DEIM Forum 2021 A25-3

位置・キーワードに基づくTop-k Similarity Joinの高速アルゴリズム

鶴岡　翔平†　　天方　大地†　　新井　悠介†　　原　　隆浩†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1-5
E-mail: †{tsuruoka.shohei,amagata.daichi,yusuke.arai,hara}@ist.osaka-u.ac.jp

あらまし　GPS 付きのモバイル端末および IoT 環境の普及により，位置情報サービスでは，位置およびキーワード（テキスト）を共に含むデータを大量に生成している．これにより，位置・キーワードデータベースに多くの注目が集まっている．本稿では，位置・キーワードデータベースにおける重要なオペレータである Top-k spatio-textual similarity join 問題に取り組む．本問題は，類似度が最も大きいデータのペアを k 個出力する．この問題は，複製検出，推薦，およびクラスタリング等の応用を持つ．Top-k spatio-textual similarity join を処理する際の主なボトルネックはデータ間の類似度の計算である．この計算を可能な限り削減するため，既存研究ではデータ間の類似度の計算をスキップするためのフィルタを提案している．しかし，既存アルゴリズムは，初期の閾値が小さい，フィルタリングコストが大きい，また，多くのデータのペアを一度にフィルタリングできない，といった問題がある．本稿では，これらの弱点を克服し，高速に厳密解を出力するアルゴリズムを提案する．実データを用いた実験により，提案アルゴリズムは既存アルゴリズムよりも高速であることを示す．

キーワード　位置・キーワードデータ，Join

1 はじめに

GPS 付きのモバイル端末および IoT 環境の普及により，位置情報サービスがより実生活に馴染んでいる．例えば，Google Map，OpenStreetMap，およびソーシャルネットサービス（Facebook および Instagram 等）におけるジオタグは，日常的に利用されている．これらのサービスのユーザ（またはアプリケーション）は，位置情報（チェックイン情報等）およびキーワード（ハッシュタグ等）を共に含むデータを大量に生成している．これらのデータは，（サービスが）類似検索 [4], [6], [7], [13], [14], [15], [16], [19], [20] を提供することによって，サービスの質の向上，利益の増加，およびイベント検出等に用いられていることから，位置･キーワードデータベースに注目が集まっている．一方で，実世界における位置･キーワードデータベースには以下のような問題がある：データベース内のデータは複数の（異なる）ソースから収集されており [9], [10]，よく似た，または全く同一のデータが含まれてしまう．このとき，いくつかのデータは複製されていたり，似ているが（GPU エラーや誤字等によって）異なる情報で表されている場合がある．この状況下では，データベース内には無駄なデータが大量に存在してしまい，類似検索の質が低下してしまう．そのため，冗長なデータを取り除いてデータベースをクリーニングすることが重要であり，これを実現する spatio-textual similarity join [3] も同様に重要である．

文献 [3], [10], [12] は，閾値に基づく spatio-textual similarity join 問題に取り組んでいる．位置情報に関する閾値およびキーワード集合に関する閾値が与えられた時，あるデータのペアは，どちらの類似度も閾値以上である場合に類似していると言う．閾値に基づく spatio-textual similarity join は，このようなペアを全て検索する問題である．この問題は入力として 2 つの閾値を要求するが，これらを適切に指定することは現実的に難しい．例えば，閾値が小さい場合，解のサイズが大量になりかねない．一方で，閾値が大きい場合，解となるデータのペアが（ほぼ）無いということが起こり得る．この問題を解決するため，top-k spatio-textual similarity join 問題が考えられている [8]．この問題では，解のサイズを入力として要求するため，ユーザが望むサイズのデータのペアを出力できる．本稿では，top-k spatio-textual similarity join 問題に取り組む．

本問題における主なボトルネックは，データ間の類似度の計算である．この計算をできるだけ避けるため，文献 [8] では signature と呼ばれるフィルタリングを提案している．あるデータ $o \in O$（O はデータ集合）とある部分集合 $O' \subset O$ が与えられた時，このアルゴリズムでは，o の signature（空間領域とキーワードのペア）set を O' から計算する．2 つのデータ $o, o' \in O$ が類似する（解に含まれる）ためには，signature set 間に共通部分が必要であるという事象を用いて，このフィルタリングでは signature set に共通部分がないデータのペアを無視する．これにより，join 処理を高速化しているが，このアルゴリズムにも幾つか問題がある．まず，ランダムな k ペアから解の閾値を計算するため，閾値が小さく，フィルタリングを効率化できない．次に，signature set の共通部分を計算するコストも小さくないため，フィルタリングにかかる計算時間が大きい．最後に，必ず各 $o \in O$ に対してフィルタリングを実行しており，$O_i \times O_j$（$O_i, O_j \subset O$）内の全てのデータのペアを一度にフィルタリングできない．

