Experimental Evaluation of JupySim

Misato Horiuchi* Yuya Sasaki* Chuan Xiao* Makoto Onizuka*

1 EXPERIMENTAL EVALUATION

We show the experimental evaluation to show accuracy, efficiency, and scalability of our methods. We assume that computational notebook similarity search is generally run on client computers. Thus, in our experiment, we use a Mac notebook with 2.30GHz Intel Core i5 and 16GB memory. We implement and execute all algorithms by Python. We use PostgreSQL and Neo4J to store tabular data and workflow graphs, respectively.

1.1 Experimental Setting We describe dataset, comparison method, and queries that we used in our experimental study.

Dataset. We use 111 computational notebooks shared on Kaggle as the dataset. They are used for various purposes, including machine learning, data analysis, and data visualization.

We convert them into workflow graphs as execution order of cells is from top to bottom, since most of the computational notebooks on Kaggle are usually supposed to run cells in that order. In addition, we obtain the contents of the tabular data by executing all the cells in the order and saving into the database. The time for converting computational notebook into workflow graph was as long as the running time of computational notebooks for tabular data collection because the graph construction time is shorter enough to be ignored than the data collection time. Tables 1–3 show statistics of workflow graphs, source codes, and tabular data, respectively.

Measures of content similarity. We describe the measures of content similarity in our experiments.

(1) Source code. We separate the strings of source codes in the cells into a set of words. The delimiters are space, new line, period, and equal. We define the similarity between two sets of source codes as the Jaccard similarity coefficient for the sets of words.

(2) Tabular data. Given two tabular data D_A and D_B, we separate them into columns, we define each of col_A and col_B as a set of unique values per columns.

Table 1: Statistics of workflow graphs

Elements	Max	Min	Average	Total
# of nodes with $\ell_{\mathcal{S}}$	123	5	29.92	3321
# of nodes with $\ell_{\mathcal{D}}$	44	0	8.86	983
# of nodes with $\ell_{\mathcal{O}}$	107	0	26.59	2952
# of edges	282	12	73.10	8114
Max in-degrees	8	1	2.71	-
Max out-degrees	21	2	4.97	-
# of libraries	22	2	6.86	762
# of DataFrame	43	1	4.28	475
# of png	46	1	8.79	976
# of text	47	1	13.52	1501

Table 2: Statistics for lines of source codes

Max	Min	Per cell	Per notebook	Total
9043	1	232.52	48261.63	772186

Table 3: Statistics for tabular data

Max [MB]	Min [Byte]	Average [MB]	Total [MB]
248.7	144	85.9	9538.9

We define the number of columns in D_A is $|D_A|$, and it is fewer than the number of rows in D_B. We define $Jacc(col_A, col_B)$ as the similarity between col_A and col_B which is calculated as Jaccard similarity coefficient. Then we select $|D_A|$ pairs in descending order to $Jacc(col_A, col_B)$. Injection $T_{col} : col_A \to col_B$ explains correspondence of the pairs, we calculate the similarity of the table by $\frac{1}{s}\sum_{col_A \in D_A} Jacc(col_A, T_{col}(col_A))$.

(3) Library. We use the Jaccard similarity coefficient to sets of words of library names used in the computational notebooks.

(4) Output. We use similarity of outputs as whether two outputs match in the type {DataFrame, text, png}. The similarity is one if they match, and zero if they do not match.

Comparison methods. We use six methods for evaluation: Graph-based, Graph-based w/o opt, Set-based, Set-based w/o opt, C, and Juneau. Graph-based and Graph-based w/o opt are the graph-based similarity methods with and without all optimization techniques, respectively. Set-based is the set-based similarity method with optimization techniques.Other

*Graduate school of Information Science and Technology, Osaka University. {horiuchi.misato, sasaki, chuanx, onizuka}@ist.osaka-u.ac.jp

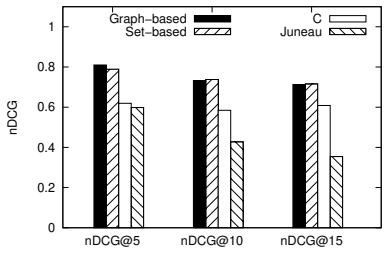

Figure 1: Search accuracy

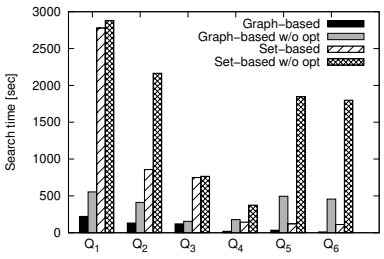

Figure 2: Top-10 search time

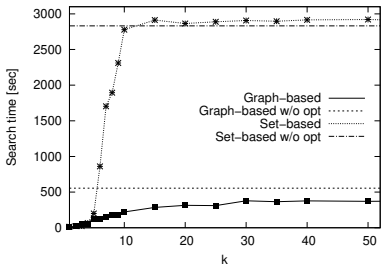

Figure 3: Search time on rank k

optimization techniques cannot apply to the set-based similarity. Set-based w/o opt does not use optimization techniques and can be considered as a baseline that independently uses similarity measures for each content. C uses only code similarity. Juneau is a data search method [2], which approximately finds similar tabular data.

