Language Models are Explorers for Join Discovery on Data Lakes

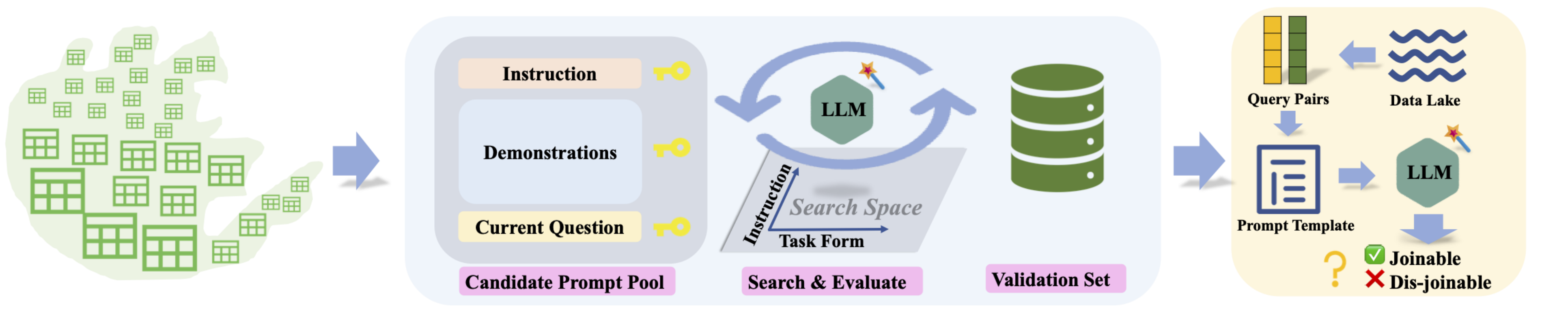
VLDB 2024 Submission

Yaohua Wang[1], Bolin Ding[1], Haibin Wang[1], Rong Zhu[1], Zhijian Ma[1], Biqing Huang[2], Jingren Zhou[1]
[1]DAMO Academy, Alibaba Group [2]Tsinghua University

Paper Code

Abstract

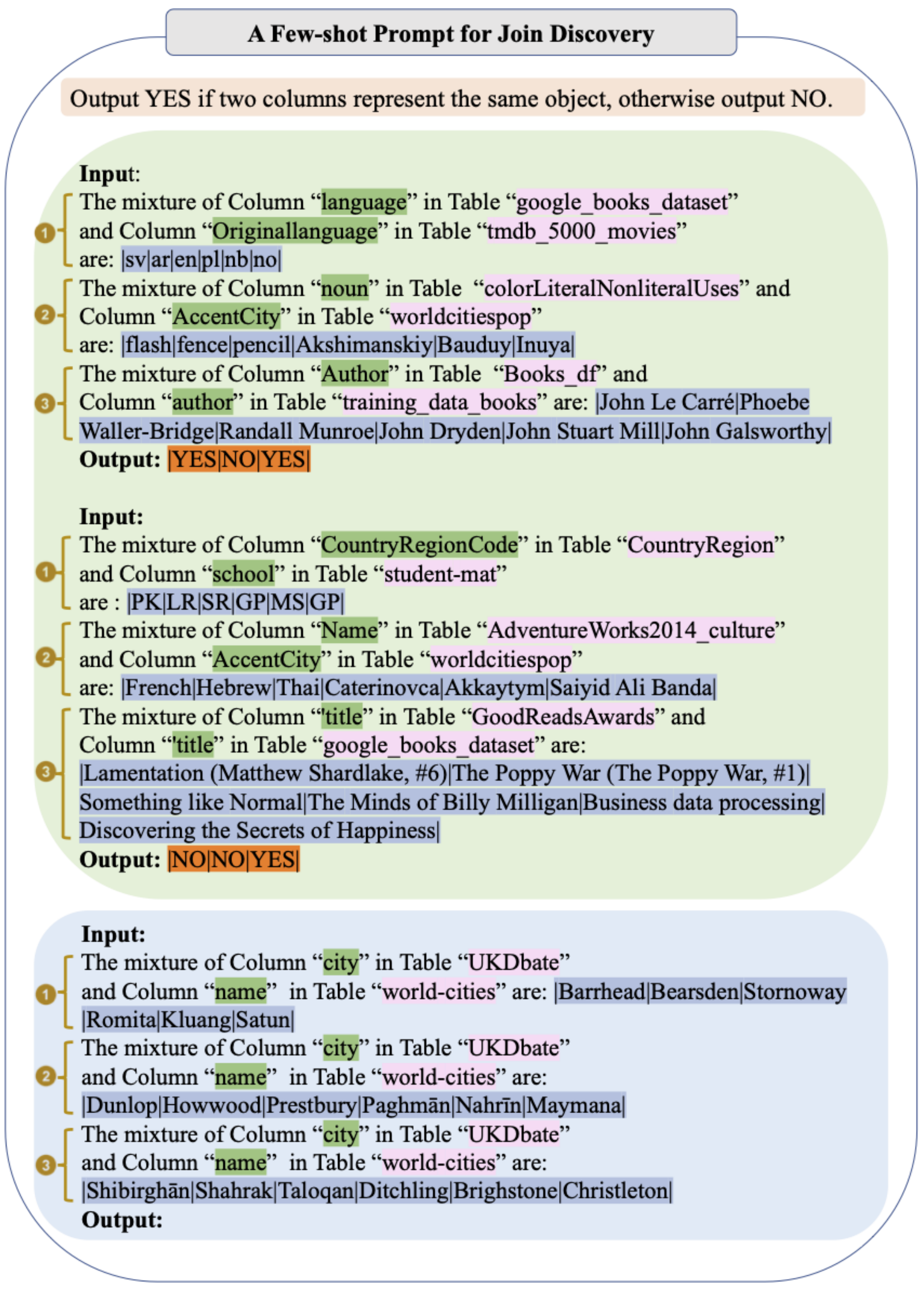
TL;DR: **JD-SCOPE** is a novel system for join discovery on data lakes with LLMs.