本稿では，より高速に top-k spatio-textual similarity join 処理を行うアルゴリズムを提案する．提案アルゴリズムは，既存アルゴリズムのようにオンラインでインデックス（signature sets）を構築せず，オフラインで構築したインデックス（aR

木 [11] の拡張）を用いてフィルタリングを行う．提案アルゴリズムは以下のような特長を持つ：(1) 初期の閾値を低コストで精度良く計算する．(2) $O_i \times O_j$ 内の全てのデータのペアが解に含まれない場合，これらを一度にフィルタリングする．(3) signature set を用いずに，$o \times O_i$ 内の全てのデータのペアを（解に含まれない場合に）一度にフィルタリングする．本稿の貢献を以下に要約する．

- バッチフィルタリングを実現するため，aR 木を拡張したインデックスを提案する．
- 上のインデックスを用いた top-k spatio-textual similarity join を正確に解くアルゴリズムを提案する．
- 実データを用いた実験により，提案アルゴリズムの有効性を示す．

本稿は以下のように構成される．まず，2 章で本稿の前提知識を紹介し，3 章で提案アルゴリズムについて述べる．4 章で評価実験について述べ，5 章で関連研究を紹介する．最後に 6 で本稿をまとめる．

2 予 備 知 識

データ集合を O で表す．本稿では，O に更新は無いものとする．各データ $o \in O$ は，$o = \langle p, s \rangle$ で表される．ここで，$o.p$ は o の位置（$\langle x, y \rangle$）情報を示し，$o.s$ は o が持つキーワードの集合を示す．データのペア $\langle o_i, o_j \rangle$ の類似度を計算するため，まず，位置情報に関する類似度を定義する．

定義 1. データ o_i および o_j の位置情報に関する類似度 $sim_p(o_i, o_j)$ は，$sim_p = 1 - \frac{dist(o_i.p, o_j.p)}{dist_{max}}$ と定義する．ここで，$dist(o_i.p, o_j.p)$ は $o_i.p$ および $o_j.p$ 間のユークリッド距離であり，$dist_{max}$ は O が存在する空間における最大の距離である．

上の定義から，$sim_p(o_i, o_j) \in [0, 1]$ である．次に，キーワード集合の類似度について考える．本稿では，集合間の類似度として最も利用されているジャカード類似度を用いる．

定義 2. データ o_i および o_j のキーワード集合に関する類似度 $sim_s(o_i, o_j)$ は，$sim_s = \frac{|o_i.s \cap o_j.s|}{|o_i.s \cup o_j.s|}$ と定義する．

上の定義に基づき，$\langle o_i, o_j \rangle$ の類似度を定義する．

定義 3. データ o_i，o_j，および重みパラメータ $\alpha \in [0, 1]$ が与えられた際，これらのデータの類似度は，$\alpha \cdot sim_p(o_i, o_j) + (1 - \alpha) sim_s(o_i, o_j)$ と定義する．

定義から，α が大きい（小さい）場合は位置情報（キーワード集合）を重視していることが分かる．最後に，本稿における問題を定義する．

定義 4. データ集合 O，重みパラメータ α，および解のサイズ k が与えられたとする．Top-k spatio-textual similarity join 問題は，類似度が最も大きい k 個のデータのペア $\langle o_i, o_j \rangle \in O \times O$ $(i \neq j)$ を出力する．

先行研究 [8] と同様に，O はメインメモリにロードされている環境を想定する．本稿の目的は，定義 4 で紹介した問題を高速かつ正確に解くアルゴリズムを設計することである．

3 提案アルゴリズム

Top-k spatio-textual similarity join を高速に処理するためには，不要な類似度の計算を可能な限り削減する必要がある．既存アルゴリズムでは signature を用いてこれを実現しようとしているが，フィルタリングコストが大きい．また，signature では，$O_i \times O_j$ 内の全てのデータのペアが解に含まれない場合にも一度にフィルタリングできない．

