情報処理学会論文誌　データベース　Vol.10　No.2　8–18 (June 2017)

位置・ソーシャル関係・キーワードに基づく Top-k データモニタリング

西尾 俊哉[1,a)]　天方 大地[1,b)]　原 隆浩[1,c)]

受付日 2016年12月8日, 採録日 2017年4月7日

概要：近年，多くのアプリケーションでは，PoI（Point of Interest）がパブリッシュ/サブスクライブ（Pub/Sub）モデルに基づいてデータを発信しており，ユーザは生成されたデータの中から自身が興味を持つもののみを取得する．また，位置情報サービスやソーシャルネットワークサービスの普及により，位置やキーワード，ソーシャル関係を用いた検索への関心が高まっている．本研究では，Pub/Sub モデルで生成されたデータから，ユーザにとって有用な上位 k 個のデータ（Top-k データ）をモニタリングする問題に取り組む．この上位 k 個のデータを収集する際，PoI の位置，指定したキーワードとの一致度，およびデータを生成した PoI とのソーシャル関係からデータのスコアを計算する．データが発生した際に，すべてのクエリに対して Top-k データの更新をチェックする方法は，多数のクエリが存在する環境に対応できない．この問題を解決するため，クエリを四分木により管理し，発生したデータが上位 k 個となりうるクエリにのみアクセス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，提案アルゴリズムの有効性を示す．

キーワード：Pub/Sub，ソーシャルネットワーク，Top-k クエリ

Geo-social Keyword Top-k Data Monitoring

Shunya Nishio[1,a)]　Daichi Amagata[1,b)]　Takahiro Hara[1,c)]

Received: December 8, 2016, Accepted: April 7, 2017

Abstract: Recently, in many applications, PoIs have generated data objects based on Publish/Subscribe (Pub/Sub) model, and users receive only their preferable data objects. In addition, due to the prevalence of location based services and social network services, locations, keywords, and social relationships are considered to be meaningful for data retrieval. In this paper, we address the problem of monitoring k data objects that are the most relevant to users' preferences, where the score of a data object is calculated based on the location of PoI, the user-specified keywords, and the social relationship between the user and the PoI that generates the data object. If we have a lot of queries, it is time-consuming to access all queries when a data object is generated. To solve this problem, we propose an algorithm that maintains queries with a Quad-tree and accesses only queries with possibilities that a generated data object becomes top-k data objects. Our experiments using real datasets verify the effectiveness of our proposed algorithm.

Keywords: Pub/Sub, social network, Top-k query

1. はじめに

スマートフォンやタブレットなどの GPS を利用可能な端末の普及にともない，位置情報を考慮した検索が多くのアプリケーションにとって必要不可欠になっている [7], [13], [20]．その一例として，位置依存ソーシャル

[1] 大阪大学大学院情報科学研究科マルチメディア工学専攻
Department of Multimedia Engineering, Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] nishio.syunya@ist.osaka-u.ac.jp
[b)] amagata.daichi@ist.osaka-u.ac.jp
[c)] hara@ist.osaka-u.ac.jp

サービスである Yelp [*1] や Foursquare [*2] などがあげられる．これらのサービスでは，位置やキーワードに基づく検索をユーザに提供している．また，Facebook [*3] や Twitter [*4] に代表される SNS の急速な発展により，ソーシャル関係を考慮した検索も注目を集めている [1], [15]．具体的には，SNS におけるソーシャル関係を用いたソーシャルフィルタリングによりユーザの嗜好を抽出し，ユーザの要求に合う検索が実現できる [4]．たとえば，Facebook における “いいね” など，ユーザは興味のある PoI（Point of Interest）に対してソーシャル関係を持つ．この関係は，**図 1** におけるユーザと PoI 間の辺として表現でき，図 1 はユーザ u_1 および u_2 のソーシャル関係を表現している．このとき，ユーザ u_2 は PoI p_1 とソーシャル関係があるため，p_1 が発信するデータを受信する．また，ユーザ u_1 と u_2 は，ソーシャル関係がある PoI が類似しており，2 人の嗜好は類似しているといえる．このとき，p_1 は u_1 とソーシャル関係はないが，p_1 は u_2 とソーシャル関係があるため，u_1 の嗜好に合う可能性は高い．そこで，ソーシャル関係を考慮し，p_1 が発信するデータを u_1 が受信しやすくすることにより，u_1 は既知でなかった嗜好に合うデータを受信できる．つまり，ソーシャル関係を考慮することにより，あるユーザにとって既知でない嗜好に合うデータも検索可能になると同時に，PoI は広告配信などを効率化できる．

