of both models in the following aspects, where Model 1 refers to Jellyfish-8B and Model 2 refers to GPT-3.5: (1) Faithfulness to Instruction: Model 1: Provides a detailed explanation, comparing the names, prices, and inferred functionalities of both products. It also elaborates on the specific nature of Product A (Dreamweaver for Mac) versus the more comprehensive package suggested by Product B (Web Premium). Model 2: Also states that the products are different, citing differences in names and prices. However, it provides less detail and does not delve into the possible functionalities or the specific nature of the products as much as Model 1 does. (2) Justification: Model 1: Justifies the conclusion by explaining the implications of the product names and prices. It infers that Product A is a specific software for Mac users, while Product B is a more comprehensive suite, which is a logical and well-supported argument. Model 2: Justifies the conclusion by pointing out the differences in names and prices but does not provide as thorough an analysis of what those differences imply about the products. (3) Clarity and Completeness: Model 1: The response is clear, detailed, and logically structured. It ensures that the user understands why the products are different. Model 2: The response is clear but lacks the depth and completeness found in Model 1’s explanation. (4) Conclusion: Model 1: Concludes with a clear and well-supported final answer. Model 2: Concludes with a clear final answer but lacks the depth of reasoning provided by Model 1. (5) Winner: Model 1. Based on the analysis, GPT-4o decides that Jellyfish-8B performed better because it adhered more strictly to the original instruction by providing a clear, concise response without unnecessary repetition. In addition, with its built-in knowledge, Jellyfish-8B pointed out the difference in functionalities, whereas GPT-3.5 merely described the difference on the surface.

Furthermore, we find out that when reasoning, GPT-3.5 even fails to respond with a correct answer of matching or not for EM, as shown in its mediocre performance on datasets like Amazon-Google and DBLP-GoogleScholar in Table 4. In contrast to the above example of Jellyfish-8B’s landslide win, GPT-3.5 only has a slight edge when it wins. For instance, in an example of the Amazon-Google dataset, GPT-4o points out that GPT-3.5 has more focused justification and additional insights into the implications of the differences between the products, yet it also mentioned that GPT-3.5’s repetition of the final answer is a minor deviation from the instruction’s format.

6.9 Comparison of Efficiency

With 8 GPUs of A100 80G, instruction tuning spends around 5 hours for Jellyfish-13B, 3 hours for Jellyfish-7B and Jellyfish-8B. For inference on single GPU of A100 80G, Jellyfish-7B, 8B, and 13B spend 0.07, 0.08, and 0.15 seconds, respectively, on average to process an instance. As a reference, GPT-4 spends an average of 1 – 8 seconds per instance. Although LLMs require substantial computational resources, thereby increasing the cost of use and compromising the efficiency, some non-LLM methods, such as RoBERTa and those built upon it (e.g., IPM), need fine-tuning when applied to unseen datasets. This fine-tuning time should be counted towards total time expense for fair comparison. Moreover, advanced learning techniques enables Jellyfish models to be quantized [59] or distilled to improve efficiency, which will be considered in the future.

7 Related Works

Since works on LLMs have been introduced in Section 2.2, we briefly review related works on DP here. *Seen Tasks.* The tasks targeted in this paper collectively form the most critical part of DP, and they have been extensively studied.

- **ED:** Traditional methods mainly depend on hand-crafted rules [12], pattern discovery [13], outlier detection [76], or statistical modeling [34, 97]. Recent works employ more advanced ML techniques such as few-shot learning based on a noisy channel model (HoloDetect) [28], or resort to a series of ML pipelines (Raha) [65], including feature engineering, clustering, and classification.

- **DI:** While rule-based solutions [80, 88] remain one of the prevalent approaches, another stream of works develops ML models for this task, including variational autoencoders [72], generative adversarial networks [106], and attention mechanisms [102, 92]. To seek better imputation performance, recent progress utilizes PLMs to capture semantics [66].
- **SM:** The use of similarity matrices is a traditional way [83]. More advanced methods utilize ML techniques [23], including deep learning models [87]. SMAT [109] is an approach leveraging attention-based deep learning. A recent attempt employs GPT-4 for SM [86].
- **EM:** The procedure is divided into blocking and in-block pairwise matching for the sake of efficiency. Blocking groups pairs of entities that potentially match into the same block, and then pairwise matching is performed within each block to find matching entities. Traditional solutions for blocking mostly rely on attribute equivalence, hashes, or similarities [74]. Recently, the feasibility of using DL methods for blocking has also been examined [91], following the use of DL for pairwise matching [69]. In addition, there are tools that handle both steps such as Megallan [43] and Ditto [56]. A recent evaluation validates the effectiveness of in-context learning in enhancing LLMs’ EM performance [75].

Unseen Tasks. We review the related studies on CTA and AVE.

- **CTA:** As a typical table understanding task, it often appears in the studies on table representation learning [36, 17, 89]. These approaches fine-tune PLMs, typically BERT [18] and its variants [38, 61]. Recently, ChatGPT has been utilized to solve this task [44].
- **AVE:** Early approaches employ LSTM-CRF [46, 116]. With the prevalence of PLMs, like CTA, many solutions to AVE resort to using BERT [105, 98, 117]. A recent work [4] considered fine-tuning GPT-3.5 and prompting GPT-4, and compared with open-source LLMs like Stable Beluga 2 [63] and SOLAR [95].

Generic Solution. Whereas the above solutions are specialized for a task, recent progress developed generic solutions to DP based on GPT-3 [70], or GPT-3.5 and GPT-4 [108], basically employing various prompt engineering techniques on frozen LLMs. Fine-tuning GPT-3.5 and ChatGPT for a variety of table-related tasks has also been investigated [53], and several DP tasks are covered.

Other DP Tasks. Besides the ones covered by this paper, there are many other DP tasks. We name a few examples.

