DISCOVERGPT: Multi-task Fine-tuning of Large Language Model for Related Table Discovery

Xuming Hu[1][*], **Xiao Qin**[2], **Chuan Lei**[2], **Asterios Katsifodimos**[2], **Zhengyuan Shen**[2], **Balasubramaniam Srinivasan**[2], **Huzefa Rangwala**[2]
[1]The Hong Kong University of Science and Technology (Guangzhou),
[2]Amazon Web Services
xuminghu97@gmail.com {drxqin,chuanlei,akatsifo, donshen,srbalasu,rhuzefa}@amazon.com

Abstract

Natural language understanding over tabular data is crucial for data discovery tasks such as *joinable* and *unionable* table search. State-of-the-art approaches adopt large language models (LLMs) trained over massive text corpora to assess the table semantic relatedness, typically following a pretrain-and-finetune paradigm with labeled tabular data. Recent studies incorporate auxiliary tasks such as entity resolution and column type classification in the fine-tuning phase to improve the performance. However, there is a lack of studies on how different supervisions complement or even contrast each other, leading to a suboptimal performance on the final data discovery tasks. In this paper, we propose a simple yet effective multi-task fine-tuning framework named DISCOVERGPT that holistically discovers and leverages the intricate relationships among the supervisions to optimize the model performance on the data discovery task. Moreover, DISCOVERGPT is plug-and-play that allows a broad range of open-domain auxiliary tasks to be incorporated, by utilizing the generative power of LLMs. We demonstrate the usability and effectiveness of DISCOVERGPT with baseline comparisons and ablation studies. DISCOVERGPT outperforms the top baseline by up to 7% in F1 score[1].

1 Introduction

With the ever-increasing volume of open data scattered on the Internet as well as closed data owned by enterprise organizations, discovering relationships such as *joinable* and *unionable* relation between tables becomes a fundamental task to enable a holistic view of the data for many downstream applications, ranging from data governance, data exploration to data integration (Fan et al., 2023a; Paton et al., 2024; Chapman et al., 2020). As the tables are created, ingested, organized and maintained by different personnel using various data

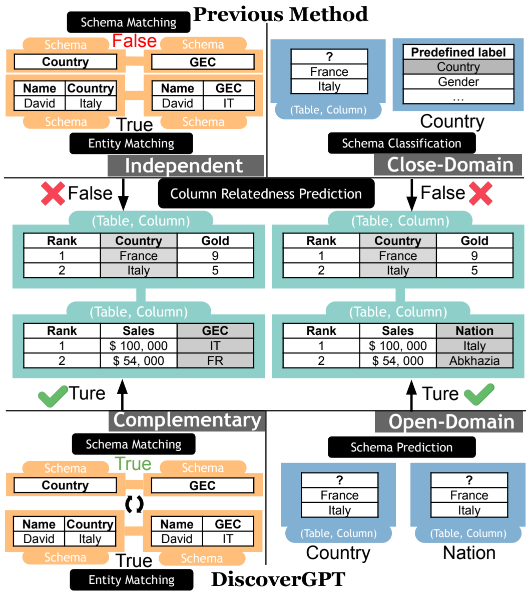


Figure 1: DISCOVERGPT vs. previous methods.

systems, they are becoming inevitably inconsistent and messy. Automatically identifying relationships that are not already defined can be extremely challenging due to both syntactic and semantic discrepancies in the data. LLMs have laid the foundation for the *representation learning-based related table discovery approaches* (Dong et al., 2021; Khatiwada et al., 2023; Hu et al., 2023). These methods typically leverage the LLM as the first layer to convert tabular data into dense vector representations with rich semantic information for the later matching procedure. The LLMs are often further fine-tuned to better understand the nature of tabular data and data discovery task.

State-of-the-art methods and their limitations. The forefront of technology is divided into two main categories: 1) *Table pre-training methods*, which typically utilize table metadata reconstruction (e.g., TAPAS (Herzig et al., 2020), TUTA (Wang et al., 2021)) or content prediction approaches (e.g., TABBIE (Iida et al., 2021)) to enhance LLMs' understanding of table content, and 2) *Table task-assisted training methods*, such as

[*]Work done during an internship at Amazon Web Services.
[1]We will release our code and data upon acceptance.

Starmie (Fan et al., 2023b), Unicorn (Tu et al., 2023), and Table-GPT (Li et al., 2023), introduce auxiliary tasks and their corresponding training objectives, enabling the model to understand tabular data from different perspectives and granularities. Although these methods can improve LLMs' capability to comprehend tabular data and further adapt to specific tasks through fine-tuning, they share common issues. 1) These methods overlook the complementarity between different tasks. For instance, the interactions among tasks on different levels are neglected during the training. As shown in Figure 1, prior methods fail to consider that the "Country" and "GEC" columns could be implicitly matched in the Entity Matching task during the training of the Schema Matching task. This oversight led the model to an incorrect prediction that these two columns as unrelated (i.e., False) in the Column Relatedness Prediction task. 2) The tasks used for pre-training and auxiliary training involve table content classification, requiring predefined labels, which limits the applicability in open-domain and few-shot training tasks. Using Figure 1 as an example, the Schema Classification task is limited to predefined labels, which hinders its ability to generalize from "Country" to "Nation" in open-domain downstream tasks. This limitation leads to erroneous predictions as well.

Our approach. We introduce DiscoverGPT to address the aforementioned issues. 1) By considering the intricate interactions between different levels of tasks such as cells, rows, columns, and tables, DiscoverGPT adaptively assigns weights to different tasks through the exchange of information between them. As shown in Figure 1, DiscoverGPT effectively established the relationship between "Country" and "GEC" by facilitating information exchanged between the Entity Matching and Schema Matching tasks during training. This allows the model to accurately predict their relatedness in the Column Relatedness Prediction task. Here, answers often require reasoning across multiple tables, a process that DiscoverGPT enhances by accurately capturing inter-table relationships. 2) We extend table-assisted training tasks to a broader range of open-domain auxiliary ones, such as Row Population and Column Population, that do not require any labels. As illustrated in Figure 1, the open-domain Schema Prediction task is not confined to predefined labels. For the same column, it can simultaneously predict "Country" and "Nation" classes, thereby accurately predicting the columns as related. Moreover, to construct a dataset for the general-domain data discovery task and enhance the model robustness, we utilize real-world tables and further introduce a series of real-world noise, such as table word abbreviations, misspellings, and omissions, into these tables. We employed manual annotations to label 2,167 entries, indicating the relatedness between columns across different tables.