メインアイデア. 解を高速に特定するためには，解に含まれないペアを一度に小さい計算コストでフィルタリングすることが望ましい．つまり，$O_i \times O_j$ 内の全てのデータのペアが解に含まれない場合，これらを一度にフィルタリングできる技術が重要である．ここで，τ を（途中）解の閾値とすると，上のフィルタリングは，データのペアの類似度の上界値を推定することにより実現できる．これを実現するためには，いくつか課題が存在する．まず，τ が大きくないと，フィルタリングできない場合が多発してしまう．しかし，精度の高い閾値を小さい計算コストで求めることも自明ではない．さらに，$O_i \times O_j$ 内の全てのデータのペアの類似度の上界値の求め方に関しても同様に自明ではない．

これらの課題を解決するため，aR 木を拡張し，閾値の計算および類似度の上界値の計算に利用する．閾値の計算では，類似度の大きいデータのペアを含む葉ノードを推定するモデルを提案する．また，類似度の上界値の計算は，以下の 2 つの事象を利用している：(1) R 木の構造により，2 つのノード間の距離の最小値を求めることができるため，位置情報に関する類似度の上界値を推定できる．(2) $O_i \times O_j$ 内の全てのデータのペアのキーワード集合に関する類似度はオフラインで計算できる．つまり，この値を aR 木に持たせることにより，それらのデータのペアの類似度の上界値を $\mathcal{O}(1)$ 時間で計算できる．また，本稿で提案する aR 木は，$o \times O_i$ 内のデータのペアをフィルタリングするように設計されている．

概要. まず，前処理として aR 木をオフラインで構築する．この前処理は一度のみ行い，提案する aR 木は任意の k および α に対応できる．そして，k および α が与えられた時，提案アルゴリズムは以下の技術を用いて解を特定する．

（1） Init-Threshold：類似度の大きいデータのペアを含む aR 木の葉ノードを推定し，閾値 τ および解を初期化する．

（2） Filter-1：O_i および O_j が与えられた時，$O_i \times O_j$ 内の全てのデータのペアの類似度の上界値が τ 以下であれば，これらのペアを無視する．

（3） Filter-2：o および O_j が与えられた時，$o \times O_j$ 内の全てのデータのペアの類似度の上界値が τ 以下であれば，これらのペアを無視する．

3.1 データ構造

本稿で提案する aR 木は，オリジナルの aR 木 [11] に対して，葉ノードでキーワード集合を管理するという点で異なる．以降では，ノードについて言及する際，提案する aR 木のノードを

指す．ノード n_i を根とする部分木で管理されているデータ集合を O_i とする．ノード n_i は以下で構成される．

- r_i: O_i に含まれるデータで構成される最小外接矩形．
- agg_i: O_i に含まれるデータのペア $\langle o, o' \rangle$ の $sim_s(o, o')$ の最大値．
- w_i: O_i に含まれるデータが持つキーワードの和集合（葉ノードのみ管理）．

ここで，agg_i は Filter-1 で，w_i は Filter-2 で利用される．また，aR 木はメインメモリで管理する．

オフラインアルゴリズム． aR 木の構築アルゴリズムを紹介する．aR 木を構築する際の主なボトルネックは，set similarity join を必要とする agg_i の計算である．この課題を既存の set similarity join アルゴリズム [17] を利用して解決する．

葉ノードの集合を N とする．まず，

（1） R 木を構築する．

（2） 各葉ノード $n_i \in N$ に対して，

（a） w_i を $\bigcup_{o \in O_i} o.s$ をスキャンして計算する．

（b） agg_i を O_i 上で PP-Join [17] を実行して計算する．

ここで，(2-b) において，PP-Join は入力として閾値が必要だが，0 が入力されるものとし，各データのペアの sim_s を計算するごとに閾値を更新するものとする．次に，各中継ノード n_j について考える．$O_i \subseteq O_j$ である時，$\mathrm{agg}_i \leqq \mathrm{agg}_j$ である．つまり，$n_j.C$ を n_j の子ノードの集合とすると，$n_i \in n_j.C$ のとき，$\mathrm{agg}_i \leqq \mathrm{agg}_j$ であるから，$\max_{n_i \in n_j.C} \mathrm{agg}_i$ を閾値として PP-Join を実行すれば，agg_j を効率的に求められる．そのため，

