D-009

# 空間類似結合における K 近傍グラフ作成の効率化

On Constructing a KNN graph under STSjoin

今井健太郎 *　大森匡*　新谷隆彦*　藤田秀之*

Kentaro Imai　Tadashi Ohmori　Takahiko Shintani　Hideyuki Fujita

## 1　はじめに

現実世界を表すデータは位置情報とテキスト両方を持つものが多い．例えば Flickr では，個々の写真データは撮影場所の緯度経度と内容を表すタグを持つ．このようなデータ集合 $D$ から距離 $\epsilon$ 以下で内容類似度 $\theta$ 以上となる組 $(x, y)$ を全て求める演算が空間類似結合 (STSjoin)[1] である．一方，STSjoin で各 $x \in D$ について類似度の高い順に $k$ 個の $y \in D$(KNN 解) を求める，K 近傍 (KNN) グラフ計算も情報抽出には重要である．本稿はこの計算の効率化を扱う．

一般に KNN 解は非対称なため，従来の類似結合用の全解計算手法 PPjoin[2] を単純に KNN 解用に修正すると解が脱落する [4, 5]．発表者らは STSjoin で Prefix Filtering 法 [3] など古典的な技法の戦略を計算中に強化する技法なども試してきたが，STSjoin では特に空間分割したセル間での計算時の照合処理とインデックス量削減が課題であった [5]．本稿では，類似結合の KNN グラフ計算法 L2Knng[4] を空間類似結合の枠組みに合わせて変形適用した技法を述べ，上記の課題を解決できることを示す．

## 2　問題の定義と従来技法

### 2.1　空間類似結合 (STSjoin) の定義

データ集合 $D$ のレコード $x$ は 3 つ組 $(x.id, x.loc, x.text)$ とする．ここで $x.id$, $x.loc$ はそれぞれレコードの識別番号, 位置情報である．$x.text$ は $x$ の持つテキスト情報であり，そのテキストに出現する項 (タグ，単語) の集合で表す．また $x, y \in D$ について，$dist(x, y)$ を $x, y$ 間のユークリッド距離，$simO(x, y)$ を $x.text, y.text$ 間の overlap 類似度 (各項 $t$ に重み $w(t)$ が与えられているとして，overlap 類似度 $simO(x, y) = \sum_{t \in x.text \cap y.text} w(t)$) とする．この時 $D$ 上の STSjoin とは，距離閾値 $\epsilon$，類似度閾値 $\theta$ を与えたとき，$dist(x, y) \leq \epsilon \wedge simO(x, y) \geq \theta$ を満たす組 $(x, y)$ を全て求める演算である．本稿では STSjoin の K 近傍 (KNN) グラフ計算，すなわち各 $x$ について STSjoin の制約を満たす $y$ のうち，類似度上位 $k$ 個の $y$(KNN 解) を求める問題を扱う．以下，類似度閾値 $\theta$ を $\theta_0$ とし，$x$ から見て類似度上位 $k$ 番目の $y$ との類似度を $\theta_{x,k}$ と記す．(重み $w(t)$ は idf を用いる．)

STSjoin の全解計算手法に，空間をグリッド分割し高速化する手法があり [1]，空間データを $\epsilon \times \epsilon$ のセルに分割し，左下隅から順にセル $R$ をスイープしながら，$R$ 自身と，$R$ の左下, 下, 右下, 左の隣接セル $S_1, S_2, S_3, S_4$ との間で距離制約下での類似結合を行う．本稿もこの枠組みに従う．

### 2.2　類似結合の全解計算の従来技法

**Prefix Filtering 法** (PF 法)　データ集合 $D$ 上の類似結合とは，類似度閾値 $\theta$ を与えた時，$simO(x, y) \geq \theta$ を満たす組 $(x, y)$ を全て求める演算である．類似結合の戦略の一つに Prefix Filtering(PF) 法 [3] がある．$D$ に出現する各項 $t$ に重み $w(t)$ を与える．重みの昇順にソートされた $x.text$ の項 $t_{x1}, t_{x2}, ..., t_{xv}$ について，$\beta(\theta) = \mathrm{argmin}_k(\sum_{j=1}^{k} w(t_{xj}) \geq \theta)$ 番目から末尾までの項 $t_{\beta(\theta)}, t_{\beta(\theta)+1}, ..., t_{xv}$ を $x$ の probe prefix($ppref(x)$) と呼ぶ．$simO(x, y) \geq \theta$ を満たす $x, y$ は $ppref(x)$ の項を一つ以上共有する．PF 法ではこれを用い，各項 $t_{xj}$ のインデックス $I[t_{xj}]$ の登録データ数を減らし，かつ，$x$ によるインデックスのプローブ範囲も削減する．(本稿では $x$ の末尾から順に $ppref(x)$ の範囲の項でインデックスをプローブする場合で述べる．)