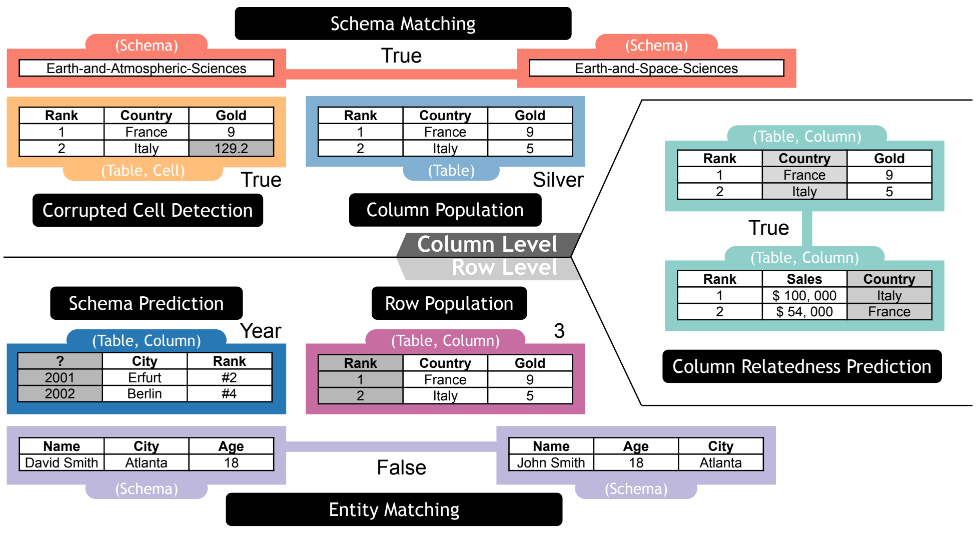
Contributions. Our main contributions are threefold: (1) We proposed a multi-task fine-tuning framework DiscoverGPT that discovers and leverages intricate interactions among sub-tasks for data discovery task. (2) DiscoverGPT is plug-and-play, in which we incorporate a variety of open-domain auxiliary tasks that utilize the generative power of LLMs. (3) We conducted extensive experimental evaluations to demonstrate the effectiveness of DiscoverGPT, which outperforms the best performing baseline by up to 7% in F1 score.

2 Preliminary

Column Relatedness Prediction (CRP) determines whether two columns from two different tables are related based on semantic similarity or underlying relationships. Specifically, we consider the scenario where only a small amount of labeled training data is available. We define a database table as $\mathcal{T}$. For each table $T \in \mathcal{T}$, it is composed of m columns $\{t_1, t_2, \cdots, t_m\}$. Similar to previous studies (Dong et al., 2023), given two database tables T, T', CRP can determine whether any column $t \in T$ can be joined with any column $t' \in T'$. We adopt $S(T, t, T', t')$ to represent the text transformed by the quadruple (T, t, T', t'). The trained language model f_{LM} takes it as input and outputs the CRP result $r(T, t, T', t') = f_{LM}(S(T, t, T', t')) \in \{true, false\}$, where *true* indicates that the two columns can be joined based on semantic similarity or underlying relationships, and *false* indicates they cannot be joined.

3 Multi-task Learning For Column Relatedness Prediction

The ability to predict column relatedness is crucial for effective data discovery, especially when dealing with large datasets. The CRP task requires a substantial amount of labeled data for training accurate prediction models. However, annotating such



Rank	Country	Gold
1	France	9
2	Italy	129.2

Rank	Country	Gold
1	France	9
2	Italy	5

Rank	Country	Gold
1	France	9
2	Italy	5

Rank	Sales	Country
1	$ 100, 000	Italy
2	$ 54, 000	France

?	City	Rank
2001	Erfurt	#2
2002	Berlin	#4

Rank	Country	Gold
1	France	9
2	Italy	5

Name	City	Age
David Smith	Atlanta	18

Name	Age	City
John Smith	18	Atlanta

Figure 2: Illustrations of Column Relatedness Prediction and the six sub-tasks (Corrupted Cell Detection, Schema Matching, Column Population, Row Population, Entity Matching, and Schema Prediction).

data is labor-intensive, requiring domain expertise and extensive resources. Moreover, different domains may have unique characteristics and requirements, necessitating the re-annotation of data for each specific domain. This complexity poses a challenge in building robust CRP models across multiple domains.

3.1 Task Formalization

To overcome the limitations of labeled data availability, we propose utilizing unlabeled data to generate sub-task data for training CRP models. From the perspective of understanding database table row and column information, we propose six sub-tasks, as illustrated in Figure 2. For the CRP and these six sub-tasks, we provide definitions.

(1) Column Relatedness Prediction refers to the task of determining whether two columns from different tables are related or can be joined based on their semantic similarity or underlying relationship. We formalize CRP as the process of deciding whether there exists a meaningful connection between two columns in distinct tables. For example, a CRP task in Figure 2 determines whether the "Country" column in the first table and the "Country" column in the second table are related, indicating a potential join between these columns.

(2) Corrupted Cell Detection aims to detect whether each cell value in a given table is corrupted or not. This task involves identifying any inconsistencies, errors, or anomalies present in the table's cell values. For instance, in Figure 2, the Corrupted Cell Detection task would flag the cell value "129.2" in the "Gold" column as corrupted.

(3) Schema Matching refers to the task of determining whether two schemas point to the same subject or have a matching relationship. Here, we use the term "schema" to denote the names of entity attributes in a database table. Given two schemas as input, the goal is to predict whether they correspond to the same concept or entity. In the example provided in Figure 2, the Schema Matching task would determine that the schemas "Earth-and-Atmospheric-Sciences" and "Earth-and-Space-Sciences" match, indicating a shared subject.

(4) Column Population involves identifying additional columns that can be added to a given table. Given a table as input, the task is to generate new column suggestions that complement the existing columns. In the provided example in Figure 2, the Column Population task would suggest adding a "Silver" column to the table based on the existing "Rank", "Country", and "Gold" columns.

(5) Entity Matching focuses on determining whether two different tuples/rows from separate tables refer to the same real-world object. By comparing the attribute values of the tuples, the task aims to identify if they represent the same entity. In the given example in Figure 2, the Entity Matching task would determine whether the tuple with the name "David Smith," age "18", and city "Atlanta" matches with the tuple having the name "John Smith", age 18, and city "Atlanta".