- Data repairing corrects erroneous values in a dataset. Typical solutions are HoloClean [80] and Baran [64]. HoloClean can detect errors and perform repairing subsequently. Baran only repair errors and resort to Raha to detect errors. Recent advancements [50, 78] utilized Bayesian inference to capture dependencies between attributes.
- Data fusion is the process of integrating multiple data sources that contain information about the same set of entities, with possibly conflicting attribute values. Surveys of early attempts are available [54, 6], with a detailed comparison of various fusion methods on deep web data [54]. More recent endeavors targeted multi-truth data fusion [2] and golden record [29].
- Data transformation is the process of converting data from one format into another format. Notable approaches are transformation by user-specified examples [27] and learning from large collections of paired table columns [39]. In addition, the aforementioned generic DP solution also covers this task [70].

Data Preparation. DP is also studied in the name of data preparation, which manipulates raw data into a form that can be readily analyzed. A notable Python library is DataPrep [48]. In addition to the DP tasks listed above, data augmentation [10, 68, 60, 115] is another key operation in data

preparation. Another line of work studies dataset discovery [3, 45, 21, 71], particularly for integrating data lake tables [41] where joinable [19], unionable [40], and related table search [112] are often used for identifying candidates. Despite search speed being a key concern, LLMs are anticipated to be used on top of their outcomes for automated integration in a data lake [1].

8 Conclusions

Contributions. We studied the problem of instruction-tuning LLMs as universal DP task solvers. By devising data preparation and knowledge injection techniques, we proposed Jellyfish, which enables users to craft instructions for DP tasks. Another notable feature of Jellyfish is its interpretation ability, providing explanations of its outputs. We tuned three base models ranging from 7B to 13B, which can operate on a local GPU without compromising data security. The experiments demonstrated the competitiveness of Jellyfish against existing DP solutions, impressive generalizability to new tasks, the ability of retaining performance in NLP tasks, as well as the competence in interpretation.

Limitations. We investigated six DP tasks, whereas there are still many other tasks (e.g., data repairing, data fusion, and data transformation).

We discovered that our reasoning data compromises the 13B model’s DP performance, possibly because OpenOrca-Platypus2-13B’s reasoning and logic do not align well with the underlying logic of DP. In contrast, the 7B and 8B models, derived from native models Mistral-7B and Llama 3-8B, respectively, can benefit from the use of reasoning data. Due to the lack of (approximately) 13B size for these two base models, we cannot deliver better DP performance than the 13B model while preserving the interpretation ability. Nonetheless, we believe that our instruction data can apply to more advanced base models, with which better DP performance and interpretation ability could be both achieved.

Another limitation is that our prompt is designed as instance-based, rather than the table-based setting which was adopted in many non-LLM approaches. This is partially due to the token limitation of the LLMs we used (e.g., 4096 tokens for a 13B model), and compromises efficiency when we use our models for large-scale datasets. In our future work, we will consider designing prompts that take multiple instance or a table at a time.

Furthermore, when using our models for practical data mining pipelines, we also need to carefully consider the issues of preprocessing and postprocessing. For example, the input data may be scanned copies and contain hierarchical tables, while we focus on relational tables in this work.

Future Work. Our future work aims to address the limitations outlined above, in particular, expanding the instruction data of Jellyfish to encompass more DP tasks, such as data repairing and data transformation. Furthermore, our future research directions include the development of a quantized or distilled model to enhance processing speed, as well as a multi-agent system for adaptable, conversational, code-free DP pipeline.

Acknowledgements

This work is supported by NEC Corporation and JSPS Kakenhi 23K17456, 23K25157, 23K28096, and JST CREST JPMJCR22M2. We thank Prof. Makoto Onizuka, Yuki Arase, and Yuya Sasaki for providing equipment support for completing this research.

References

- [1] S. Arora, B. Yang, S. Eyuboglu, A. Narayan, A. Hojel, I. Trummer, and C. Ré. Language models enable simple systems for generating structured views of heterogeneous data lakes. *arXiv preprint arXiv:2304.09433*, 2023.

- [2] F. Azzalini, D. Piantella, E. Rabosio, and L. Tanca. Enhancing domain-aware multi-truth data fusion using copy-based source authority and value similarity. *The VLDB Journal*, 32(3):475–500, 2023.
- [3] A. Bogatu, A. A. Fernandes, N. W. Paton, and N. Konstantinou. Dataset discovery in data lakes. In *ICDE*, pages 709–720. IEEE, 2020.
- [4] A. Brinkmann, R. Shraga, and C. Bizer. Product attribute value extraction using large language models. *arXiv preprint arXiv:2310.12537*, 2023.
- [5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *NeurIPS*, 33:1877–1901, 2020.
- [6] G. K. Canalle, A. C. Salgado, and B. F. Loscio. A survey on data fusion: what for? in what form? what is next? *Journal of Intelligent Information Systems*, 57:25–50, 2021.
- [7] A. Chapman, E. Simperl, L. Koesten, G. Konstantinidis, L.-D. Ibáñez, E. Kacprzak, and P. Groth. Dataset search: a survey. *The VLDB Journal*, 29(1):251–272, 2020.
- [8] L. Cheng, X. Li, and L. Bing. Is GPT-4 a good data analyst? *arXiv preprint arXiv:2305.15038*, 2023.
- [9] Z. Cheng, J. Kasai, and T. Yu. Batch prompting: Efficient inference with large language model APIs. *arXiv preprint arXiv:2301.08721*, 2023.
- [10] N. Chepurko, R. Marcus, E. Zraggen, R. C. Fernandez, T. Kraska, and D. Karger. ARDA: automatic relational data augmentation for machine learning. *arXiv preprint arXiv:2003.09758*, 2020.
- [11] F. Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.
- [12] X. Chu, I. F. Ilyas, and P. Papotti. Holistic data cleaning: Putting violations into context. In *ICDE*, pages 458–469. IEEE, 2013.
- [13] X. Chu, J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang, and Y. Ye. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*, pages 1247–1261, 2015.
- [14] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [15] T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.
- [16] S. Das, A. Doan, P. S. G. C., C. Gokhale, P. Konda, Y. Govind, and D. Paulsen. The magellan data repository. <https://sites.google.com/site/anhaidgroup/useful-stuff/the-magellan-data-repository>.
- [17] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu. TURL: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40, 2022.
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [19] Y. Dong, C. Xiao, T. Nozawa, M. Enomoto, and M. Oyamada. DeepJoin: Joinable table discovery with pre-trained language models. *arXiv preprint arXiv:2212.07588*, 2022.