The data lake is a repository of independent data tables. **PromptingJD performs searches and evaluations to find the suitable prompt template for join discovery.** **The optimal prompt template found can be used on data lakes.**

Join discovery is a typical task in data lake research. It finds joinable relationships between columns of different tables which is critical for data integration in schema-less data lakes. On the other hand, LLMs have achieved promising results in many natural language tasks recently. They behave well with the in-context learning ability that only needs a few examples and no fine-tuning, showing great advantages that can be applied to join discovery. However, directly applying the prompt generation methods in NLP will cause a migration problem. It means that strategies useful in NLP scenarios cannot be used in data lakes or obtain limited efficacy, leading to sub-optimal prompts. This is because of the differences in the task definition, data modalities and the availability of labeled samples between the two scenarios. The unsuitable prompts not only fail to inspire the superior ability of LLMs but even mislead LLMs to output incorrect results, leading to performance degradation. Therefore, a novel Join Discovery System with Comprehensive and Optimized Prompt Engineering (JD-SCOPE) is proposed to deal with join discovery on data lakes. It first constructs unsupervised examples for demonstrations of the prompt and the validation set for hyper-parameters tuning. Then it explores the prompt template automatically. Meanwhile, JD-SCOPE also ensures stable outputs of LLMs, thus avoiding the impact of randomness. In addition, JD-SCOPE can be extended to semantic join discovery to explore column pairs that are conceptually correlated but not identical (e.g., *country code* and *language code*) for a comprehension understanding of data. JD-SCOPE paves a way to ground data lakes with LLMs and harnesses the knowledge and logical reasoning power of LLMs for join discovery. To evaluate join discovery effectively, the JD-Lake dataset is curated and benchmarked with baselines and a series of popular LLMs. The experiments prove that JD-SCOPE alleviates this migration problem well and empirically outperforms alternatives, showing the superiority of JD-SCOPE.

What the final prompt template looks like



A Few-shot Prompt for Join Discovery

Output YES if two columns represent the same object, otherwise output NO.

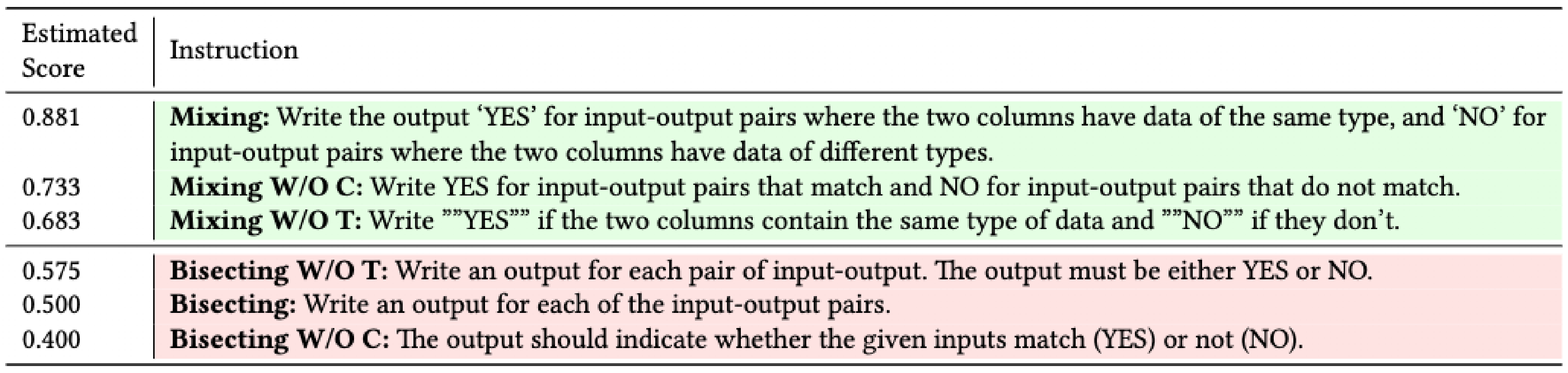
Input:
① The mixture of Column “language” in Table “google_books_dataset” and Column “Originallanguage” in Table “tmdb_5000_movies” are: |sv|ar|en|pl|nb|no|
② The mixture of Column “noun” in Table “colorLiteralNonliteralUses” and Column “AccentCity” in Table “worldcitiespop” are: |flash|fence|pencil|Akshimanskiy|Bauduy|Inuya|
③ The mixture of Column “Author” in Table “Books_df” and Column “author” in Table “training_data_books” are: |John Le Carré|Phoebe Waller-Bridge|Randall Munroe|John Dryden|John Stuart Mill|John Galsworthy|
Output: |YES|NO|YES|