(6) Schema Prediction involves predicting the masked column name in a table based on the table header, row content, and one of the column names being masked. The task requires understanding the context and semantics of the table to infer the missing column name. In the provided example in Figure 2, the Schema Prediction task would predict

Dataset Name	Train	Validation	Test
Column Population	5,000	1,667	1,667
Row Population	5,000	1,667	1,667
Corrupted Cell Detection	5,000	1,667	1,667
Entity Matching	5,000	1,667	1,667
Schema Matching	5,000	1,667	1,667
Schema Prediction	5,000	1,667	1,667
Column Relatedness Prediction	500	-	1,667

Table 1: Dataset Statistics for 6 Subtasks and Target Task Constructed from NYC Open Data.

"Year" as the masked column name based on the given table information.

(7) Row Population focuses on predicting the next possible value for a specific column in a given table. By analyzing the existing values in the column, the task aims to generate predictions for the subsequent entry. In the provided example in Figure 2, the Row Population task would predict "3" as the next possible value for the "Rank" column based on the existing values in the table.

3.2 Data Generation

We constructed the training, validation, and test datasets for 6 sub-tasks using tables from NYC Open Data in an unsupervised manner. The training set comprises 2,709 distinct tables. The validation and test sets involve 30 tables from various domains, which were cropped and augmented with noise (misspelled table cells, missing words, etc.) yielding 3,150 distinct tables (a half for validation and the other half for testing). Each sub-task's training data consists of 5,000 entries, while the test and validation sets each contain 1,667 entries. The data for this task was generated by specific rules and then corrected by human to ensure data quality. This CRP task consists of 500 training entries and 1,667 test entries. Please refer to Appendix 8.1 for more details. We provide the dataset statistics for our 6 sub-tasks and CRP task in Table 1.

The six sub-tasks proposed in our paper encompass both classification and text generation tasks. We utilize the LLM-base model, which is a versatile text generation model capable of distinguishing different sub-tasks based on input prompt prefixes. Figure 3 shows the various prefixes we use to distinguish different tasks.

The `DiscoverGPT` is pretrained on natural language text rather than structured text. To aid the model in better understanding tabular information, we convert each row in the table into natural language sentences. For example, for the first row in the table corresponding to CRP task in Figure 3, the entities can be represented using the sentence

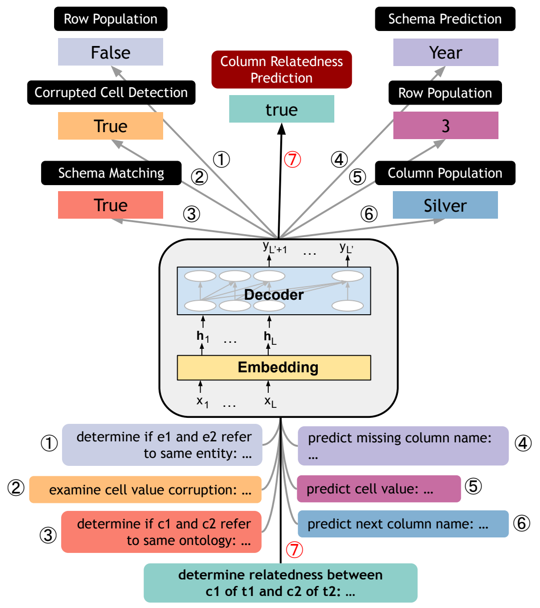

Row Population
False
Corrupted Cell Detection
True
Schema Matching
True
Column Relatedness Prediction
true
Schema Prediction
Year
Row Population
3
Column Population
Silver
①
②
③
④
⑤
⑥
⑦
$y_{L'+1}$ ... $y_{L'}$
Decoder
h_1 ... h_L
Embedding
x_1 ... x_L
① determine if e1 and e2 refer to same entity: ...
② examine cell value corruption: ...
③ determine if c1 and c2 refer to same ontology: ...
④ predict missing column name: ...
⑤ predict cell value: ...
⑥ predict next column name: ...
⑦ determine relatedness between c1 of t1 and c2 of t2: ...

Figure 3: Forms of input and output for the six subtasks and the CRP task in the `DiscoverGPT`.

pattern "The ... is ..." as follows: *(The Rank is 1. The Country is France. The Gold is 9.)*. Similarly, we represent the entire table as the following string: *((The Rank is 1. The Country is France. The Gold is 9.) (The Rank is 2. The Country is Italy. The Gold is 5.))*. For details on the conversion of tables to strings in the 6 sub-tasks and the target task, please refer to Appendix 8.3.

3.3 Preliminary Experiments

We conducted preliminary experiments to evaluate the impact of each sub-task individually and the combined effect of all six sub-tasks on the CRP task. The results in Table 2 demonstrated that utilizing any of the sub-tasks, as well as training on all six sub-tasks simultaneously, led to improved F1 scores, indicating that each sub-task contributes positively to the overall objective.

Sub-Task	Precision	Recall	F1
None	82.42	87.39	84.83
Corrupted Cell Detection	86.84	91.53	89.12
Schema Matching	86.93	90.28	88.57
Column Population	87.25	91.81	89.47
Row Population	87.04	89.41	88.21
Entity Matching	86.82	91.38	89.04
Schema Prediction	86.19	89.90	88.01
All	88.24	94.56	**91.29**

Table 2: Performance on the CRP task after adopting different subtasks (based on the LLaMA2-7B model).

Note that there is a collaborative interaction among different tasks, allowing them to complement and enhance each other's performance. Additionally, given the varying significance of each sub-task to the target task, CRP, treating all tasks as equally important fails to facilitate their mutual enhancement and may also lead to potential overfitting. Specifically, sub-tasks that present excessive difficulty are assigned lesser importance as overtraining on these could lead to overfitting, ultimately detracting from the overall objective.

3.4 Self-Adaptive Weighted Multi-task Learning

To address this issue, we can manually set weights that reflect the importance of the six sub-tasks:

$$\mathcal{L}_{manual}(W) = \sum_{i=1}^{c} w_i \mathcal{L}_{task_i}(W), \tag{1}$$

where $c = 6$ is the number of sub-tasks, $\mathcal{L}_{task_i}(W)$ is the loss function for training the DISCOVERGPT model on the i-th sub-task, which is the cross-entropy loss for classification tasks.

However, manually adjusting the six weight parameters would require a significant amount of time. If we treat the six weights as trainable parameters, the values of w_i ($i \in \{1, 2, \cdots, 6\}$) would gradually decrease to 0, resulting in loss function $\mathcal{L}_{manual}(W) = 0$ and preventing proper training.