（3） “ボトムアップ順” で，各中継ノード n_j に対して，$\max_{n_i \in n_j.C} \mathrm{agg}_i$ を閾値として PP-Join を実行し，agg_j を計算する．

上の操作により，aR 木が効率的に得られる．

3.2 フィルタリング

本節では，Filter-1 および Filter-2 の詳細を紹介する．また本節では，τ は与えられているものとする（τ の計算は 3.3 節で述べる）．

Filter-1． 葉ノード n_i および n_j が与えられたと想定する．このとき，$O_i \times O_j$ 内の全てのデータのペアを一度の類似度の上界値の計算によりフィルタリングする．この上界値を導出するため，$dist(r_i, r_j)$ を 2 つの最小外接矩形 r_i および r_j 間の最小距離と定義する．また，n_i と n_j に共通する祖先ノードの中で，最も深いノードを $n_{i,j}$ とする．今，$usim(O_i, O_j)$ を，$usim(O_i, O_j) = \alpha(1 - \frac{dist(r_i, r_j)}{dist_{max}}) + (1 - \alpha)\mathrm{agg}_{i,j}$ とする．このとき，以下が成り立つ．

定理 1. 閾値 τ および 2 つの葉ノード n_i, n_j が与えられた時，$usim(O_i, O_j) < \tau$ であれば，$O_i \times O_j$ 内の全てのデータのペアは解に含まれない．

証明. 任意のデータのペア $\langle o, o' \rangle$（$o \in O_i$ および $o' \in O_j$）について考える．定義から，$dist(o, o') \geqq dist(r_i, r_j)$ である．また，$\mathrm{agg}_{i,j} \geqq \max_{o_a, o_b \in O_i \cup O_j} sim_s(o_a, o_b)$ であるため，$\mathrm{agg}_{i,j} \geqq sim_s(o, o')$ が成り立つ．そのため，$sim(o, o') \leqq usim(O_i, O_j)$ であり，$usim(O_i, O_j) \leqq \tau$ であれば，$sim(o, o') \leqq \tau$ となる．よって，本定理は成り立つ． □

上界値の定義から分かる通り，$usim(O_i, O_j)$ の計算コストは $\mathcal{O}(1)$ 時間であり，多くのデータのペアを効率的にフィルタリングできることが分かる．

Filter-2． 次に，Filter-1 でフィルタリングできなかった葉ノードのペア $\langle n_i, n_j \rangle$ について考える．ノード単位でフィルタリングできなかった場合でも，$o \times O_j$ 内の全てのペアをフィルタリングすることは可能である．これを特定する方法について述べる．

今，$dist(o, r_j)$ を $o \in O_i$ と r_j との最小距離と定義する．そして，$usim(o, O_j) = \alpha(1 - \frac{dist(o, r_j)}{dist_{max}}) + (1 - \alpha)\frac{|o.s \cap w_j|}{\min\{|o.s|, |w_j|\}}$ と定義する．このとき，以下が成り立つ．

定理 2. 閾値 τ，$o \in O_i$，および O_j が与えられた時，$usim(o, O_j) < \tau$ であれば，$o \times O_j$ 内の全てのデータのペアは解に含まれない．

証明. あるデータ $o' \in O_j$ が与えられた時，定義から，$dist(o, o') \geqq dist(o, r_j)$ である．また，$w_j = \bigcup_{o' \in O_j} o.s$ であるため，全ての $o' \in O_j$ に対して，$|o.s \cap w_j| \geqq |o.s \cap o'.s|$ が成り立つ．また，全ての $o' \in O_j$ に対して，$\min\{|o.s|, |w_j|\} \leqq |o.s \cup o'.s|$ であり，$\frac{|o.s \cap w_j|}{\min\{|o.s|, |w_j|\}} \geqq sim_s(o, o')$ が成り立つ．つまり，$usim(o, O_j) \geqq sim(o, o')$ となる． □

Filter-1 と比較すると，Filter-2 は積集合を計算するため，コストがやや大きい．しかし，一度の積集合の計算で $o \times O_j$ 内の全てのデータのペアをフィルタリングできるため，$o \times O_j$ 内の全てのデータ間の類似度計算と比べると非常に効率的である．

