4L-07

計算ノートブック類似検索技術の開発

堀内 美聡 † 山崎 翔平 ‡ 佐々木 勇和 § 肖 川 ¶ 鬼塚 真 ‖
大阪大学 † 大阪大学 ‡ 大阪大学 § 大阪大学 ¶ 大阪大学 ‖

1 はじめに

計算ノートブックは対話型のプログラミングインタフェースであり，代表例として Jupyter Notebook が挙げられる．機械学習やデータ分析を主な用途として多くのユーザに利用され，GitHub や Kaggle に公開されている計算ノートブックの数は膨大である [1]．

これらの既存の計算ノートブックを再利用するため，高精度かつ高速な計算ノートブックの類似検索技術の需要が高まっている．ここで，内容とファイル名が一致していない計算ノートブックが多数存在するため，キーワード検索は現実的でない [2]．加えて，計算ノートブックのソースコードはセル単位で区切られており，表形式データや使用ライブラリ，セルの出力と関係し合っているため，単にこれらの要素の類似度のみに基づいた評価で適切に類似検索を行うことは困難である．そのため，ソースコードに対する類似検索技術 [3, 4] や表形式データのみを対象とした検索技術 [5] により類似した計算ノートブックを検索することは難しい．

これらのことから，計算ノートブックの要素間の関係性やセルの実行順序を考慮した検索技術が必要であるが，セルの実行順や関係性を考慮した計算ノートブック検索は困難であり，既存研究も存在しない．

本研究では，計算ノートブックの各要素の類似度計算に加えて，要素間の関係性とセルの実行順序を捉えることが可能な新たな類似検索手法を提案する．この手法では，計算ノートブックをワークフロー化しサブグラフマッチングを利用することによって，データ処理や表形式データ読み込みなどの順序を考慮した類似検索を可能とする．さらに，インデックスの構築と要素の類似度計算順の最適化することで，類似検索を高速化する．実験では実データを用いて検索時間の評価を行い，提案手法の高速性を示す．

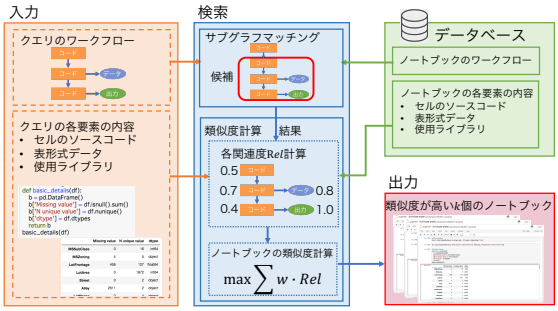

図 1 類似検索フレームワーク

2 類似検索フレームワーク

2.1 概観

類似検索では，ソースコード，表形式データ，ライブラリ，セルの出力の形式を入力とし，類似性が上位 k 個の計算ノートブックを出力する．類似性が高い計算ノートブックは，ソースコード，表形式データ，ライブラリがそれぞれ類似しており，セルの出力の形式が多く一致しているものとする．さらに，ソースコードの実行や出力などの順序も類似している．

フレームワークの全体像を図 1 に示す．提案フレームワークでは，クエリおよび検索対象の計算ノートブックのワークフローを DAG で表し，それぞれクエリグラフ Q，ワークフローグラフ W とする．これらのノードは，セルのソースコード，表形式データ，出力であり，有向辺で実行順序や関係を表す．ライブラリはグラフ属性である．

類似ノートブックの素朴な検索手法は，ワークフローグラフに対しクエリグラフのサブグラフマッチングを行い，対応するノード間の類似性を関連度として計算する．関連度の重み和を取ることによって，その計算ノートブックの類似度を計算する．

Developing a similarity search algorithm for computational notebooks
† Misato Horiuchi, Osaka University
‡ Shohei Yamasaki, Osaka University
§ Yuya Sasaki, Osaka University
¶ Chuan Xiao, Osaka University
‖ Makoto Onizuka, Osaka University