To enable different tasks to complement each other and reflect the importance of each task by automatically adjusting the weights of six tasks through training, we adopted a training method in which the weights can converge to non-zero values. First, we introduce a concept similar to "temperature" to modify the classification task results. A higher "temperature" leads to more evenly distributed classification results, with lower relevance to the model's output and lower training intensity. We use $\sigma > 0$ to represent the "temperature", and the modified probability of the true label being c given the input x and the model weights W can be expressed using the following Softmax function:

$$p(y = c|x, W) = \text{Softmax}\left(\frac{1}{\sigma^2} f^W(x)\right)_c \in [0, 1]^k, \tag{2}$$

where k is the number of labels, $f^W(x) \in \mathbb{R}^k$ is the model output on the input x, and $\text{Softmax}(1/\sigma^2 \cdot f^W(x))_c \in [0, 1]$ is the c-th element in the vector $\text{Softmax}(1/\sigma^2 \cdot f^W(x)) \in [0, 1]^k$.

After performing Maximum Likelihood Estimation (MLE) on $p(y|x, W)$, we obtain the corresponding loss function:

$$\begin{aligned}
\mathcal{L}_{one}(W) =& -\log p(y = c|x, W) \\
=& -\log \text{Softmax}\left(\frac{1}{\sigma^2} f^W(x)\right)_c \\
=& \frac{1}{\sigma^2} \mathcal{L}_{one}^{CE}(W) \\
& - \frac{1}{\sigma^2} \log \sum_{i=1}^{k} \exp\left(f^W(x)_i\right) \\
& + \log \sum_{i=1}^{k} \exp\left(\frac{1}{\sigma^2} \cdot f^W(x)_i\right),
\end{aligned} \tag{3}$$

where $\mathcal{L}_{one}^{CE}(W) = -\log \text{Softmax}(1/\sigma^2 \cdot f^W(x))_c$ is the cross-entropy loss function for classification purposes, and $f^W(x)_i \in \mathbb{R}$ is the i-th element of the vector $f^W(x) \in \mathbb{R}^k$. However, this loss function is computationally expensive. To reduce this cost, we assume that our sub-tasks are reasonably set in terms of difficulty. In this scenario, σ will not be heavily updated and will remain close to 1. When $\sigma \approx 1$, the following equation approximately holds true:

$$\begin{aligned}
& \sum_{i=1}^{k} \exp\left(\frac{1}{\sigma^2} \cdot f^W(x)_i\right) \\
\approx & \sigma \cdot \left(\sum_{i=1}^{k} \exp\left(f^W(x)_i\right)\right)^{\frac{1}{\sigma^2}}.
\end{aligned} \tag{4}$$

Incorporating Eq. 4 into Eq. 3 yields an approximation of the loss function:

$$\mathcal{L}_{one}^{app}(W) = \frac{1}{\sigma^2} \mathcal{L}_{one}^{CE}(W) + \log \sigma, \tag{5}$$

which is a concave function with respect to σ. This ensures that both σ and $1/\sigma^2$ can converge to non-zero values.

Similarly, for text generation tasks of length L, its loss function is the average of L individual classification task cross-entropy losses. When $\sigma \approx 1$, loss of the predicting the sequence of length L with "temperature", $\mathcal{L}_{seq}(W)$, satisfies:

$$\begin{aligned}
\mathcal{L}_{seq}(W) \approx& \frac{1}{L} \sum_{j=1}^{L} \mathcal{L}_{one_j}^{app}(W) \\
=& \frac{1}{\sigma^2} \mathcal{L}_{seq}^{CE}(W) + \log \sigma \\
=& \mathcal{L}_{seq}^{app}(W),
\end{aligned} \tag{6}$$

where $\mathcal{L}_{one_j}^{app}(W)$ is the cross-entropy loss function for predicting the j-th token, $\mathcal{L}_{seq}^{CE}(W)$ is the average cross-entropy loss for predicting all L tokens, and $\mathcal{L}_{seq}^{app}(W)$ is the approximation of the sequence prediction loss when $\sigma \approx 1$.

Model	Base Model	Sub-task	Algorithm	Accuracy	Precision	Recall	F1
Unicorn (Zero-shot)	RoBERTa	×	×	50.00	50.00	100.00	66.67
Starmie (Finetuned)	RoBERTa	Unicorn	×	80.56	81.27	79.61	80.43
AutoTUS (Finetuned)	RoBERTa	Unicorn	×	80.98	80.90	81.71	81.30
Unicorn (Finetuned)	RoBERTa	Unicorn	×	81.80	84.53	77.94	81.82
TABBIE	LLaMa2-7B	Corrupt Cell Detection	×	86.14	86.68	88.33	87.50
Table-GPT	LLaMa2-7B	Table-GPT	×	87.66	87.25	93.62	90.32
DiscoverGPT	T5-base	Ours	×	85.66	87.13	85.39	85.44
	T5-base	Ours	SAW	92.74	92.96	92.65	92.72
	Falcon-7b	Unicorn	×	85.24	92.35	75.96	83.36
	Falcon-7b	Unicorn	SAW	91.06	94.37	86.81	90.43
	Falcon-7b	Ours	×	87.82	80.83	98.27	88.70
	Falcon-7b	Ours	SAW	93.94	93.71	93.83	93.77
	LLaMa2-7B	Ours	×	88.38	88.24	94.56	91.29
	LLaMa2-7B	Ours	AutoSTL	90.12	90.45	90.88	90.66
	LLaMa2-7B	Ours	AdaTask	91.53	91.70	91.62	91.66
	LLaMa2-7B	Ours	SAW	94.53	94.17	95.05	94.61
	LLaMa2-13B	Ours	×	89.82	86.39	97.15	91.45
	LLaMa2-13B	Ours	SAW	**95.01**	94.82	95.72	**95.17**

Table 3: Performance of the CRP task. "Sub-task" represents the "Sub-task Training Dataset". "Algorithm" indicates the specific adaptive learning method used for each model. "SAW' stands for the use of Self-Adaptive Weights. In the "Sub-task" column, "Unicorn" refers to the training dataset used in the Tu et al. (2023), comprising 7 data matching tasks, "Table-GPT" represents the dataset we reproduced according to the description in the Li et al. (2023). "Ours" represents the dataset used in this study, consisting of 6 types of sub-tasks.

During the training of the LLM model, the loss function for multiple sequence prediction is equal to the average cross-entropy loss for each token in each sequence. Therefore, following the same derivation process as above, we can obtain the approximation of the loss function for a single sub-task when $\sigma \approx 1$:

$$\mathcal{L}_{task}^{app}(W) = \frac{1}{\sigma^2}\mathcal{L}_{task}^{CE}(W) + \log \sigma, \tag{7}$$

where $\mathcal{L}_{task}^{CE}(W)$ is the average cross-entropy loss for predicting all sequences in this task.