Queries and parameters. We use six queries whose query graphs have different attributes for each node. Q_1–Q_3, and Q_4–Q_6 are the same topology, respectively. Each of them is a fragment of a computational notebook in the dataset.

In set-based similarity, we set ($\alpha_{\mathcal{S}}$, $\alpha_{\mathcal{D}}$, $\alpha_{\mathcal{L}}$, $\alpha_{\mathcal{O}}$) as (32, 2, 1, 1) for Q_1–Q_3, and (32, 1, 1, 1) for Q_4–Q_6. In graph-based similarity, we set ($\beta_{\mathcal{S}}$, $\beta_{\mathcal{D}}$, $\beta_{\mathcal{L}}$, $\beta_{\mathcal{O}}$) as (8, 1, 1, 1) for all queries. In both similarity, ℓ_θ is $\ell_{\mathcal{D}}$ for all queries.

1.2 Results and Analysis We show the experimental results to validate the accuracy, efficiency, and scalability of our method.

1.2.1 Accuracy We evaluate the accuracy of search results based on user experiments. In the user experiments, nine users manually scored similarity of computational notebooks to Q_4–Q_6. The score range is from 1 to 5, and we use the average of each score of the manual evaluations for the queries.

Figure 1 shows nDCG [1] of Graph-based, Set-based, C, and Juneau. Graph-based and Set-based achieve high nDCG. Since Graph-based is more accurate than Set-based, we can confirm that it is effective to preserve the computation order of cells and relationship of contents. From these results, we can see that our problem definitions are useful for the similarity search.

1.2.2 Efficiency We evaluate the efficiency of our methods and optimization techniques.

Figure 2 shows the search time for the Top-10 search. The search time of Graph-based is much smaller than that of Set-based. This indicates that subgraph matching is effective to reduce the computation costs without sacrificing the accuracy. Optimization techniques can accelerate the search method for both Graph-based and Set-based.

Figure 3 shows the search time with varying k. For all k, Graph-based is the most efficient among all the methods. In particular, when k is small, the effectiveness of optimization techniques is large. This is because pruning works well due to high k-th similarity. The search time of methods without optimization techniques is constant because they compute all computational notebooks without pruning.

As the ablation study, we evaluate which optimization techniques are effective in Graph-based. We remove/add each optimization technique; pruning (PR for short), Order optimization (OR for short) , caching (CA for short) , and indexing (IND for short).

Table 4 shows the search time of our graph-based similarity methods with/without the techniques for Query 1. From Table 4, we can see that the search time increases when PR is not applied. Therefore, PR is the most effective technique to reduce the computation time. The addition of OPT and CA makes it faster, while IND is less effective in reducing the computation time. This is because the computation time for subgraph matching is smaller than content similarity computation, and thus, effectiveness of IND is smaller than the other techniques.

1.2.3 Scalability We finally evaluate the scalability of our graph-based similarity methods with optimization. We use Q_1–Q_3.For the scalability test, we generate copies of computational notebooks with random additions and deletions of nodes and libraries from the real computational notebooks. We prepare two environments; Original and Without the most similar. Original generates copies from all computational notebooks, while Without the most similar excludes the most similar computational notebooks of Q_1–Q_3. This causes that Original can be easily pruned a large amount of computational notebooks than Without the most simi-

Table 4: Ablation study

Method	Search time [sec]
PR + OR + CA + IND	208.66
OR + CA + IND	506.62
PR + CA + IND	333.47
PR + OR + IND	251.28
PR + OR + CA	209.77
CA + IND	508.96
Graph-based w/o opt	554.81

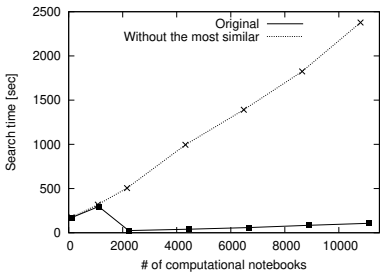

Figure 4: Search time on dataset size

lar, because k-th similarity becomes much high.

Figure 4 shows the search time varying with the number of computational notebooks. Our method increases the search time less than linearly. Even if the number of computational notebooks becomes 100 times, the search time becomes about 20 times. If there are many similar computational notebooks in databases, the computational costs become very small. We here note that costs of workflow graph construction equals to executing computational notebooks. These results show that our search method and workflow graph construction are scalable.

References

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the ICML*, pages 89–96, 2005.

[2] Y. Zhang and Z. G. Ives. Finding related tables in data lakes for interactive data science. In *Proceedings of the ACM SIGMOD*, pages 1951–1966, 2020.