2.2 検索の高速化

素朴な手法では，検索結果を求めるために，全てのワークフローグラフに対して，全ての要素の関連度計算を行う．ワークフローグラフの一部の関連度しか計算していなくても，上位 k 件に入らないことが明らかである場合に，計算の枝刈りをする．次に，計算コストが特に大きい関連度計算以外の関連度を先に計算し，その関連度計算の前に枝刈りが実行できるように計算順序を最適化する．さらに，サブグラフマッチングの結果に，グラフの一部が同一のものが複数存在する場合に，関連度計算結果をキャッシュしておくことにより重複した計算を削減する．最後に，W は Q よりノード数が少ない，あるいは辺の最大入出次数が小さい場合，明らかに W と Q のサブグラフマッチング の結果は 0 個である．そのため，ワークフローグラフの構造情報をインデックス化し，サブグラフマッチングの枝刈りを行う．

3 実験

データセットには， Kaggle で共有されている 112 件の計算ノートブックを使用する．セルの実行順は上から順であるとしてワークフローグラフへの変換を行なった．

比較手法: 比較手法としては，ワークフローグラフを利用する検索手法として，2.1 節で示した素朴な手法 (naive)，素朴な手法に対して 2.2 節で示した高速化を適用した提案手法 (proposal) とする．また，ワークフローグラフを利用しない検索手法として，サブグラフマッチングを行わず単に関連度の和による類似度計算によって検索する手法 (not workflow-based) およびその手法に 2.2 節と同等の高速化を適用したもの (not workflow-based (fast)) とする．これは，Q のノードそれぞれと関連度が高い W のノードの組み合わせに基づき類似度計算をする検索方法である．加えて，ソースコードの関連度は，セル単位の比較ではなく，計算ノートブックのすべてのセルを結合した文字列に対して関連度計算を行う．

検索時間: 入力には，ワークフローグラフの形は同一であるが，各ノードの変換元が異なるクエリを 4 つ使用する．これらはデータセット内の 2 つの計算ノートブックの一部を切り出してワークフローグラフに変換したもの (query1, 3)，およびそれらの一部のノードの内容をを入れ替えたもの (query2, 4) である．これらのクエリで Top-10 検索を実行した検索時間の結果は図 2 となった．“proposal” は “naive” より検索時間を短縮している．加えて，いずれのクエリにおいても，ワークフローグラフを利用する “naive” は，利用しない “not workflow-based” より計算時間が短縮している．これは，ワークフローグラフを利用する場合はサブグラフマッチングの結果に含まれるノードしか関連度の計算対象でないので，計算コストの大きい関連度計算の回数が削減されたためである．

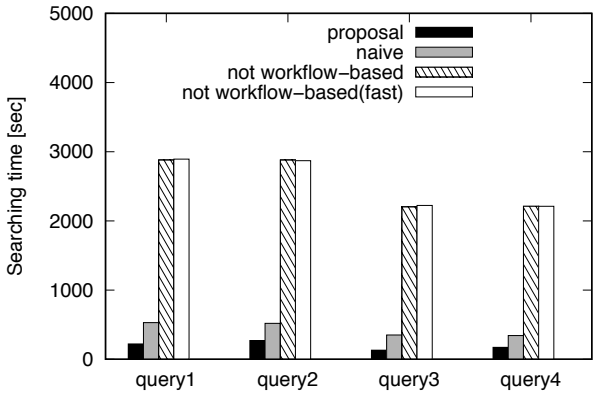

図 2　各クエリでの Top-10 検索の検索時間

謝辞 本研究は JSPS 科研費 JP20H00583 の助成を受けたものです。

参考文献

[1] Cong Yan and Yeye He. Auto-suggest: Learning-to-recommend data preparation steps using data science notebooks. In *Proceedings of the ACM SIGMOD*, pp. 1539–1554, 2020.

[2] Souti Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. What's wrong with computational notebooks? pain points, needs, and design opportunities. In *Proceedings of the ACM CHI*, pp. 1–12, 2020.

[3] Jens Krinke. Identifying similar code with program dependence graphs. In *Proceedings of the WCRE*, pp. 301–309, 2001.

[4] Rainer Koschke, Raimar Falke, and Pierre Frenzel. Clone detection using abstract syntax suffix trees. In *Proceedings of the WCRE*, pp. 253–262, 2006.

[5] Yi Zhang and Zachary G Ives. Finding related tables in data lakes for interactive data science. In *Proceedings of the ACM SIGMOD*, pp. 1951–1966, 2020.