- [20] H. Face. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2024.
- [21] G. Fan, J. Wang, Y. Li, D. Zhang, and R. Miller. Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning. *arXiv preprint arXiv:2210.01922*, 2022.
- [22] Flipkart.com. Flipkart products. <https://www.kaggle.com/datasets/PromptCloudHQ/flipkart-products>.
- [23] A. Gal, H. Roitman, and R. Shraga. Learning to rerank schema matches. *IEEE Transactions on Knowledge and Data Engineering*, 33(8):3104–3116, 2019.
- [24] C. Gao, X. Lan, N. Li, Y. Yuan, J. Ding, Z. Zhou, F. Xu, and Y. Li. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *arXiv preprint arXiv:2312.11970*, 2023.
- [25] R. Hai, C. Koutras, C. Quix, and M. Jarke. Data lakes: A survey of functions and systems. *IEEE Transactions on Knowledge and Data Engineering*, 2023.
- [26] J. Han, J. Pei, and H. Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.
- [27] Y. He, X. Chu, K. Ganjam, Y. Zheng, V. Narasayya, and S. Chaudhuri. Transform-data-by-example (TDE) an extensible search engine for data transformations. *PVLDB*, 11(10):1165–1177, 2018.
- [28] A. Heidari, J. McGrath, I. F. Ilyas, and T. Rekatsinas. HoloDetect: Few-shot learning for error detection. In *SIGMOD*, pages 829–846, 2019.
- [29] A. Heidari, G. Michalopoulos, I. F. Ilyas, and T. Rekatsinas. Record fusion via inference and data augmentation. *ACM/JMS Journal of Data Science*, 2023.
- [30] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [31] S. Hong, X. Zheng, J. Chen, Y. Cheng, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, et al. MetaGPT: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.
- [32] N. Houlsby, A. Giurugu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, pages 2790–2799. PMLR, 2019.
- [33] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [34] Z. Huang and Y. He. Auto-detect: Data-driven error detection in tables. In *SIGMOD*, pages 1377–1392, 2018.
- [35] Hugging Face. Llama and llama 2 variants. <https://huggingface.co/models?other=llama>, 2023.
- [36] H. Iida, D. Thai, V. Manjunatha, and M. Iyyer. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584*, 2021.
- [37] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

- [38] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.
- [39] Z. Jin, Y. He, and S. Chauduri. Auto-transform: learning-to-transform by patterns. *PVLDB*, 13(12):2368–2381, 2020.
- [40] A. Khatiwada, G. Fan, R. Shraga, Z. Chen, W. Gatterbauer, R. J. Miller, and M. Riedewald. SANTOS: Relationship-based semantic table union search. *SIGMOD*, 1(1):1–25, 2023.
- [41] A. Khatiwada, R. Shraga, W. Gatterbauer, and R. J. Miller. Integrating data lake tables. *PVLDB*, 16(4):932–945, 2022.
- [42] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.
- [43] P. Konda, S. Das, A. Doan, A. Ardlan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, S. Prasad, et al. Magellan: toward building entity matching management systems over data science stacks. *PVLDB*, 9(13):1581–1584, 2016.
- [44] K. Korini and C. Bizer. Column type annotation using ChatGPT. *arXiv preprint arXiv:2306.00745*, 2023.
- [45] C. Koutras, G. Siachamis, A. Ionescu, K. Psarakis, J. Brons, M. Fragkoulis, C. Lofi, A. Bonifati, and A. Katsifodimos. Valentine: Evaluating matching techniques for dataset discovery. In *ICDE*, pages 468–479. IEEE, 2021.
- [46] Z. Kozareva, Q. Li, K. Zhai, and W. Guo. Recognizing salient entities in shopping queries. In *ACL*, pages 107–111, 2016.
- [47] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *SOSP*, 2023.
- [48] S. F. U. D. S. Lab. DataPrep. <https://dataprep.ai/>.
- [49] A. N. Lee, C. J. Hunter, N. Ruiz, B. Goodson, W. Lian, G. Wang, E. Pentland, A. Cook, C. Vong, and "Teknium". OpenOrcaPlatypus: Llama2-13B model instruct-tuned on filtered OpenOrcaV1 GPT-4 dataset and merged with divergent STEM and logic dataset model. <https://huggingface.co/Open-Orca/OpenOrca-Platypus2-13B>, 2023.
- [50] A. Lew, M. Agrawal, D. Sontag, and V. Mansinghka. PClean: Bayesian data cleaning at scale with domain-specific probabilistic programming. In *AISTATS*, pages 1927–1935. PMLR, 2021.
- [51] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *NeurIPS*, 33:9459–9474, 2020.
- [52] H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022.
- [53] P. Li, Y. He, D. Yashar, W. Cui, S. Ge, H. Zhang, D. R. Fainman, D. Zhang, and S. Chaudhuri. Table-GPT: Table-tuned GPT for diverse table tasks. *arXiv preprint arXiv:2310.09263*, 2023.
- [54] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *arXiv preprint arXiv:1503.00303*, 2015.
- [55] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