**PPjoin**　類似結合の全解計算法の代表に PPjoin[2] がある．PPjoin は各レコード $x$ の重さ (=$x$ の全項の重み和) の昇順に，各 $x$ についてインデックス登録済みレコードとの間でプローブ（探索照合）による解候補生成と検証を行い，その後で $x$ をインデックスへ登録する．プローブ処理は $ppref(x)$ を用いて行い，途中見つけた解候補 $y$ について position filter などのフィルタを用いて候補削減した後に検証する．$x$ のインデックス追加の際は，index prefix $ipref(x)(\subseteq ppref(x))$ を用いてインデックス量を減らす．

## 3　STSjoin の KNN グラフ計算手法

**PF/Knng**　STSjoin で KNN 解を求める時の自明な戦略の 1 つは，$x$ からのプローブ中に見つかる暫定的な KNN 解候補との類似度 $\theta_{x,k}$ を使って $ppref(x)$ を短縮修正してプローブ範囲を狭めていくことである．これを probe prefix

*電気通信大学大学院情報理工学研究科
The Univ. of Electo-Communications

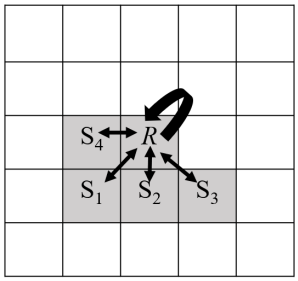


図 1: L2KnngSimple0 のセル間プローブ．自セル $R$ の位置をスイープしながら，$R \to R$ プローブ，$R \to S$ プローブ，$S \to R$ プローブを順に行う．

の実行時短縮処理と呼ぶ．しかし，PPjoin は解の対称性を前提とした手続きだが，KNN 解は非対称のため，この短縮処理を PPjoin の候補生成部に直に組み込むと解が脱落する [4, 5]．そこで，PPjoin ではなく PF 法にのみ基づき，初めに $\theta_0$ により全 $x$ をインデックス登録した後，各 $x$ からインデックス $I[t_{xj}]$ の探索・候補生成を行い，候補 $y$ が見つかるたびに検証 ($simO()$ の計算) と $\theta_{x,k}$ の修正をやり，$I[t_{xj}]$ 処理終了の度に $\theta_{x,k}$ により $ppref(x)$ を短縮する，という非対称型の計算方法ができる．本稿ではこの手法を PF/Knng と呼んでおく.[1]　先行して試した評価 [5] では，PF/Knng では解の非対称性により，グリッド分割した時の隣接セル間プローブが $R \leftrightarrow S$ の双方向で必要なことと，$\theta_0$ を下げると探索・候補生成時の照合コスト（$simO()$ の計算回数) が膨大になる点が課題だった．

**PPjoin/Knng**　PF/Knng の課題を受け清水・今井らは，対称型計算を使う PPjoin を基に，position filter を KNN 解用に強化し解候補を削減する方式 (PPjoin/Knng と呼ぶ) を試みた [5]．PPjoin/Knng は PF/Knng に比べ，隣接セル間プローブに伴う $simO$ の計算回数コストを最大 1/10 に削減できた．一方，計算時間自体は PF/Knng と同程度であり，効率化にはインデックス量の削減自体が必要なことが分かっている．

## 4　STSjoin への L2Knng の適用

類似結合で KNN グラフを効率的に計算する手法に L2Knng[4] がある．L2Knng では予め初期解を計算しておき，データ集合 $D$ を初期解の $\theta_{x,k}$ の昇順にソートし，この順に各レコード $x$ につき次の (i) から (iii) を行う：
(i)　$x$ からインデックスへのプローブ処理（＝候補生成）．(ただし，プローブする範囲 $ppref(x)$ は，初期解の $min(\theta_{z,k})(z \in D)$ 対応で作る, posfilter 等もあり),

[1] 本稿の PF/Knng では，さらに，PPjoin の position filter(posfilter) を KNN 解計算用に修正したフィルタも用いて解の候補削減を行うことと，$I[t]$ のエントリを posfilter の降順ソートして可能性のないスキャンの打ち切りも入れた．また，本稿を通して，2 点間距離制約等による候補削減は候補生成時に行っている．