3.3 オンラインアルゴリズム

3.1 節および 3.2 節で aR 木により効率的に多くのデータのペアをフィルタリングできることを示した．残る課題は，閾値の初期値を高精度かつ高速に求めることである．

Init-Threshold． 閾値 τ の初期化においても aR 木を用いる．最小外接矩形 r_i における左下および右上の点をそれぞれ p_i^l および p_i^u とする．この時，関数 $f(n_i)$ を $f(n_i) = \alpha \cdot (1 - \frac{dist(p_i^l, p_i^u)}{dist_{max}}) + (1 - \alpha)\mathrm{agg}_i$ と定義する．最小外接矩形が小さい，または，agg_i が大きいとき，O_i 内に類似度の大きいデータのペアが存在しやすく，また，$f(n_i)$ は大きくなる．

上の事象を利用して閾値 τ を計算する．各葉ノード $n_i \in N$ に対して $f(n_i)$ を計算し，その値が大きい l 個の葉ノードを N_l で管理する．次に，各 $n_j \in N_l$ に対して，O_j における self-join を行い，$sim(\cdot, \cdot)$ が最も大きい k 個のデータのペアを暫定解とし，τ を得る．

1 章で紹介した通り，spatio-textual similarity join 問題のアプリケーションでは多くの複製データが存在する．このようなデータは，同じ葉ノードで管理される傾向にある．そのため，全てのペアを列挙せずとも，Init-Threshold で実行している “ローカル” の join（self-join）により，少ないペア数の列挙で高精度な閾値を得ることができる．

Join アルゴリズム．ここから，これまで紹介した技術を実装した提案アルゴリズムを紹介する．

（1） まず，Init-Threshold で解を初期化し，閾値 τ を得る．

（2） 次に，葉ノードのペア $\langle n_i, n_j \rangle$ について考える．

（a） これらのノードに対して Filter-1 を実行し，$usim(O_i, O_j) \leqq \tau$ であれば，全ての $O_i \times O_j$ を無視する．

（b） そうでない場合，各 $o \in O_i$ と O_j に対して，Filter-2 を実行し，$usim(o, O_j) \leqq \tau$ であれば，全ての $o \times O_j$ を無視する．

（c） そうでない場合，各 $o' \in O_j$ と $sim(o, o')$ を計算し，τ と解を更新する．

（3） （2）の操作を，各葉ノードのペアに対して実行する．

ここで，$n_i \in N_l$ であった場合，Filter-1 において $\langle n_i, n_i \rangle$ は無視する．この条件と定理 1 および 2 から，提案アルゴリズムは厳密解を保証する．

4 評価実験

本章では，提案アルゴリズムの性能評価のために行った実験の結果を紹介する．全ての実験は Ubuntu 16.04 LTS OS, 3.00GHz Intel Xeon Gold 6154，および 512GB RAM を搭載した計算機で行った．

4.1 設　　定

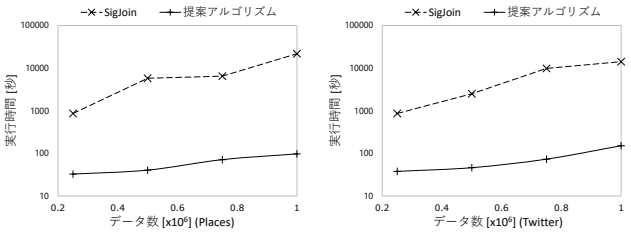
本実験では，Places[1] と Twitter[2] の 2 つの実データを用いた．データ数はそれぞれ 1.000,000 であり，Places および Twitter はそれぞれ 26,407 および 277,849 種類のキーワードを持つ．また，1 つのデータが含むキーワード数の平均は，Places および Twitter でそれぞれ 2.9 および 4.4 である．

本実験では，既存アルゴリズムの SigJoin [8] と提案アルゴリズムを評価した．両アルゴリズムはシングルスレッドで動作し，g++ 5.4.0 と O3 フラグでコンパイルした．

デフォルトとして，$k = 100$，$\alpha = 0.5$ とした．あるパラメータの影響を調べる際は，それ以外のパラメータを固定している．また，予備実験から，$k < 100$ の時は $l = k$，$k \geqq 100$ の時は，$l = k/2$ とした．