- [56] Y. Li, J. Li, Y. Suhara, A. Doan, and W.-C. Tan. Deep entity matching with pre-trained language models. *PVLDB*, 14(1):50–60, 2020.
- [57] S. Lin, J. Hilton, and O. Evans. TruthfulQA: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.
- [58] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.
- [59] S.-y. Liu, Z. Liu, X. Huang, P. Dong, and K.-T. Cheng. LLM-FP4: 4-bit floating-point quantized transformers. *arXiv preprint arXiv:2310.16836*, 2023.
- [60] T. Liu, J. Fan, Y. Luo, N. Tang, G. Li, and X. Du. Adaptive data augmentation for supervised learning over missing data. *PVLDB*, 14(7):1202–1214, 2021.
- [61] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [62] W. Lu, J. Zhang, J. Zhang, and Y. Chen. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*, 2024.
- [63] D. Mahan, R. Carlow, L. Castricato, N. Cooper, and C. Laforce. Stable beluga 2. <https://huggingface.co/stabilityai/StableBeluga2>, 2023.
- [64] M. Mahdavi and Z. Abedjan. Baran: Effective error correction via a unified context representation and transfer learning. *PVLDB*, 13(12):1948–1961, 2020.
- [65] M. Mahdavi, Z. Abedjan, R. Castro Fernandez, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang. Raha: A configuration-free error detection system. In *SIGMOD*, pages 865–882, 2019.
- [66] Y. Mei, S. Song, C. Fang, H. Yang, J. Fang, and J. Long. Capturing semantics for imputation with pre-trained language models. In *ICDE*, pages 61–72. IEEE, 2021.
- [67] Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. <https://ai.meta.com/blog/meta-llama-3/>, 2024.
- [68] Z. Miao, Y. Li, and X. Wang. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In *SIGMOD*, pages 1303–1316, 2021.
- [69] 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 *SIGMOD*, pages 19–34, 2018.
- [70] A. Narayan, I. Chami, L. Orr, and C. Ré. Can foundation models wrangle your data? *PVLDB*, 16(4):738–746, 2022.
- [71] F. Nargesian, K. Pu, B. Ghadiri-Bashardoost, E. Zhu, and R. J. Miller. Data lake organization. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):237–250, 2022.
- [72] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling incomplete heterogeneous data using VAEs. *Pattern Recognition*, 107:107501, 2020.
- [73] OpenAI. March 20 ChatGPT outage: Here’s what happened, 2023.
- [74] G. Papadakis, D. Skoutas, E. Thanos, and T. Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys*, 53(2):1–42, 2020.

- [75] R. Peeters and C. Bizer. Entity matching using large language models. *arXiv preprint arXiv:2310.11244*, 2023.
- [76] N. Prokoshyna, J. Szlichta, F. Chiang, R. J. Miller, and D. Srivastava. Combining quantitative and logical data cleaning. *PVLDB*, 9(4):300–311, 2015.
- [77] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 2023.
- [78] J. Qin, S. Huang, Y. Wang, J. Zhu, Y. Zhang, Y. Miao, R. Mao, M. Onizuka, and C. Xiao. BClean: A bayesian data cleaning system. *arXiv preprint arXiv:2311.06517*, 2023.
- [79] S. Razniewski, A. Yates, N. Kassner, and G. Weikum. Language models as or for knowledge bases. *arXiv preprint arXiv:2110.04888*, 2021.
- [80] T. Rekatsinas, X. Chu, I. F. Ilyas, and C. Ré. HoloClean: Holistic data repairs with probabilistic inference. *PVLDB*, 10(10):1190–1201, 2017.
- [81] A. Reviews. Amazon reviews: Unlocked mobile phones. <https://www.kaggle.com/datasets/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones>.
- [82] D. Ritze, O. Lehmberg, Y. Oulabi, and C. Bizer. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *WWW*, pages 251–261, 2016.
- [83] T. Sagi and A. Gal. Schema matching prediction with applications to data source discovery and dynamic ensembling. *The VLDB Journal*, 22:689–710, 2013.
- [84] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- [85] J. Savelka, K. D. Ashley, M. A. Gray, H. Westermann, and H. Xu. Can gpt-4 support analysis of textual data in tasks requiring highly specialized domain expertise? *arXiv preprint arXiv:2306.13906*, 2023.
- [86] E. Sheerit, M. Brief, M. Mishaeli, and O. Elisha. Rematch: Retrieval enhanced schema matching with llms. *arXiv preprint arXiv:2403.01567*, 2024.
- [87] R. Shraga, A. Gal, and H. Roitman. Adnev: Cross-domain schema matching using deep similarity matrix adjustment and evaluation. *PVLDB*, 13(9):1401–1415, 2020.
- [88] S. Song, Y. Sun, A. Zhang, L. Chen, and J. Wang. Enriching data imputation under similarity rule constraints. *IEEE transactions on knowledge and data engineering*, 32(2):275–287, 2018.
- [89] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, and W.-C. Tan. Annotating columns with pre-trained language models. In *SIGMOD*, pages 1493–1503, 2022.
- [90] D. Tang, Z. Chen, K. Kim, Y. Song, H. Tian, S. Ezzini, Y. Huang, and J. K. T. F. Bissyande. Collaborative agents for software engineering. *arXiv preprint arXiv:2402.02172*, 2024.
- [91] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, and A. Doan. Deep learning for blocking in entity matching: a design space exploration. *PVLDB*, 14(11):2459–2472, 2021.
- [92] S. Tihon, M. U. Javaid, D. Fourure, N. Posocco, and T. Peel. DAEMA: Denoising autoencoder with mask attention. In *ICANN*, pages 229–240, 2021.
- [93] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