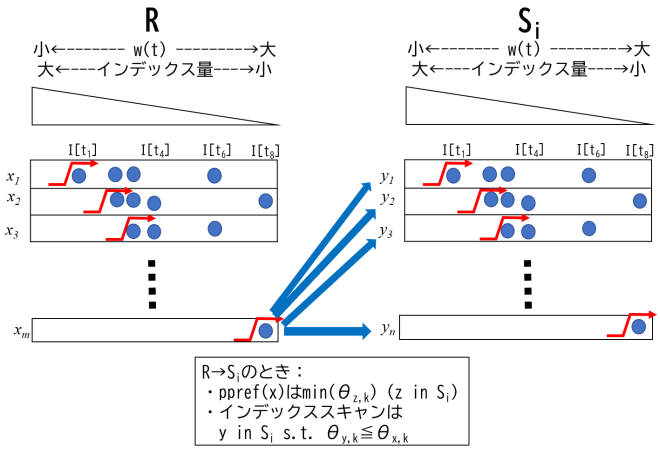


図 2: L2KnngSimple0 の $R \to S$ プローブ処理

(ii)KNN 解候補検証 ($x$ からインデックスへのプローブで見つかった $y$ について，$x$ からの KNN 解に $y$ が入るか，$y$ からの KNN 解に $x$ がなるか，について，適当なフィルタ条件の下で類似度 $simO(x, y)$ を計算し KNN 解を更新),
(iii) $x$ のインデックス登録 ($ipref(x)$ は初期解の $\theta_{x,k}$ 対応で作る).
以上により，L2Knng は KNN 解を落とさずにインデックス量を削減できる．そこで以下では，L2Knng を STSjoin に変形適用する方法を案出して，それによる STSjoin 下のインデックス量削減と効率改善の効果を調べる．

以下，L2Knng を STSjoin に合わせて変形適用する方法として，L2KnngSimple0(以降は Simple0 と略記) を述べる．Simple0 の手順を以下の (1)〜(3.3) で与える：

(1) データ集合 $D$ の初期解を求める．本稿では，各 $x$ について $simO(x, y) \geq \theta_0$ を満たす $y$ が $k$ 個見つかった時点で $x$ からのプローブを終了する，という条件で PF/Knng を動かし初期解を求めた．(この方法で KNN 解となる相手が $k$ 個いない $x$ は正解確定であり，以後の処理からは削除．)
(2) $D$ を，初期解の $\theta_{x,k}$ の昇順にソートした後，幅 $\epsilon$ のグリッド分割を行い，各セルにレコードを割り当てる．(図 1 を参照)
(3) セル $R$ を左下端のセルから（図 1 に従い) スイープしながら，自セル $R$ とその隣接セル $S_1$，$S_2$，$S_3$，$S_4$ について以下の (3.1)〜(3.3) を行う．
(3.1) $R \to R$ プローブ: $R$ のデータ集合に L2Knng を用いる．$x \in R$ からのプローブで用いる $ppref(x)$ は，$R$ の初期解の $min(\theta_{z,k})(z \in R)$ によって決める．($R$ に閉じた範囲での距離制約下の KNN 解の候補生成と検証になる)
(3.2) $R \to S$ プローブ: $R$ の各レコード $x$ から $S$ のインデックスへ L2Knng のプローブ (と候補生成，検証処理) を行う．このとき，$ppref(x)$ は $S$ の初期解の $min(\theta_{z,k})(z \in S)$ に応じて作り，(3.3) との重複を避けるため，$x \in R$ からは，$S$ に含まれる $y$ のうち $\theta_{y,k} \leq \theta_{x,k}$ の範囲の $y$ のみを

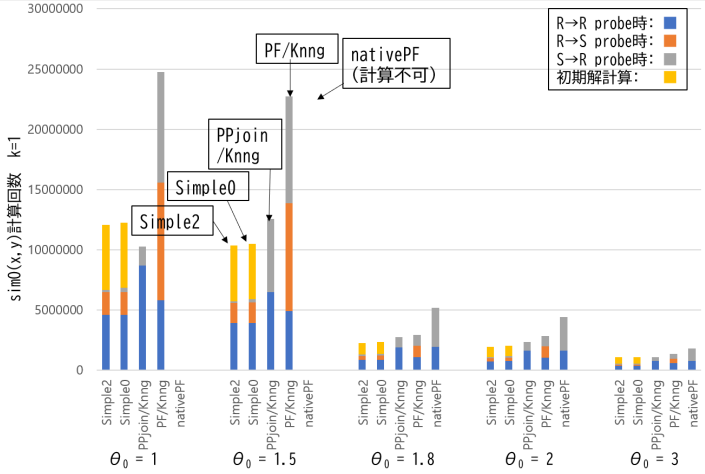


図 3: $simO(x, y)$ 計算回数 ($k$=1), 横軸:$\theta_0$, 縦軸:計算回数 (回)