4 Experimental Evaluation

Dataset. In our comparison of DiscoverGPT (T5-base) with DiscoverGPT (Falcon-7b), we utilized the dataset from six sub-tasks and the target task CRP, as detailed in Section 3.2, for training, while the target task CRP was also used for testing. Moreover, we ensured a nearly 1:1 ratio of positive to negative examples in each classification task. For the comparison between Unicorn and DiscoverGPT (Falcon-7B), the Unicorn model was trained using the complete training dataset of seven data matching tasks. To draw a parallel with DiscoverGPT, the DiscoverGPT (Falcon-7B) was also pretrained on these seven data matching tasks and then fine-tuned on the target task CRP.

Metric. In the evaluation process, we primarily utilize metrics including F1 Score, Accuracy, Precision, and Recall. It's important to note that in the task of binary classification ("*true*" and "*false*"), Micro F1 is functionally identical to Macro F1.

Baselines. To validate the effectiveness of the sub-task and Self-Adaptive Weight method we proposed, we compared our method with Unicorn (Tu et al., 2023), TABBIE (Iida et al., 2021), Starmie (Fan et al., 2023b), AutoTUS (Hu et al., 2023) and Table-GPT (Li et al., 2023), as well as self-adaptive multi-task learning baselines such as AutoSTL (Zhang et al., 2023b) and AdaTask (Yang et al., 2023). It should be noted that other baseline models utilize an encoder-decoder network structure designed specifically for single-table structures, which makes them incomparable directly with DiscoverGPT.

4.1 Main Result

In Table 3, we compared the performance of Unicorn (Zero-shot), Unicorn (Finetune on target task CRP) and DiscoverGPT (Falcon-7b trained on Unicorn dataset and finetuned on target task CRP). Additionally, it contrasts the predictive performances of DiscoverGPT (T5-base) and DiscoverGPT (LLMs like Falcon-7B, LLaMa2-7B and LLaMa2-13B), examining the impact of using LLMs with and without self-adaptive weights. We derive the following insights from the result:

• When only a limited portion of the Unicorn training data is utilized, the DiscoverGPT models based on T5-base and Falcon-7b demonstrate superior performance compared to Unicorn. This indicates enhanced transfer learning capabilities within our framework.

• Self-adaptive weights (SAW) are highly effective compared to other adaptive learning techniques. Across T5-base, Falcon-7b, LLaMa2-7B, and LLaMa2-13B models, SAW consistently leads to an F1 score improvement ranging from 5% to 7% over models without adaptive weights. When comparing SAW with AutoSTL and AdaTask, the SAW-enhanced models exhibit an F1 score advantage of approximately 3% to 4%, as demonstrated by LLaMa2-7B achieving an F1 score of 94.61 with SAW, compared to 91.66 with AdaTask. These results highlight SAW's ability to more effectively adapt to task difficulties, significantly reducing overfitting on challenging tasks and further boosting performance on the target task CRP.

• Falcon-7B, LLaMa2-7B, and LLaMa2-13B each demonstrate their efficiency by requiring fewer subtask data than T5-base to attain comparable F1 scores on the target task. This efficiency reflects the enhanced prior knowledge and superior generalization abilities inherent in LLMs.

• The adaptive weight method, while effective, yields diminishing returns in performance enhancement as the model size increases. The improvement observed in T5-base is more pronounced than in LLMs. We believe this is because the tabular knowledge in the subtask data we input may have already been encompassed during the pre-training phase of the LLMs.

4.2 Analysis of Sub-task Complementarity

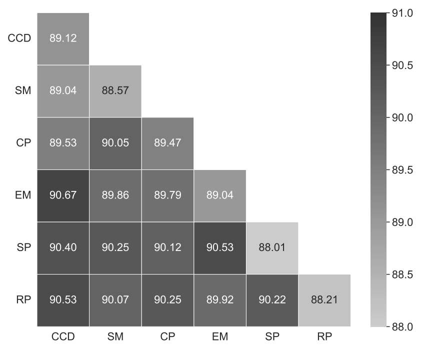


Figure 4: Complementarity matrix of 6 sub-tasks, each sub-task is represented by its corresponding acronym (based on the LLaMA2-7B model).

To assess how the six sub-tasks complement each other, we conducted experiments where we trained using only one or two tasks at a time. We then documented the F1 scores on the target task CRP, as depicted in the task complementarity matrix in Figure 4. In this matrix, the F1 scores on the diagonal represent outcomes from pre-training on a single sub-task followed by training and testing on the column relatedness prediction task. The F1 scores in other positions reflect the results of jointly pre-training on two sub-tasks and then proceeding to train and test on the column relatedness prediction task. Note that in these experiments, we did not implement self-adaptive weight adjustments, meaning that σ remained fixed at 1.

The three sub-tasks (CCD, SM, CP) designed to enhance the model's comprehension of columns are grouped into set C, while the sub-tasks focused on row understanding (EM, SP, RP) are categorized as set R. As depicted in Figure 4, the training results using tasks from both C and R show higher average and median F1 scores, 90.22 and 90.25 respectively, compared to the results from other settings which have an average of 89.31 and a median of 89.30. This observation led us to experimentally prove that tasks focusing on column understanding and those concentrating on row understanding complement each other, enhancing overall model performance.

4.3 Analysis of DiscoverGPT's Latency and Scalability

Token Length	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)	Latency (ms)
256	90.28	86.85	92.47	91.14	178
512	92.44	92.59	93.05	92.82	387
1024	94.13	94.42	95.12	94.77	824

Table 4: Performance and Latency of DiscoverGPT across Different Token Lengths.

To evaluate the scalability of DiscoverGPT w.r.t. the latency, we conducted experiments by varying token lengths, representing different row and column configurations. The results demonstrate DiscoverGPT's ability to handle increasing data sizes with reasonable response times.

Table 4 shows that as token length increases, both accuracy and F1 scores improve consistently, indicating that DiscoverGPT effectively utilizes the additional information from larger tables. For example, with 256 tokens, the model achieves an accuracy of 90.28% and an average latency of 178 ms, while at 1024 tokens, accuracy rises to 94.13% with a latency of 824 ms. This scaling pattern suggests that performance gains are balanced with acceptable latency increases, demonstrating the model's robustness and efficient handling of larger datasets. DiscoverGPT thus maintains an effective trade-off between accuracy and response time, making it suitable for applications with different data sizes.

SAW	Weights						F1
	EM	CCD	SM	SP	RP	CP	
×	**0.9673**	**0.9956**	**0.9795**	0.9369	0.9109	0.9239	95.07
✓	**0.9563**	**0.9377**	**0.9567**	0.9316	0.9282	0.9354	95.38

Table 5: The weights of the 6 sub-tasks when F1 is maximized in the sub-task weight space searching.