- [94] J. Tu, J. Fan, N. Tang, P. Wang, G. Li, X. Du, X. Jia, and S. Gao. Unicorn: A unified multi-tasking model for supporting matching tasks in data integration. *Proceedings of the ACM on Management of Data*, 1(1):1–26, 2023.
- [95] Upstage. Solar-0-70b-16bit. <https://huggingface.co/upstage/SOLAR-0-70b-16bit>, 2023.
- [96] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.
- [97] P. Wang and Y. He. Uni-detect: A unified approach to automated error detection in tables. In *SIGMOD*, pages 811–828, 2019.
- [98] Q. Wang, L. Yang, B. Kanagal, S. Sanghai, D. Sivakumar, B. Shu, Z. Yu, and J. Elsas. Learning to extract attribute value from product via question answering: A multi-task approach. In *KDD*, pages 47–55, 2020.
- [99] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- [100] L. Weng. Prompt engineering. <https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/>, 2023.
- [101] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.
- [102] R. Wu, A. Zhang, I. Ilyas, and T. Rekatsinas. Attention-based learning for missing data imputation in HoloClean. *MLSys*, 2:307–325, 2020.
- [103] Z. Wu, R. Peng, X. Han, S. Zheng, Y. Zhang, and C. Xiao. Smart agent-based modeling: On the use of large language models in computer simulations. *arXiv preprint arXiv:2311.06330*, 2023.
- [104] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*, 2023.
- [105] H. Xu, W. Wang, X. Mao, X. Jiang, and M. Lan. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *ACL*, pages 5214–5223, 2019.
- [106] J. Yoon, J. Jordon, and M. Schaar. GAIN: Missing data imputation using generative adversarial nets. In *ICML*, pages 5689–5698, 2018.
- [107] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- [108] H. Zhang, Y. Dong, C. Xiao, and M. Oyamada. Large language models as data preprocessors. *arXiv preprint arXiv:2308.16361*, 2023.
- [109] J. Zhang, B. Shin, J. D. Choi, and J. C. Ho. SMAT: An attention-based deep learning solution to the automation of schema matching. In *ADBIS*, pages 260–274. Springer, 2021.
- [110] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.
- [111] T. Zhang, X. Yue, Y. Li, and H. Sun. Tablellama: Towards open large generalist models for tables. *arXiv preprint arXiv:2311.09206*, 2023.

- [112] Y. Zhang and Z. G. Ives. Finding related tables in data lakes for interactive data science. In *SIGMOD*, pages 1951–1966, 2020.
- [113] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, et al. Siren’s song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.
- [114] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.
- [115] Z. Zhao and R. Castro Fernandez. Leva: Boosting machine learning performance with relational embedding data augmentation. In *SIGMOD*, pages 1504–1517, 2022.
- [116] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li. Opentag: Open attribute value extraction from product profiles. In *KDD*, pages 1049–1058, 2018.
- [117] T. Zhu, Y. Wang, H. Li, Y. Wu, X. He, and B. Zhou. Multimodal joint attribute prediction and value extraction for e-commerce product. *arXiv preprint arXiv:2009.07162*, 2020.

A Experimental Setup

The following hyperparameters are used for tuning Jellyfish models:

- `lora_target`: q-proj, k-proj, v-proj, o-proj;
- `per_device_train_batch_size`: 2;
- `gradient_accumulation_steps`: 2;
- `learning_rate`: 3e-5;
- `num_train_epochs`: 5.0;
- `lora_rank`: 32;
- `lora_alpha`: 32.

The following hyperparameters are used for inference:

- `temperature`: 0.35;
- `top_p`: 0.9;
- `top_k`: 10.

B Instruction Data Prompts

B.1 DP Task Data

For DP task data, we show the prompt for each task, using one dataset as an example. Then, we show the prompt for reasoning data, which slightly differs from DP task data. The prompts for inference are the same as tuning, except that dataset-specific knowledge is optional. The prompts for reasoning ground truth collection and head-to-head judge are used for Mixtral.

We use Jellyfish-13B’s prompts as examples. For other models, users may adjust them according to the prompt format of their base models (e.g., using “[INST] [/INST]” blocks for the 7B model).

DP Task Data – ED (Adult)

(system message) *You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can.*

(task description) *Your task is to determine if there is an error in the value of a specific attribute within the whole record provided. The attributes may include age, workclass, education, marital status, occupation, relationship, race, sex, hours per week, country, and income.*

(injected knowledge) *Errors may include, but are not limited to, spelling errors, inconsistencies, or values that don’t make sense given the context of the whole record.*

(instance content) *Record [age: "18-21", workclass: "Private", education: "Some-college", maritalstatus: "Never-married", occupation: "Other-service", relationship: "Own-child", race: "White", sex: "Male", hoursperweek: "30", country: "United-States", income: "eLssThan50K"]*

Attribute for Verification: [income: "eLssThan50K"]

(question) *Is there an error in the value of the "income" attribute?*

(output format) *Choose your answer from: [Yes, No]*

DP Task Data – DI (Restaurant)

(system message) You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can.

(task description) You are presented with a restaurant record that is missing a specific attribute: the city. Your task is to deduce or infer the city of the restaurant using the available information in the record. You may be provided with fields like 'Name', 'Address', 'Phone', and 'Type' to help you in the inference.

(instance content) Record: [name: "darbar", addr: "44 w. 56th st.", phone: "212-432-7227", type: "indian"].

(question) Based on the provided restaurant record, what would you infer is the value for the missing attribute "City"?

(output format) Answer the name of the city.

DP Task Data – SM (MIMIC-III)

(system message) You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can.

(task description) Your task is to determine if the two attributes (columns) are semantically equivalent in the context of merging two tables. Each attribute will be described by its name and a brief description. Your goal is to assess if they refer to the same information based on these names and descriptions provided.

(instance content) Attribute A is [name: "visit_occurrence-visit_end_date", description: "the end date of the visit. if this is a one-day visit the end date should match the start date."].

Attribute B is [name: "admissions-dischtime", description: "dischtime provides the date and time the patient was discharged from the hospital."].

(question) Are Attribute A and Attribute B semantically equivalent?

(output format) Choose your answer from: [Yes, No]

DP Task Data – EM (Beer)

(system message) You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can.

(task description) You are tasked with determining whether two Products listed below are the same based on the information provided. Carefully compare all the attributes before making your decision.

(injected knowledge) Note that missing values (N/A or "nan") should not be used as a basis for your decision.

(instance content) Product A: [name: "Sequoia American Amber Ale", factory: "Wig And Pen"]

Product B: [name: "Aarhus Cains Triple A American Amber Ale", factory: "Aarhus Bryghus"]

(question) Are Product A and Product B the same Product?

(output format) Choose your answer from: [Yes, No]

B.2 Reasoning Data

To construct reasoning data, we use the following prompt.

Reasoning Data

(system message) [same as DP Task Data] While answering, provide detailed explanation and justify your answer.