ここで，PoI からデータが与えられたとき，事前に登録された興味に基づいてユーザにデータを配信するパブリッシュ/サブスクライブ（Pub/Sub）モデルに基づくアプリケーションが多く存在する [6], [14]．Pub/Sub モデルとは，ユーザが自身の興味をクエリとして事前にシステムに登録しておくことで，データ発生源がデータを発信するとクエリに合うデータを受信することができるモデルである．この Pub/Sub モデルにおいて，ユーザの興味に合うデータをすべて配信すると，あるユーザは膨大な量のデータを受信し，またあるユーザはデータをほとんど受信しないという状況が生じる可能性がある．そのため，ユーザにとって有用な上位 k 個のデータを検索する Top-k 検索が実用的である [3]．また，Pub/Sub モデルでは，データが頻繁に生成され，あるユーザに対する上位 k 個のデータは頻繁に変化するため，ユーザに対する上位 k 個のデータをつねにモニタリングすることは重要である．

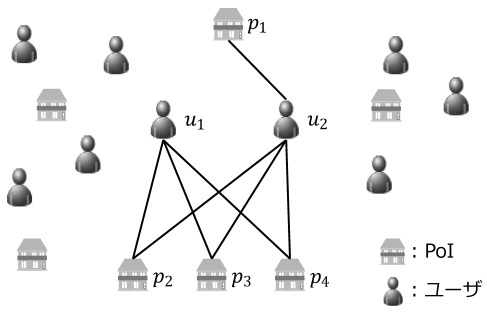


図 1　ユーザと PoI とのソーシャル関係
Fig. 1　Social relationships between users and PoIs.

[*1] http://www.yelp.com/
[*2] https://ja.foursquare.com/
[*3] https://www.facebook.com/
[*4] https://twitter.com/

本論文では，Pub/Sub モデルで生成されたデータから，ユーザにとって有用な上位 k 個のデータ（Top-k データ）をモニタリングする問題に取り組む．この Top-k クエリでは，PoI の位置，ユーザが指定したキーワードとの一致度，およびデータを生成した PoI とのソーシャル関係からデータのスコアを計算する．システムには多数の Top-k クエリが登録されており，Top-k データはクエリごとに異なる．そのため，クエリが多く存在している場合，すべてのクエリに対して Top-k データの更新をチェックする単純な方法は，多大な時間がかかってしまう．その結果，各ユーザの Top-k データを高速に更新できず，リアルタイム性を保証することが難しい．

この問題を解決するため，クエリを四分木 [17], [19] により管理し，生成されたデータが Top-k データとなりうるクエリにのみアクセスするアルゴリズムを提案する．四分木の各ノードには，そのノードを根とする部分木で管理されているクエリの情報を集約したものが保存されている．そして，ノードに保存された情報から，生成されたデータが Top-k データとなりえない部分木を枝刈りし，アクセスするクエリの数を削減する．これにより，データが生成されたとき，各ユーザの Top-k データを高速に更新できる．その結果，単純なアルゴリズムよりもアクセスするクエリの数を削減しつつ，正確に Top-k データを更新できる．実データを用いた実験の結果から，提案アルゴリズムの有効性を確認した．

以下では，2 章で本論文の問題を定義する．3 章で提案アルゴリズムについて説明し，4 章で実データを用いた実験の結果を示す．5 章で関連研究について述べ，最後に 6 章で本論文のまとめと今後の課題について述べる．