4.4 Analysis of the Dependency of Target Tasks on Sub-tasks

We have proven in Section 3.3 that sub-task training has a positive effect on the target task. In order to quantify the dependence of the target task on different sub-tasks, we use Gaussian Process Minimization (GPM) to maximize F1 on the target task CRP to search the sub-task weight space, and analyze the sub-task weights when F1 is maximized. GPM is particularly effective for optimizing hyperparameters in scenarios where each evaluation of the function is costly. It requires an objective function f, a function g to minimize f, and a set of variable space X, upon which f depends. In our context, we define X as the weights of the six sub-tasks. The function f is set as the negative of the F1 score on the target task test set, following pre-training and target task training. To expedite the search, our exploration is limited to the range of $[0.9, 1]^6$. We give the detailed algorithm process of GPM in Appendix 8.2.

To manage training overhead, we focused our experiments on LLaMa2-7B-base. The results after sampling 30 times are shown in Table 5. The first row of Table 5 shows that the weights assigned to the classification sub-tasks are higher than that for the text generation sub-tasks. This implies that the target task CRP is more dependent on the classification sub-tasks. This finding aligns with the results from the self-adaptive weight adjustment using LLaMa2-7B-base DISCOVERGPT, as seen in the second row of Table 5. Furthermore, this consistency underscores the effectiveness of self-adaptive weight adjustment in the training process.

5 Related Works

Related Table Discovery. Related table discovery methods focus on identifying relationships such as joinble and unionable between the tables in a data repository. Early works (Zhu et al., 2019, 2016) assumed that the relationships can be identified by examining the exact string match and focused on designing indexing and data compression methods to support efficient discovery. In a data lake scenario where there exists inconsistency in data representation, many works (Dong et al., 2023, 2021; Fernandez et al., 2018; Khatiwada et al., 2023; Hu et al., 2023; Chen et al., 2023b) explored the idea of using dense representation to resolve the syntactical mismatch in a semantic embedding space. Zhu et al. (2017); Li et al. (2024b); Lou et al. (2024) found the relationships with consideration of the data transformation lineage. Most of the learning-based solutions focused on optimizing the effectiveness by focusing on one discovery task.

Data tasks with LLM. Recent studies showed that LLMs, as general problem solvers , demonstrated their strength in addressing problems such as entity resolution, schema matching, error detection in tabular data for data cleaning, data imputation for repairing dirty data sources (Narayan et al., 2022; Zhang et al., 2023a; Chen et al., 2023a) due to the ability to process natural language and having the prompt interface that allows developer to effectively define the task for them. However, they are limited by the high compute cost and the domain knowledge absent from the LLM training process (Arefeen et al., 2024). The existing works mostly focused on prompt engineering by designing efficient prompting strategy to save cost and providing and selecting domain knowledge for effective domain adaption. Fan et al. (2023c); Li et al. (2024a) studied efficient prompting strategy with demonstrating example selection for entity resolution. Aycock and Bawden (2024) employed topic models to select in-context examples, enhancing translation quality across multiple domains and language pairs. Sheetrit et al. (2024) proposed a LLM-based solution for schema matching with schema pruning and searching algorithm to reduce the computation.

6 Conclusion

We present DISCOVERGPT, a multi-task fine-tuning framework that tackles related table discovery problem by discovering the relationships between different supervisions to maximize the performance gain. DISCOVERGPT takes a holistic approach to leverage a wide range of supervisions together in one training job and seamlessly discovers and optimizes the relationships among these supervisions. It is a plug-and-play framework with the ability to incorporate a wide range of auxiliary tasks by leveraging the generative power of LLMs. Experimental evaluations with up to 7% improvement in F1 score demonstrate the superiority of DISCOVERGPT, comparing against baselines.

7 Limitation

The limitations of this study involves two different aspects: technical constraints and application-based constraints: 1) On the technical front, we primarily focused on experimenting the proposed methods and framework for data discovery tasks as the main tasks. The effectiveness of the technical proposals for arbitrary main tasks on relational tables was not studied. In this regard, the generalizability of DISCOVERGPT to other main tasks is unclear. 2) In terms of applications, the data employed in our experiments are sourced from NYC Open Data. Beyond the open data, certain domain-specific data, such as those in finance, healthcare and energy, are yet to be incorporated. Additionally, we have exclusively focused on tabular data in English, thus expanding our method to support multi-lingual tabular data is a potential avenue for future research.

References

Md Adnan Arefeen, Biplob Debnath, and Srimat Chakradhar. 2024. Leancontext: Cost-efficient domain-specific question answering using llms. *Natural Language Processing Journal*, page 100065.

Seth Aycock and Rachel Bawden. 2024. Topic-guided example selection for domain adaptation in llm-based machine translation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 175–195.

Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *VLDB J.*, 29(1):251–272.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2023a. An empirical survey of data augmentation for limited data learning in nlp. *Transactions of the Association for Computational Linguistics*, 11:191–211.

Leiyuan Chen, Chengsong Huang, Xiaoqing Zheng, Jinshu Lin, and Xuan-Jing Huang. 2023b. Tablevlm: Multi-modal pre-training for table structure recognition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2437–2449.

Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. 2021. Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 456–467. IEEE.

Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. Deepjoin: Joinable table discovery with pre-trained language models. 16(10):2458–2470.

Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023a. Table discovery in data lakes: State-of-the-art and future directions. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, pages 69–75. ACM.

Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023b. Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning. *Proc. VLDB Endow.*, 16(7):1726–1739.

Meihao Fan, Xiaoyue Han, Ju Fan, Chengliang Chai, Nan Tang, Guoliang Li, and Xiaoyong Du. 2023c. Cost-effective in-context learning for entity resolution: A design space exploration. *CoRR*, abs/2312.03987.

Raul Castro Fernandez, Essam Mansour, Abdulhakim A Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Seeping semantics: Linking datasets using word embeddings for data discovery. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 989–1000. IEEE.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xuming Hu, Shen Wang, Xiao Qin, Chuan Lei, Zhengyuan Shen, Christos Faloutsos, Asterios Katsifodimos, George Karypis, Lijie Wen, and S Yu Philip. 2023. Automatic table union search with tabular representation learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3786–3800.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 3446–3456. Association for Computational Linguistics.

Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J Miller, and Mirek Riedewald. 2023. Santos: Relationship-based semantic table union search. *Proceedings of the ACM on Management of Data*, 1(1):1–25.

Huahang Li, Longyu Feng, Shuangyin Li, Fei Hao, Chen Jason Zhang, Yuanfeng Song, and Lei Chen. 2024a. On leveraging large language models for enhancing entity resolution. *CoRR*, abs/2401.03426.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*.