(task description – question) [same as DP Task Data]

(output format) After your reasoning, finish your response in a separate line with and ONLY with your final answer. Choose your final answer from [Yes, No].

We use the following prompt to collect ground truth from Mixtral (for the Beer dataset in EM).

Reasoning Ground Truth Collection

(system message – output format) [same as Reasoning Data]

(injected knowledge) Note that different factories can belong to the same parent company. The company name of Product B may occur in its product name.

(answer hint) You can use the "Hint" below, but your response cannot contain any information from it.

Hint: the final answer is "No"

In the above prompt, we inject a piece of knowledge specific to the dataset. With this additional knowledge, Mixtral can produce high-quality reasoning result. Note that such knowledge is not prompted to Jellyfish models, as for unseen datasets such knowledge is not always available. In addition to the injected knowledge, Mixtral also receives a hint to the answer of yes or no. As such, we can guarantee that the output reason always points to the correct direction. A sample answer from Mixtral is given below.

To determine if Product A and Product B are the same, we need to compare their names and factories. The names of the products are "Sequoia American Amber Ale" and "Aarhus Cains Triple A American Amber Ale". While both are American Amber Ales, the names are not identical, with Product B having additional words "Aarhus Cains Triple A". Therefore, based on the name, we cannot conclude that the products are the same. Next, let's consider the factories. Product A is produced at "Wig And Pen" while Product B is produced at "Aarhus Bryghus". These are different factories, and the fact that they may belong to the same parent company or use similar production processes does not change the fact that the products are made at different locations. Therefore, based on the comparison of the names and factories, we can conclude that Product A and Product B are not the same product.

Final answer: No

C Injected Knowledge

Table 13: General knowledge.

Prompt
Missing values (N/A or "nan") should not be used as a basis for your decision.
If there are missing values, you should make inferences based only on the information that is available.

Table 14: Task-specific knowledge.

Task	Prompt
ED	Errors may include, but are not limited to, spelling errors, inconsistencies, or values that don't make sense given the context of the whole record. (Used when showing the whole record)
	Errors can include, but are not limited to, spelling errors, inconsistencies, or values that don't make sense for that attribute. (Used when showing only one attribute)
	Capitalization should not be a factor in deciding whether there is an error or not.
DI	Note that values such as 'nan' and 'N/A' mean missing values, and they are not considered as errors. (used when we decide not to treat missing values as errors)
	Note that values such as 'nan' and 'N/A' mean missing values, and they ARE errors. (used when we decide to treat missing values as errors)
EM	To determine if two values are identical, you need to examine both their full names and corresponding acronyms.

Table 15: Dataset-specific knowledge.

Task	Dataset	Prompt
ED	Adult	Both the 'age' attribute and the 'hoursperweek' attribute can represent a range of integer values.
		Verify the consistency of target attribute with related attributes to identify any errors.
	Hospital	The value of attribute "score" can be a percentage number.
DI	Restaurant	The city can often be deduced from the area code of the phone number and the specific street name.
EM	Amazon-Google	Different editions, versions, or operating systems for the same software are all considered as different products.
		You should compare the two product numbers first.
	Beer	Note that different factories can belong to the same parent company.
		Beverages that undergo different production processes, such as the use of various types of wood in the barrelling process, may be considered distinct products.
	Fodors-Zagats	The type of a specific restaurant might vary between different datasets.
	iTunes-Amazon	The length of the same song might vary slightly across different datasets due to rounding or data entry discrepancies.
	DBLP-ACM	The names of authors might be presented in various formats or sequences, even when referring to the same article.
	DBLP-GoogleScholar	The names of authors might be presented in various formats or sequences, even when referring to the same article.

D Few-Shot Prompting

We apply few-shot prompting by manually selecting a subset of data instances from the dataset and labeling them. For instance, a few-shot example for the Beer dataset is presented as follows:

Few-Shot Prompting

(system message – injected knowledge) [same as DP Task Data]
(1st example's instance content) ### Instruction: Product A: [name: "Shirt Tail Amber", factory: "Iron Hill Brewery & Restaurant"]
Product B: [name: "Iron Hill Shirt Tail Amber", factory: "Iron Hill Maple Shade"]
(1st example's question) Are Product A and Product B the same Product?
(1st example's output format) Choose your answer from: [Yes, No]
(1st example's answer) ### Response: Yes
(other examples) ...
(instance content – output format) [same as DP Task Data] ### Response:

The example follows the same format of instance content, question, and output format as in the DP task data. It also provides the answer indicated by *### Response: Yes*. Whereas we only show an positive example here, it is suggested to include both positive and negative examples. After the final example, the instance to be processed is presented in the prompt, and the model follows the same output format as demonstrated in the examples.

Since ground truths are usually not available in real applications, users can handcraft few-shot examples for inference. On the other hand, few-shot examples can be automatically generated by randomly injecting errors for ED and DI, such as missing values, typographical/formatting errors, and randomly swapping values for two columns in a tuple or two tuples in a column. For SM and EM, we can employ rule-based methods (e.g., blocking rules [43]) to quickly find a few matches and use them as few-shot examples.

D.1 Error Detection

The few-shot examples for the Flights and Rayyan datasets are given as follows.

Flights – 1st Example

Instruction:
Record [datasource: "flightview", flight: "AA-3063-SLC-LAX", scheduled departure time: "nan", actual departure time: "8:40 p.m.", scheduled arrival time: "nan", actual arrival time: "9:11 p.m."]
Attribute for Verification: [scheduled departure time: "nan"]
Question: Is there an error in the value of the "scheduled departure time" attribute?
Choose your answer from: [Yes, No]
Response:
Yes

Flights – 2st Example

Instruction:
Record [datasource: "aa", flight: "AA-3823-LAX-DEN", scheduled departure time: "9:00 p.m.", actual departure time: "nan", scheduled arrival time: "12/02/2011 12:15 a.m.", actual arrival time: "nan"]
Attribute for Verification: [scheduled arrival time: "12/02/2011 12:15 a.m."]
Question: Is there an error in the value of the "scheduled arrival time" attribute?
Choose your answer from: [Yes, No]
Response:
Yes

Flights – 3rd Example

Instruction:

Record [datasource: "flightview", flight: "AA-616-DFW-DTW", scheduled departure time: "9:05 a.m.", actual departure time: "10:11 a.m.", scheduled arrival time: "12:35 p.m.", actual arrival time: "1:18 p.m."]