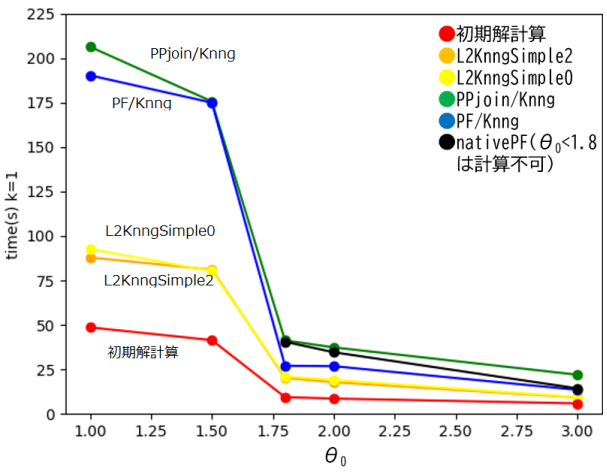


図 4: 計算時間 ($k$=1), 横軸:$\theta_0$, 縦軸:時間 (秒)

プローブの対象とする.（図 2 参照）

(3.3) $S \to R$ プローブ: $S$ の各レコード $x$ から $R$ のインデックスへ L2Knng のプローブ (と候補生成･検証処理) を行う．このとき，$ppref(x)$ は $R$ の初期解の $min(\theta_{z,k})(z \in R)$ に応じて作り，(3.2) との重複を避けるため，$x \in S$ からは，$R$ に含まれる $y$ のうち $\theta_{y,k} < \theta_{x,k}$ の範囲の $y$ のみをプローブの対象とする．

---

Simple0 の特徴は，セル $R$ のインデックス作成と $R$ に閉じた KNN 解計算をした後で L2Knng におけるレコードの $\theta_{z,k}$ の大小関係順を守った隣接セル間の結合（＝照合と KNN 解候補生成）を行い，STSjoin に特有なセルあたりのレコードのインデックス作成を 1 回だけ行う設計にしたことである．また，元の L2Knng 自体はレコード $x$ によるプローブをインデックス側のデータ集合の KNN 解の閾値の最小値 $min(\theta_{z,k})$ に応じた $ppref(x)$ まで行うため，ブロック分割を使ってプローブ範囲を削減する工夫を入れる [4] が，Simple0 はこのブロック分割を行なっていない．STSjoin における L2Knng の初期解を利用したインデックス削減の効果，特に $simO$ の計算回数の削減と計算時間削減の効果を知ることが Simple0 の目的である．

また，この他に，Simple0 の改良版として，(3-1) の $R$-$R$ 間の照合処理直後にその時点の $\theta_{z,k}(x \in R)$ によって $R$ を再ソートしてインデックスを再作成する処理を入れることと，(3.2)(3.3) の $ppref(x)$ をその時点のインデックス側のセルの $min(\theta_{z,k})$ に応じて決めるように修正する，の 2 点の改良を行った L2KnngSimple2(Simple2 と略記) も作成した．

評価には，位置情報と写真内容を表すようにその写真の所有者=ユーザが付与したタグ集合を持つ Flickr 写真共有データサービス (www.flickr.com) から，2011 年 1 月から 2016 年 3 月までの東京・神奈川を含む領域指定でデータ収集（243890 件）し，そこからランダムに選んだ 10 万件を用いた．Flickr では，レコード 1 つが写真 1 つを表し，レコードあたり所有者であるユーザ id，座標，タグ集合，写真 url，等がある．このレコード集合 10 万件に，同一座標で同一ユーザのレコード群を 1 レコードに集約する処理をかけて，結果，67572 件のレコード集合を用いた．また，相異なるユーザ間のレコードのみを KNN 解の対象とするように設定した.（全方式で，距離制約と同一ユーザ判定による候補の削減は候補生成部で行うよう統一している.）出現タグ総数は 47221 種類，1 タグを 1 項として扱い，出現頻度 1%のタグが 1 点になるよう係数調整した idf 値でタグを重みづけした．レコードあたりのタグ重み総和の平均値は 7.19，中央値は 3.81 である．$\epsilon$ = 4km に固定し，$\theta_0$ は有用な KNN グラフを求められる範囲を意図して $\theta_0 = 3$ から 1 までとして，$k = 1$ と 5 を試した．各方式の作成と実行は 3.0GHz の Intel Core i7-9700 上の Python3.6.8 で統一し，Simple0, Simple2, PF/Knng, PPjoin/Knng，の各手法について，$R$-$R$ セル内の処理コスト，$R$ から隣接セル $S$ を照合するときの計算コスト，$S$ から $R$ を照合する計算コスト，を調べた．計算コストは，検証フェーズにおける $simO(x, y)$ の計算回数と，CPU 計算時間の 2 つを用いる．