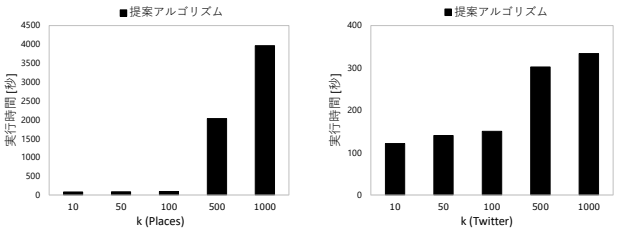
4.2 実験結果

データ数の影響．各アルゴリズムのスケーラビリティを調べるため，データ数を変えた実験を行った．図 1 は，実験結果を示している．この結果より，提案アルゴリズムは SigJoin よりもデータ数に対して大幅にスケーラビリティが高いことが分かる．例えば，データ数が 1,000,000 の時，Places（Twitter）において，提案アルゴリズムは SigJoin よりも 224（94）倍高速である．この性能差は，Init-Threshold（高精度な閾値を小さいコストで計算）および Filter-1（不要なデータのペアを一度にまとめてフィルタリング）に特に起因している．これらのアプローチにより，提案アルゴリズムは SigJoin と比べて大幅に少ない類似度の計算回数で正確な解を出力している．（SigJoin の性能は常に提案アルゴリズムよりも悪かったため，以降では SigJoin の結果を省略する．）



(a) Places　　(b) Twitter

図 1　データ数の影響



(a) Places　　(b) Twitter

図 2　k の影響

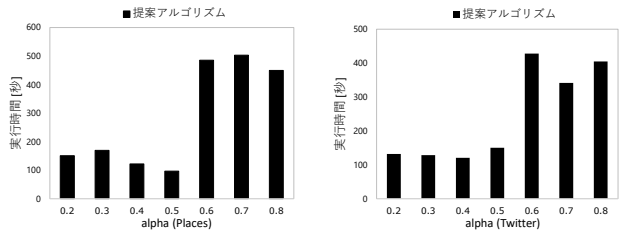
k の影響．次に，出力サイズの影響について調べる．図 2 は k が実行時間に与える影響を表している．この結果から，k が大きくなると実行時間が長くなることが分かる．これは，k が大きくなると Init-Threshold で得られる閾値が小さくなり，探索空間が大きくなってしまうためである．また，k が大きいと，Places と Twitter で実行時間が大きく異なる．Places は Twitter と比べて，上位 k 番目の閾値付近の類似度を持つ（類似度が大きい）データのペアが多く存在していることが観測された．この性質のため，k が大きい時に類似度の計算回数が多くなり，実行時間が長くなってしまう．

α の影響．最後に，α を変えた実験の結果を，図 3 により紹介する．Places および Twitter の結果から，α が小さい場合（キーワード集合の類似度への重みが大きい場合）の方が実行時間が短いことが分かる．この場合，位置が近く，かつ，キーワード集合が似ているデータのペアの類似度は非常に高くなる．また，このようなペアの数は全てのペアに対して非常に少ないため，Init-Threshold でこのようなペアを計算した後，枝刈り効率が高くなり，実行時間が短くなる．一方，α が大きい場合は位置の近さを重視する．位置が近いデータのペアは大量に存在するため，枝刈り効率が下がり，実行時間が長くなりやすい．

5 関連研究

本稿で取り組んだ問題は，複製検出のみでなく，位置情報に基づく推薦 [12] やクラスタリング [2] にも応用されている．本問題における既存アルゴリズムは 1 章において既に紹介しているため，本章では閾値に基づく join 問題の既存アルゴリズムに

1：https://archive.org/details/2011-08-SimpleGeo-CC0-Public-Spaces
2：http://www.ntu.edu.sg/home/gaocong/datacode.html



(a) Places (b) Twitter

図 3 α の影響

ついて紹介する．

文献 [3] において閾値に基づく spatio-textual join 問題が提案されており，set similarity join 問題のアルゴリズム [18] を拡張したアルゴリズムを提案した．文献 [10] は，signature と呼ばれるフィルタリングアルゴリズムを提案している．しかし，この文献では，位置情報を点ではなく領域として定義しており，本稿の想定と異なる．これらの研究では単一の計算機を想定しているが，MapReduce 環境におけるアルゴリズムも提案されている [1], [12]．これらの研究はデータ集合の分割方法に関して考慮しており，本稿で考慮するべき課題と異なる．