Input:
① The mixture of Column “CountryRegionCode” in Table “CountryRegion” and Column “school” in Table “student-mat” are : |PK|LR|SR|GP|MS|GP|
② The mixture of Column “Name” in Table “AdventureWorks2014_culture” and Column “AccentCity” in Table “worldcitiespop” are: |French|Hebrew|Thai|Caterinovca|Akkaytym|Saiyid Ali Banda|
③ The mixture of Column “title” in Table “GoodReadsAwards” and Column “title” in Table “google_books_dataset” are: |Lamentation (Matthew Shardlake, #6)|The Poppy War (The Poppy War, #1)| Something like Normal|The Minds of Billy Milligan|Business data processing| Discovering the Secrets of Happiness|
Output: |NO|NO|YES|

Input:
① The mixture of Column “city” in Table “UKDbate” and Column “name” in Table “world-cities” are: |Barrhead|Bearsden|Stornoway |Romita|Kluang|Satun|
② The mixture of Column “city” in Table “UKDbate” and Column “name” in Table “world-cities” are: |Dunlop|Howwood|Prestbury|Paghmān|Nahrīn|Maymana|
③ The mixture of Column “city” in Table “UKDbate” and Column “name” in Table “world-cities” are: |Shibirghān|Shahrak|Taloqan|Ditchling|Brighstone|Christleton|
Output:

An illustration of a two-shot single prompt multiple discriminations style prompt in Mixing Form with Md=3. First, all the current questions and demonstrations are put into a single prompt to reduce complexity. It can reduce the number of prompt builds and LLM calls to 1. In consequence, it need not collect and manage the outputs of multiple calls of LLMs, because the output of this prompt is a sequence of discriminations. Second, a single prompt means a single inference time in theory. Also, it only needs 1 instruction and saves Md − 1 instruction tokens compared with the basic version. The token number saved is proportional to the dataset size. The larger the dataset size is, the more tokens are saved in this way. The form is called Single Prompt Multiple Discriminations Style (S-M Style), which increases the scalability of JD-SCOPE. Last, together with more demonstrations, the multiple current questions of the candidate column pairs provide a richer context in the prompt. The LLMs can see more comprehensive information and then output more stable results for each question which is insensitive to thresholds. Therefore, the prompt in Fig. 3 (III) in the paper evolves into the improved version form in the Figure above.

The instructions found and their estimated scores by JD-SCOPE



Estimated Score	Instruction
0.881	**Mixing:** Write the output ‘YES’ for input-output pairs where the two columns have data of the same type, and ‘NO’ for input-output pairs where the two columns have data of different types.
0.733	**Mixing W/O C:** Write YES for input-output pairs that match and NO for input-output pairs that do not match.
0.683	**Mixing W/O T:** Write ””YES”” if the two columns contain the same type of data and ””NO”” if they don’t.
0.575	**Bisecting W/O T:** Write an output for each pair of input-output. The output must be either YES or NO.
0.500	**Bisecting:** Write an output for each of the input-output pairs.
0.400	**Bisecting W/O C:** The output should indicate whether the given inputs match (YES) or not (NO).

The retrieved instructions with high and low estimated scores in the search and evaluation phase are picked and shown in the table above. It can be seen that the instructions with high estimated scores (green background) share the advantage of clear, precise and detailed statements, while the low score instructions (pink background) are ambiguous and rough. The SoTA JD-SCOPE uses the instruction of 0.881 in the *Mixing Form*, proving this task form is the most consistent with the thinking logic of LLMs for this task and dataset.