Attribute for Verification: [datasource: "flightview"]

Question: Is there an error in the value of the "datasource" attribute?

Choose your answer from: [Yes, No]

Response:

No

Rayyan – 1st Example

Instruction:

Record [article_title: "A re-appraisal of screening for colour vision impairments", article_language: "nan", journal_title: "Child: Care, Health & Development", journal_abbreviation: "nan", journal_issn: "0305-1862", article_jvolumn: "23", article_jissue: "5", article_jcreated_at: "1/1/97", article_pagination: "391-398", author_list: "D. M. B. Hall", "E. Holroyd"]

Attribute for Verification: [journal_abbreviation: "nan"]

Question: Is there an error in the value of the "journal_abbreviation" attribute?

Choose your answer from: [Yes, No]

Response:

Yes

Rayyan – 2nd Example

Instruction:

Record [article_title: "Nurturing students' interest in primary care research through summer training in meta-analysis.", article_language: "eng", journal_title: "Academic Medicine: Journal Of The Association Of American Medical Colleges", journal_abbreviation: "nan", journal_issn: "1040-2446", article_jvolumn: "76", article_jissue: "5", article_jcreated_at: "5/1/01", article_pagination: "526", author_list: "L N Meurer"]

Attribute for Verification: [article_jissue: "5"]

Question: Is there an error in the value of the "article_jissue" attribute?

Choose your answer from: [Yes, No]

Response:

No

Rayyan – 3rd Example

Instruction:

Record [article_title: "Different renal toxicity profiles in the association of cyclosporine and tacrolimus with sirolimus in rats.", article_language: "eng", journal_title: "Nephrology, dialysis, transplantation : official publication of the European Dialysis and Transplant Association - European Renal Association", journal_abbreviation: "Nephrol. Dial. Transplant.", journal_issn: "1460-2385", article_jvolume: "23", article_jissue: "10", article_jcreated_at: "10/1/08", article_pagination: "3111-9", author_list: ""N\033cria Lloberas", "Marcel la Franquesa", "Josep M Cruzado", "Josep M Griny\ufffd.", "In\0329s Rama", "Gabriela Alperovich", "Immaculada Herrero-Fresneda", "Joan Torras", "Pepita Gim\0329nez-Bonaf\0329""]

Attribute for Verification: [article_pagination: "3111-9"]

Question: Is there an error in the value of the "article_pagination" attribute?

Choose your answer from: [Yes, No]

Response:

Yes

D.2 Data Imputation

The few-shot examples for the Flipkart and Phone datasets are given as follows.

Flipkart – 1st Example

Instruction:

Record: [Product Name: "Himmlisch ST381 Magnetic Sun Shade For Maruti Alto", description: "Himmlisch ST381 Magnetic Sun Shade For Maruti Alto (Side Window) Price: Rs. 1,899 Beat the heat this summer and feel like a VIP with Himmlisch Car Window Magnetic Sunshades. These magnetic sunshades create a mesh layer to stops the heat. Magnet border gets easily stick to your car window door edges (No need of Suction cups) Features: Block UV Rays Keeps Car Cool Easy to install and remove Durable and Exact Fit Provides Complete privacy Resists Heat Mesh Type Sunshade Package Contents: 1 x Set Of 4 Magnetic Sunshades, Specifications of Himmlisch ST381 Magnetic Sun Shade For Maruti Alto (Side Window) General Brand Himmlisch Model Number ST381 Magnetic Placement Position Side Window Color Black Dimensions Weight 4000 g Depth 1.1 cm In the Box Sales Package 4 Sun Shade Pack of 4"]

Based on the provided product record, what would you infer is the value for the missing attribute "brand"?

Answer the name of the brand.

Response:

Himmlisch

Flipkart – 2nd Example

Instruction:

Record: [Product Name: "dilli bazaaar Bellies, Corporate Casuals, Casuals", description: "Key Features of dilli bazaaar Bellies, Corporate Casuals, Casuals Material: Fabric Occasion: Ethnic, Casual, Party, Formal Color: Pink Heel Height: 0, Specifications of dilli bazaaar Bellies, Corporate Casuals, Casuals General Occasion Ethnic, Casual, Party, Formal Ideal For Women Shoe Details Weight 200 g (per single Shoe) - Weight of the product may vary depending on size. Heel Height 0 inch Outer Material Fabric Color Pink"]

Based on the provided product record, what would you infer is the value for the missing attribute "brand"?

Answer the name of the brand.

Response:

dilli bazaaar

Flipkart – 3rd Example

Instruction:

Record: [Product Name: "Shining Diva Alloy Yellow Gold Bangle Set", description: "Shining Diva Alloy Yellow Gold Bangle Set (Pack of 2) Price: Rs. 499 Accentuate Your Feminine Charm Wearing This Beautiful Bangle From The House Of Shining Diva. Made From Premium Quality Material, It Will Retain Its Quality And Lustre For Years To Come. This Bangle Is Lightweight And Skin Friendly. Featuring A Stylish Design And Great Finish, It Will Definitely Give Your Overall Look An Ethereal Dimension. This Bangle Will Surely Catch Your Fancy At Once. It Is Worth Investing In And Will Definitely Get You Noticed. This Bangle Comes In A Set Of Two. Accentuate Your Feminine Charm Wearing This Beautiful Bangle From The House Of Shining Diva. Made From Premium Quality Material, It Will Retain Its Quality And Lustre For Years To Come. This Bangle Is Lightweight And Skin Friendly. Featuring A Stylish Design And Great Finish, It Will Definitely Give Your Overall Look An Ethereal Dimension. This Bangle Will Surely Catch Your Fancy At Once. It Is Worth Investing In And Will Definitely Get You Noticed. This angle Comes In A Set Of Two."]