Shujie Li, Liang Li, Ruiying Geng, Min Yang, Binhua Li, Guanghu Yuan, Wanwei He, Shao Yuan, Can Ma, Fei Huang, et al. 2024b. Unifying structured data as graph for data-to-text pre-training. *Transactions of the Association for Computational Linguistics*, 12:210–228.

Yuze Lou, Chuan Lei, Xiao Qin, Zichen Wang, Christos Faloutsos, Rishita Anubhai, and Huzefa Rangwala. 2024. Datalore: Can a large language model find all lost scrolls in a data repository? In *ICDE 2024*.

Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *Proc. VLDB Endow.*, 16(4):738–746.

Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. 2024. Dataset discovery and exploration: A survey. *ACM Comput. Surv.*, 56(4):102:1–102:37.

Eitam Sheetrit, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Rematch: Retrieval enhanced schema matching with llms. *arXiv preprint arXiv:2403.01567*.

Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Guoliang Li, Xiaoyong Du, Xiaofeng Jia, and Song Gao. 2023. Unicorn: A unified multi-tasking model for supporting matching tasks in data integration. *Proc. ACM Manag. Data*, 1(1):84:1–84:26.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: tree-based transformers for generally structured table pre-training. In *Proc. of SIGKDD*, pages 1780–1790.

Enneng Yang, Junwei Pan, Ximei Wang, Haibin Yu, Li Shen, Xihua Chen, Lei Xiao, Jie Jiang, and Guibing Guo. 2023. Adatask: A task-aware adaptive learning rate approach to multi-task learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 10745–10753.

Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023a. Large language models as data preprocessors. *CoRR*, abs/2308.16361.

Zijian Zhang, Xiangyu Zhao, Hao Miao, Chunxu Zhang, Hongwei Zhao, and Junbo Zhang. 2023b. Autostl: Automated spatio-temporal multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4902–4910.

Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 847–864. ACM.

Erkang Zhu, Yeye He, and Surajit Chaudhuri. 2017. Auto-join: Joining tables by leveraging transformations. *Proceedings of the VLDB Endowment*, 10(10):1034–1045.

Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH ensemble: Internet-scale domain search. *Proc. VLDB Endow.*, 9(12):1185–1196.

8 Appendix

8.1 Data Generation for Column Relatedness Prediction

To prevent data leakage, we use different tables to generate the training and testing dataset. We use 3150 tables derived from 20 foundational tables to generate the training dataset, and 450 tables derived from another set of 10 foundational tables to generate the testing dataset. These foundational tables were sourced from NYC Open Data.

The process of generating tables from foundational tables involves the following steps.

(1) Each foundational table was horizontally divided into three tables with the same number of rows. For each table. 10%, 20%, $\cdots$, 100% rows are selected to generate tables with varying numbers of rows.

(2) We then use the four following settings to inject noise into cell values and column names of the generated tables: 1) injecting noise into both cell values and column names, 2) only injecting noise into cell values, 3) only injecting noise into column names, and 4) no noise injection at all. The order of rows was randomly shuffled.

(3) For each table, we generate 4 positive samples and 4 negative samples to ensure class balance in the generated data. We take the generation of training data from 3150 tables as an example.

① Firstly, for each of the 3150 tables, we generate 2 positive samples. Any column of a table can be used to generate a positive sample with itself. We randomly select two columns from the same table to generate these two samples. For example, from a table randomly selected from the 3150 tables, Table 1, one of its columns (denoted as Column A) can be paired with itself to create positive samples: (Table 1, Column A) and (Table 1, Column A).

② For two tables originating from the same foundational table, we can generate two positive and two negative samples. We randomly select two columns from the foundational table to generate two positive samples. For instance, if Table 1 and Table 2 come from the same foundational table, and Column A in Table 1 and Column B in Table 2 both originate from the same column of the foundational table, then (Table 1, Column A) and (Table 2, Column B) make up a positive sample.

③ If Column A in Table 1 and Column B in Table 2 do not originate from the same column in the foundational table, then (Table 1, Column A) and (Table 2, Column B) form a negative sample. We randomly select two sets of columns, A and B, that do not come from the same column to create two negative samples.

④ Finally, we randomly select two tables originating from different foundational tables, denoted as Table 1 and Table 2. Any column from Table 1 paired with any column from Table 2 will constitute a negative sample. We construct two negative samples in this way.

(4) For each sample, we randomly shuffle the order of the columns and remove 0% to 50% of the columns to further increase the noise. We randomly select 5,000 entries to keep for training data and 1,667 for testing data, maintaining a 1:1 ratio of positive to negative samples.

For all generated positive column pairs, we additionally employed a manual annotation process to avoid potential noisy data. Specifically, we hired five annotators, each holding a bachelor's degree and possessing fluent English skills and data analysis capabilities. These annotators were tasked with determining whether two columns were indeed related; unrelated column pairs were discarded. Each data point was labeled by three annotators simultaneously, and labels were retained based on the majority rule principle. Instances of inconsistent labeling occurred in approximately 17% of cases. All annotators received compensation based on the data they labeled.

8.2 Detailed algorithm process of Gaussian Process Minimization

Algorithm 1 Gaussian Process Minimization

```
InitialPoints ← InitialPointsGenerator($n, X$)
$x_i$ = Initial Points, $y_i = \{f(x)$ for $x$ in $x_i\}$
optimizer ← Optimizer(Initial Points)
**for** $i \leftarrow 0, 1, \cdots, n-1$ **do**
    $x \leftarrow$ Optimizer
    $y \leftarrow f(x)$
    $x_i \leftarrow x_i \cup \{x\}$,$y_i \leftarrow y_i \cup \{y\}$
**end for**
$x^* = \arg\min_{x \in x_i} f(x)$
$y^* = \min_{x \in y_i} f(x)$
**return** $x^*, y^*$
```

The Initial Points Generator generates the initial set of n points in the variable space X, and the optimizer minimizes $f(x)$ using Gaussian process.