2. 問題定義

PoI の集合を P，すべての PoI が生成するすべてのデータの集合を O，発行されたすべてのクエリの集合を Q とする．

2.1 データモデル

ある PoI p が生成するデータ $o \in O$ は，o の id（$o.id$），o を生成した PoI の id（$o.pid$），データの位置情報（$o.loc$），およびキーワード集合（$o.key$）を保持している．$o.loc$ は o を生成した PoI の位置とし，PoI の位置情報は緯度と経度によって表される 2 次元平面上の点とする．データのキーワード集合は，そのデータを表現（説明）するキーワードの集合（メタデータの一種）であり，更新はないものとする [14]．

2.2　Top-k データモニタリング

あるユーザ u が発行するクエリ $q_u \in Q$ は，q_u の id（$q_u.id$），u が指定する位置情報（$q_u.loc$），1 つ以上の任意の数のキーワードの集合（$q_u.key$），および上位何個のデータを要求するかを指定する変数（$q_u.k$）を保持している．各クエリの Top-k データを決定するため，クエリ q_u に対するデータ o のスコア $s(q_u, o)$ を計算し，スコアの最も良い上位 k 個のデータを Top-k データとする．$s(q_u, o)$ を以下のように定義し，スコアが大きいほど優れているものとする．

$$s(q_u, o) = dist(q_u, o) + key(q_u, o) + socio(q_u, p) \quad (1)$$

Top-k 検索において，各属性の重みづけは困難であるため [5], [9], [16]，本研究では，重みづけを考慮する必要のない線形和を用いる．

$dist(q_u, o)$ は，式 (2) により，クエリとデータとのユークリッド距離 $d(q_u.loc, o.loc)$ に基づいて計算される．

$$dist(q_u, o) = 1 - \frac{d(q_u.loc, o.loc)}{MAXloc} \quad (2)$$

$MAXloc$ はユーザと PoI が存在しうる領域の最大距離であり，これによりスコアを $[0, 1]$ の値に正規化している．その値を 1 から引くことにより，距離が近いものほどスコアが大きくなる．

$key(q_u, o)$ は，式 (3) により，クエリのキーワードとデータのキーワードの一致度を F 値を用いて計算する．

$$key(q_u, o) = \frac{2|q_u.key \cap o.key|}{|q_u.key| + |o.key|} \quad (3)$$

たとえば，$q_u.key = \{w_1, w_2\}$ で，$o.key = \{w_2, w_3, w_4\}$ であるとき，$key(q_u, o) = \frac{2\times1}{2+3} = 0.4$ となる．

$socio(q_u, p)$ は，クエリを発行したユーザとデータ o を生成した PoI p とのソーシャル関係により計算される．本論文で想定する環境では，ユーザは，Facebook における “いいね” のように，興味のある PoI に対してソーシャル関係を持つ．この関係は，ソーシャルグラフを用いて**図 2** のように表現できる[*5]．丸がユーザ，四角が PoI を表し，ソーシャル関係はグラフにおける辺として表現できる．ここで，ソーシャルグラフの辺の集合を E とする．また，$e_{u,p}$ をユーザ u と PoI p 間の辺とする．ユーザ u と p が生成したデータ o とのソーシャルスコアは，式 (4) により計算される．

$$socio(q_u, p) = \begin{cases} 1 & (e_{u,p} \in E) \\ \max \frac{2|P_u \cap P_{u'}|}{|P_u| + |P_{u'}|} & (e_{u,p} \notin E) \end{cases} \quad (4)$$

ここで，$P_u = \{\forall p' \in P | \exists e_{u,p'} \in E\}$ であり，$P_{u'} = \{\forall p' \in P | (\exists e_{u',p} \in E) \wedge (\exists e_{u',p'} \in E)\}$ である．本論文における想定では，ユーザは興味のある PoI に対してソーシャル関係を持つため，$e_{u,p} \in E$ である場合，ソーシャルスコアを 1 と定義する．一方，$e_{u,p} \notin E$ である場合，p に興味のあるユーザ，つまり，$e_{u',p} \in E$ であるユーザ u' と u との PoI 間の辺の一致度を F 値を用いて計算する [11]．そして，一致度の最大値をソーシャルスコアとする．