次に，join でなく search に関する既存研究に着目する．この問題も多くの研究で取り組まれており，特に，クエリと最も類似する k 個のデータを検索する問題 [4], [6], [7], [13], [14], [15], [16], [19], [20] に注目が集まっている．これらの研究では，位置情報とキーワード情報を組み込んだデータ構造（インデックス）を設計しており，文献 [5] では性能の比較実験が行われている．本稿における提案アルゴリズムでも類似したインデックスを設計しているが，提案アルゴリズムは先述の search に関するアルゴリズムとアプローチが異なる．提案アルゴリズムは，多くのデータのペアを一度にフィルタリングするが，これは search に関するアルゴリズムで考えるポイントと異なる．また，search 問題を各データをクエリとして実行することで本稿での問題を解くこともできるが，このアプローチは非効率的であることが文献 [8] において示されている．

6 まとめ

位置・キーワードデータベースが多くのアプリケーションによって重要であるという事実から，本稿では top-k spatio-textual similarity join 問題に取り組んだ．本稿で提案したアルゴリズムは，aR 木を拡張したインデックスを用いて解に含まれないデータのペアを一度に大量にフィルタリングすることで処理を高速化している．また，初期の閾値を低コストで精度良く計算することにより，フィルタリング効率を高めている．2 つの実データを用いた実験により，提案アルゴリズムの有効性を確かめた．

謝辞

本研究の一部は，文部科学省科学研究費補助金・基盤研究 (A)(18H04095)，JST さきがけ（JPMJPR1931），および JST CREST（J181401085）の支援を受けたものである．

文献

[1] J. Ballesteros, A. Cary, and N. Rishe. Spsjoin: parallel spatial similarity joins. In *SIGSPATIAL*, pages 481–484, 2011.

[2] A. Belesiotis, D. Skoutas, C. Efstathiades, V. Kaffes, and D. Pfoser. Spatio-textual user matching and clustering based on set similarity joins. *The VLDB Journal*, 27(3):297–320, 2018.

[3] P. Bouros, S. Ge, and N. Mamoulis. Spatio-textual similarity joins. *PVLDB*, 6(1):1–12, 2012.

[4] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

[5] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: an experimental evaluation. *Proceedings of the VLDB Endowment*, 6(3):217–228, 2013.

[6] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[7] C. Doulkeridis, A. Vlachou, D. Mpestas, and N. Mamoulis. Parallel and distributed processing of spatial preference queries using keywords. In *EDBT*, pages 318–329, 2017.

[8] H. Hu, G. Li, Z. Bao, J. Feng, Y. Wu, Z. Gong, and Y. Xu. Top-k spatio-textual similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 28(2):551–565, 2016.

[9] S. Liu, G. Li, and J. Feng. Star-join: spatio-textual similarity join. In *CIKM*, pages 2194–2198, 2012.

[10] S. Liu, G. Li, and J. Feng. A prefix-filter based method for spatio-textual similarity join. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2354–2367, 2014.

[11] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient olap operations in spatial data warehouses. In *SSTD*, pages 443–459, 2001.

[12] J. Rao, J. Lin, and H. Samet. Partitioning strategies for spatio-textual similarity join. In *SIGSPATIAL Workshop*, pages 40–49, 2014.

[13] J. B. Rocha-Junior, O. Gkorgkas, S. Jonassen, and K. Nørvåg. Efficient processing of top-k spatial keyword queries. In *SSTD*, pages 205–222, 2011.

[14] Y. Tao and C. Sheng. Fast nearest neighbor search with keywords. *IEEE transactions on knowledge and data engineering*, 26(4):878–888, 2013.

[15] G. Tsatsanifos and A. Vlachou. On processing top-k spatio-textual preference queries. In *EDBT*, pages 433–444, 2015.

[16] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE Transactions on Knowledge and Data Engineering*, 24(10):1889–1903, 2011.

[17] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *WWW*, pages 131–140, 2008.

[18] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems*, 36(3):1–41, 2011.

[19] C. Zhang, Y. Zhang, W. Zhang, and X. Lin. Inverted linear quadtree: Efficient top k spatial keyword search. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1706–1721, 2016.

[20] K. Zheng, H. Su, B. Zheng, S. Shang, J. Xu, J. Liu, and X. Zhou. Interactive top-k spatial keyword queries. In *ICDE*, pages 423–434, 2015.