8.3 Example of Table Conversion to Strings

We present the inputs and labels for 6 sub-tasks and 1 target task to demonstrate how this approach transforms tables into natural language:

Task Name	Input	Label
Column Relatedness Prediction	determine relatedness between c1 of t1 and c2 of t2: c1:B or c2:borough t1((The LOTY is Store/Commercial. The INCIZ is 10018. The INADDR is 63 WEST 38 STREET. The STN is WEST 38 STREET. The CRSTR1 is 5 AVENUE. The CRST2 is AVENUE OF THE AMERICAS. The INTST1 is 5 AVENUE. The INTEST2 is AVENUE OF THE AMERICAS. The ADTY is ADDRESS. The Ci is NEW YORK. The La is WEST 38 STREET. The FACT is nan. The Sta is In Progress. The DD is nan. The RESOLDESCR is nan. The RESOACTUPDD is 2023-04-08T00:47:12.000. The COMMUBOA is 05 MANHATTAN. The B is 1008400006.0. The Bor is MANHATTAN. The XCOORSTPL is 988509.0. The YCOORSTAPL is 213168.0. The OPDCHAT is MOBILE.) t2((The created_date is 2023-04-07T22:R8:30.000. The closed_date is 2023-04-07T12:57:30.000. The agency is NYPD. The agency_name is New York Ci47 Police Departnent. The complaint_type is Iplegal Parking. The descriptor is Blocked Hydranr. The location_type is Syreet/Sidewalk. The incident_zip is 10472. The incident_address is 1328 COMMONWEALTH AVENIE. The street_name is COMMONW2ALTB AVENUE. The cross_street_1 is EAST 272 STREET. The cross_street_2 is EAST 164 STREET. The intersection_street_1 is EAST 17Q STREET. The intersection_street_2 is EAST 174 SRREET. The address_type is ADDRESS. The city is BRONX. The landmark is CONMONWSALTH AVENUE. The facility_type is nan. The status is Closed. The due_)	true

Corrupted Cell Detection	examine cell value corruption: t1((The district is 1. The grade is 3. The year is 2013. The category is Econ Disadv. The number_tested is 615. The mean_scale_score is 292. The level1_n is 278. The level1_ is 45.2. The level2_n is 207. The level2_ is 33.7.) (The district is 1. The grade is 3. The year is 2014. The category is Econ Disadv. The number_tested is 594. The mean_scale_score is 286. The level1_n is 306. The level1_ is 51.5. The level2_n is 176. The level2_ is 29.6.) (The district is 1. The grade is 3. The year is 2015. The category is Econ Disadv. The number_tested is 467. The mean_scale_score is 289. The level1_n is 215. The level1_ is 46.0. The level2_n is 144. The level2_ is 30.8.) (The district is 1. The grade is 3. The year is Econ Disadv. The category is Econ Disadv. The number_tested is 459. The mean_scale_score is <pos> 305. The level1_n is 142. The level1_ is 30.9. The level2_n is 155. The level2_ is 33.8.) (The district is 1. The grade is 3. The year is 2017. The category is Econ Disadv. The number_tested is 456. The mean_scale_score is 303. The level1_n is 144. The level1_ is 31.6. The level2_n is 144. The level2_ is 31.6.) (The district is 1. The grade is 4. The year is 2013. The category is Econ Disadv. The number_tested is 602. The mean_scale_score is 294. The level1_n is 215. The level1_ is 35.7. The level2_n is 254. The level2_ is 42.2.) (The district is 1. The grade is 4. The year is 2014. The category is Econ Disadv. The number_tested is 587. The mean_scale_)	false
Schema Matching	determine if c1 and c2 refer to same schema: c1(filing), c2(filing)	true

Column Population	predict next column name: t1((The objectid is 646. The a is 748. The service_category is Vaccines. The service_type is Flu Vaccine (Influenza). The walk_in is Yes. The insurance is Yes. The children is No. The facility_name is Newtown Pharmacy. The address is 28-04 31st Street.) (The objectid is 70. The a is 224. The service_category is Vaccines. The service_type is Flu Vaccine (Influenza). The walk_in is Yes. The insurance is Yes. The children is No. The facility_name is Walgreens Drug Store. The address is 84-20 Broadway.) (The objectid is 810. The a is 840. The service_category is Vaccines. The service_type is Flu Vaccine (Influenza). The walk_in is Yes. The insurance is Yes. The children is Yes. The facility_name is Homecrest Clinic. The address is 1601 AVENUE S.) (The objectid is 630. The a is 659. The service_category is Vaccines. The service_type is Flu Vaccine (Influenza). The walk_in is Yes. The insurance is Yes. The children is No. The facility_name is Medcare Health Inc.. The address is 260 Kings Highway.) (The objectid is 366. The a is 182. The service_category is Vaccines. The service_type is Flu Vaccine (Influenza). The walk_in is Yes. The insurance is Yes. The children is No. The facility_name is Duane Reade. The address is 949 3RD AVE.) (The objectid is 139. The a is 127. The service_category is Vaccines. The service_type is Flu Vaccine (Influenza). The walk_in is Yes. The insurance is Yes. The children is No. The facility_name is Duane Reade. The address is 10309 LIBERTY AVE.) (The objectid is 761. The a	city
Entity Matching	determine if e1 and e2 refer to same entity: e1(The dbn is 01M184. The location_name is P.S. 184m Shuang Wen. The location_category is K-8. The administrative_district is 1. The removals is 0. The principal is 0. The superintendent is 0. The expulsions is 0. The sy1718_total_removals is 0.) e2(The female_1 is 201. The grade_10 is 195. The grade_9 is 31. The grade_12 is 102. The grade_11 is 149. The year is 2015-16. The female_2 is 0.4213836477987421 6. The category is English Language Learners. The total_enrollment is 477. The district is 1.)	false
Schema Prediction	predict missing column name: t1((The permitno is 101. The expiration_date is 2004-03-31. The effectivedate is 2004-02-23. The	companyname

Row Population	predict cell value: t1(R. The hispanic_students_with_2 is R. The white_students_with_2_or is R. The multi_racial_students_with is R.) (The dbn is 01M020. The location_name is P.S. 020 Anna Silver. The location_category is Elementary. The administrative_district is 1. The american_indian_alaskan_native is R. The asian_students_with_2_or is R. The black_students_with_2_or is R. The hispanic_students_with_2 is R. The white_students_with_2_or is R. The multi_racial_students_with is R.) (The dbn is 01M034. The location_name is P.S. 034 Franklin D. Roosevelt. The location_category is K-8. The administrative_district is 1. The american_indian_alaskan_native is R. The asian_students_with_2_or is R. The black_students_with_2_or is R. The hispanic_students_with_2 is 6. The white_students_with_2_or is R. The multi_racial_students_with is R.) (The dbn is 01M063. The location_name is The STAR Academy - P.S.63. The location_category is Elementary. The administrative_district is 1. The american_indian_alaskan_native is R. The asian_students_with_2_or is R. The black_students_with_2_or is R. The hispanic_students_with_2 is R. The white_students_with_2_or is R. The multi_racial_students_with is R.) (The american_indian_alaskan_native is <blank>.)	R