Generated examples of column name and table caption by JD-SCOPE

Original	Generated	Values Based on
IOC	country code	FLK, ERI, CZE, PUR, ARU
JobRole	job category	Manager, Sales Executive, Human Resources
category	book genre	Humor, Food & Cooking, Fiction, Fantasy
Postcode	number	512-0921, M4E 1G3, 66450
bank	set of letters	YZ, KL, WX, CD, QR
genes_genes	gene_information	GeneID, Essential, Class, Complex, Motif, Function
world_city	cities	Name, CountryCode, District
worldcitiespop	world_cities	Country, City, AccentCity
imdb_ijs_actors	person	firstname, lastname, gender
financial_order	transaction records	orderid, accountid, bankto, accountto, amount

The *Metadate Generation* is qualitatively analyzed in the table above. The generated column names are shown in the first block. The original columns “IOC", “JobRole", and “category" can be nicely summarized as “country code", “job category" and “book gere" respectively according to the cell values shown in column *Values Based on*. However, columns “Postcode" and “bank" are summarized as “number" and “set of letters", which lost the key information, revealing the deficiencies in the knowledge ability of LLMs. The generated table captions are in the second block. Similarly, some tables (e.g., “genes_genes", “world_city" and “worldcitiespop") can also be well summarized, and others (e.g., “imdb_ijs_actors", “financial_order") are poorly summarized and not informative enough. The generated metadata improves the join discovery by adding extra information. This ability comes from the fact that LLMs have seen massive amounts of corpus during the training phase, and this capability has been skillfully applied to the join discovery task.

The Algorithmic pseudo-code of JD-SCOPE

Algorithm 1 The algorithm of candidate instruction composition

```
Input: constructed demonstration set $\mathcal{D}$, a predefined prompt template $\mathcal{P}$, task pool $\mathcal{T}$, the LLM $\mathcal{F}$, number of demonstrations $N_e$ and number of rounds to generate instructions $N_g$.
Output: instruction list $\mathcal{I}$
1: $\mathcal{I} \leftarrow []$
2: for each $t$ in $\mathcal{T}$ do
3:     initialize $S \leftarrow \emptyset$
4:     initialize $C \leftarrow N_g$
5:     while $C > 0$ do
6:         sample a subset with $N_e$ demonstrations $\mathcal{D}' \subseteq \mathcal{D}$
7:         ensemble an instance prompt $p \leftarrow t, \mathcal{D}', \mathcal{P}$
8:         compose an instruction $i \leftarrow \mathcal{F}(p)$
9:         $S \leftarrow S \cup \{i\}$
10:        $C \leftarrow C - 1$
11:    end while
12:    $\mathcal{I} \leftarrow \mathcal{I} + [S]$
13: end for
```

Algorithm 2 The algorithm of search and evaluation

```
Input: constructed validation set $\mathcal{D}_v$, constructed demonstration set $\mathcal{D}_d$, search space $\mathcal{E}$, the LLM $\mathcal{F}$, number of samples to evaluate for each unit per round $N_s$, number of units to evaluate per round $N_u$, and number of rounds to evaluate $N_r$.
Output: the score table for all units $S$.
1: for each $i \in \mathcal{E}$ do                ▷ initialize $S(i), N_t(i), N_c(i)$
2:     $S(i) \leftarrow 0$
3:     $N_t(i) \leftarrow 0$
4:     $N_c(i) \leftarrow 0$
5: end for
6: initialize $C \leftarrow 0$
7: while $C < N_r$ do
8:     get $N_u$ units $\mathcal{E}' \subseteq \mathcal{E}$ with the highest scores acc. to $S$
9:     for each $i \in \mathcal{E}_u$ do
10:        sample $N_s$ examples subset $\mathcal{D}_v' \subseteq \mathcal{D}_v$
11:        for each $v \in \mathcal{D}_v'$ do
12:            Sample demonstrations $\mathcal{D}_d' \subseteq \mathcal{D}_d$
13:            ensemble an instance prompt $p \leftarrow v, i, \mathcal{D}_d'$
14:            if $\mathcal{F}(p)$ matches label of $v$ then
15:                $N_c(i) \leftarrow N_c(i) + 1$
16:            end if
17:        end for
18:        $N_t(i) \leftarrow N_t(i) + N_s$
19:        update historical accuracy $x(i) = \frac{N_c(i)}{N_t(i)}$
20:        update the $S(i)$ according to Eq in Sec.4.2.3
21:    end for
22:    $C \leftarrow C + 1$
23: end while
```

The algorithm of *Candidate Instruction Composition* of Section 4.2.3 to generate candidate instruction is described in detail in Algorithm 1. The algorithm of *Search and Evaluation* of Section 4.2.3 to find the optimal instruction and task form is shown in Algorithm 2.