$k$ = 1 の場合の $\theta_0$ =3 から 1 までに変えた場合について，図 3 に $simO(x, y)$ 計算回数を，図 4 に計算時間を示す．(図中の NativePF は，全解計算法として PF 法をそのまま対称解計算方式で用いた方式であり，$\theta_0 \leq 1.5$ では計算不能.)

図 3 の $simO$ 計算回数で見ると，PF/Knng に比べて PPjoin/Knng は 50%程度削減できているが，図 4 では PPjoin/Knng と PF/Knng は計算時間ではほとんど変わらない．これは，3 節で述べたように，PPjoin/Knng でも $\theta_0 = 1.5$ 付近の小さい値ではそれに応じた項のインデックスの登録レコード数が多くなりすぎ，posfilter などで $simO$ 計算不要と判定できる組は多くても，プローブ処理時のインデックスのスキャン処理が主要計算時間になるからである．

一方，図 3 における Simple0 と Simple2 各々の $simO$ 計算回数は，特に低い $\theta_0$（=1.5 から 1.0）では，初期解計算のコストと合わせても PPjoin/Knng と同程度に削減できており，しかも，Simple0（と Simple2）の計算時間は初期解計算の時間と合わせても PPjoin/Knng のそれより 4 割程削減できている（図 4）．この結果は，インデックス量の削減が直接的な処理コスト＝計算時間の削減になることを示しており，PPjoin/Knng で現れたインデックス量削減という課題を解決できたと言える．

他に，Simple2 は，全体としては Simple0 とほぼ同じ計算コストだが，$S \to R$ プローブの計算回数は $k$=1,$\theta_0$=1 で 33 万回から 19 万回と，最大 42%削減できたことが図 3 からわかっている．ただし，初期解計算後の Simple0, Simple2 それぞれの $\theta_0 = 1$ での $R-R$ 内と $R-S$ 間の $simO$ 計算回数の合計は 680 万回程度なので，Simple0 からの改良効果は微小である．

## 5 まとめ

本稿では空間類似結合における K 近傍グラフ作成の効率化を扱った．PPjoin/Knng で判明したインデックス量を削減したいという課題に対し，類似結合の手法である L2Knng を空間類似結合に合わせて変形適用した L2KnngSimple0, Simple2 でインデックス量削減を試みた．これにより $simO$ 計算回数を PPjoin/Knng と同程度に抑えつつ，計算時間を 4 割削減し効率化することができた．これにより，STSjoin で特に問題であった隣接セル間の照合処理コストを，対称解計算方式の枠組みを使う L2Knng の戦略によってほぼ解決できることがわかった．$R \to R$ の計算コストと初期解計算のコストがオーバーヘッドとして残っており，これらの削減が今後の課題である．

謝辞: 本研究発表の一部は科研費 23K11115 の助成による．

## 参考文献

[1] P. Bouros, S. Ge, and N. Mamoulis, “Spatio-Textual Similarity Joins,” Proc.VLDB, Vol.6(no.1), pp.1-12, 2012.

[2] C. Xiao, W. Wang, X. Lin, J. X. Yu, “Efficient similarity joins for near-duplicate detection,” WWW’08, pp.131-140, 2008.

[3] S. Chauduri, V. Ganti, R. Kaushik, “A Primitive Operator for Similarity Joins in Data Cleaning,” ICDE, pp.5-16, 2006.

[4] D. Anastasiu, G. Karypis, “L2Knng:Fast Exact K-Nearest Neighbor Graph Construction with L2-Norm Pruning,” CIKM’15, pp.791-800, 2015.

[5] 清水, “空間類似結合における各点あたり上位 K 解計算法に関する研究,” 電気通信大学卒業論文, 2022 年 3 月．