Based on the provided product record, what would you infer is the value for the missing attribute "brand"?

Answer the name of the brand.

Response:

Shining Diva

Phone – 1st Example

Instruction:

Record: Record: [Product Name: "UNLOCKED RIM BlackBerry Pearl Flip 8220 Smart Cell Phone - Red"]

Based on the provided cellphone record, what would you infer is the value for the missing attribute "brand"?

Answer the name of the brand.

Response:

BlackBerry

Phone – 2nd Example

Instruction:

Record: [Product Name: "OtterBox Apple iPhone 4 & 4S Protective ION Defender Series Case (Retail Packaging) Black"]

Based on the provided cellphone record, what would you infer is the value for the missing attribute "brand"?

Answer the name of the brand.

Response:

OtterBox

Phone – 3rd Example

Instruction:

Record: [Product Name: "DTECH @ 2 PECES! Universal Ring Grip/Stand Holder for any Smart Device, Universal Black Bunker Ring Stand Holder for Apple iPhone 4 4s iphone 5 Samsung Galaxy s3 SIII Samsung GALAXY S6,S6 EDGE.Note II iPad 2 3 ipad mini iPod Nokia LG HTC One X etc,RING Essentials " Cell Phone and Tablets Anti Drop Ring for iPhone 6 plus iPad mini iPad2 iPad iPod Samsung GALAXY NOTE S5 Universal Mobile Devices"]

Based on the provided cellphone record, what would you infer is the value for the missing attribute "brand"?

Answer the name of the brand.

Response:

DTECH

D.3 Schema Matching

The few-shot examples for the CMS dataset are given as follows.

CMS – 1st Example

Instruction:

Attribute A is [name: "condition-occurrence-condition_source_value", description: "the source code for the condition as it appears in the source data. this code is mapped to a standard condition concept in the standardized vocabularies and the original code is stored here for reference."]

Attribute B is [name: "inpatientclaims-admtng_icd9_dgns_cd", description: "claim admitting diagnosis code"]

Are Attribute A and Attribute B semantically equivalent?

Choose your answer from: [Yes, No]

Response:

Yes

CMS – 2nd Example

Instruction:

Attribute A is [name: "provider-npi", description: "the national provider identifier (npi) of the provider."]

Attribute B is [name: "outpatientclaims-op-physn-npi", description: "operating physician – national provider identifier number"]

Are Attribute A and Attribute B semantically equivalent?

Choose your answer from: [Yes, No]

Response:

Yes

CMS – 3rd Example

Instruction:

Attribute A is [name: "visit_detail-visit_detail_start_datetime", description: "the date and time of the visit started."]

Attribute B is [name: "outpatientclaims-desynpuf_id", description: "beneficiary code"]

Are Attribute A and Attribute B semantically equivalent?

Choose your answer from: [Yes, No]

Response:

No

D.4 Entity Matching

The few-shot examples for the Abt-Buy and Walmart-Amazon datasets are given as follows.

Abt-Buy – 1st Example

Instruction:

Product A: [name: "samsung s3 black multimedia player yps3jab", description: "samsung s3 black multimedia player yps3jab 4 gb internal flash memory 1.8 ' tft lcd display touch-sensitive led controls multi-formats support dnse 2.0 sound engine fm tuner and recorder with presets up to 25 hours audio playback up to 4 hours video playback black finish"]

Product B: [name: "samsung 4gb portable mltimdia plyr blk yps-s3jab / xaa", description: "nan"]

Are Product A and Product B the same Product?

Choose your answer from: [Yes, No]

Response:

Yes

Abt-Buy – 2nd Example

Instruction:

Product A: [name: "sony white 8 ' portable dvd player dvpfx820w", description: "sony dvpfx820 white 8 ' portable dvd player dvpfx820w swivel & flip screen with dual sensor for remote control control buttons on screen bezel 12 bit video dac with 108 mhz processing removable , rechargeable battery & car adapter included white finish"]

Product B: [name: "toshiba sd-p71s portable dvd player", description: "toshiba sd-p71s 7 ' portable dvd player"]

Are Product A and Product B the same Product?

Choose your answer from: [Yes, No]

Response:

No

Abt-Buy – 3rd Example

Instruction:

Product A: [name: "sony xplod 10-disc add-on cd/mp3 changer cdx565mxf", description: "sony xplod 10-disc add-on cd/mp3 changer cdx565mxf cd/cd-r/cd-rw and mp3 playback mp3 decoding d-bass 12-second advanced electronic shock protection fm modulator 9 modulation frequencies wireless remote"]

Product B: [name: "sony cdx-565mxf 10-disc cd/mp3 changer", description: "nan"]

Are Product A and Product B the same Product?

Choose your answer from: [Yes, No] ### Response:

Yes

Walmart-Amazon – 1st Example

Instruction:

Product A: [name: "d-link dgs-1005g 5-port gigabit desktop switch", modelno: "dgs1005g"]

Product B: [name: "d-link dgs-1005g 5-port gigabit desktop switch", modelno: "dgs-1005g"]

Are Product A and Product B the same Product?

Choose your answer from: [Yes, No]

Response:

Yes

Walmart-Amazon – 2nd Example

Instruction:

Product A: [name: "nzxt phantom crafted series atx full tower steel chassis black", modelno: "nzxt phantom"]

Product B: [name: "nzxt crafted series atx full tower steel chassis - phantom white", modelno: "phantom white"]

Are Product A and Product B the same Product?

Choose your answer from: [Yes, No]

Response:

No

Walmart-Amazon – 3rd Example

Instruction:

Product A: [name: "at t prepaid gophone samsung a187 with bluetooth blue", modelno: "a187"]

Product B: [name: "samsung a107 prepaid gophone at t", modelno: "a107"]

Are Product A and Product B the same Product?

Choose your answer from: [Yes, No]

Response:

No