[*5] グラフは重みなし無向グラフであり，辺の重みは考慮しない．

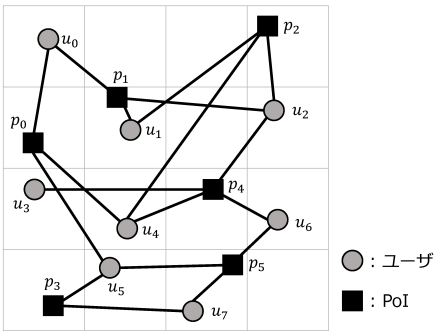


図 2　ソーシャルグラフ
Fig. 2　A social graph.

たとえば，図 2 において p_0 が o を生成した場合，$e_{u_0,p_0} \in E$ であるため，$socio(q_{u_0}, p_0) = 1$ である．次に，p_4 が o を生成した場合の $socio(q_{u_1}, p_4)$ について考える．ここで，$e_{u_1,p_4} \notin E$ である．p_4 との辺が存在するユーザは u_2，u_3，u_4 および u_6 であるため，それぞれのユーザとの PoI 間の辺の一致度を計算する．u_1 との辺が存在する PoI は p_1 および p_2 であり，u_2 との辺が存在する PoI は p_1，p_2，および p_4 であるため，u_1 と u_2 との PoI 間の辺の一致度は，$\frac{2\times2}{2+3} = 0.8$ となる．同様にして，u_3，u_4，および u_6 についても一致度を計算すると，u_1 と u_3 との一致度は 0，u_1 と u_4 との一致度は 0.4，また，u_1 と u_6 との一致度は 0 となる．よって，$socio(q_{u_1}, p_4) = \max\{0.8, 0, 0.4, 0\} = 0.8$ となる．

ここで，実際の環境ではソーシャル関係の更新（辺の追加および削除）が発生することが考えられる．この更新が起きると $socio(q_u, p)$ が変化する可能性がある．しかし，本研究ではソーシャル関係の更新は考えないものとし，更新が起きた場合においても，すでに計算済みの $socio(q_u, p)$ の値を利用するという方針をとる．

2.3　ベースラインアルゴリズム

正確に各クエリの Top-k データを更新する単純なアルゴリズムは，データが生成された際，すべてのクエリにアクセスし，データのスコアを計算するものである．そして，クエリが管理している k 番目のデータのスコアと比較し，新たに生成されたデータが，そのクエリの Top-k データとなるかを計算する．クエリが管理している k 番目のデータのスコアよりも新たに生成されたデータのスコアの方が大きければ，生成されたデータと k 番目のデータを置き換える．

このアルゴリズムでは，Top-k データが更新されないク

エリにもアクセスし，データのスコアを計算する．そのため，クエリが大量に存在している場合，各クエリの Top-k データを高速に更新できず，リアルタイム性を保証することが難しい．

3. 提案アルゴリズム

データが新たに生成されたとき，位置およびキーワードに基づくスコアの計算は生成されたデータに依存し，異なる PoI から生成されたデータであっても，PoI どうしの位置が近く，似たキーワード集合を持つデータであれば，位置およびキーワードに関するスコアは近い値となる．また，ソーシャルスコアはユーザと PoI の関係のみに依存し，同じ PoI から生成されたデータに対するソーシャルスコアはデータごとに変化しない．

そこで，クエリの位置情報に基づいてクエリを四分木で管理し，各ノードにはそのノードに含まれるクエリの id，キーワード集合，および k 番目のデータのスコアを集約した情報を保存する．四分木の構築の際，分割が終了したノードに対して，ノードの id，そのノードに含まれるクエリの id，および根ノードからそのノードまでの経路を表す数字列をノードリスト（L_N）に保存する．四分木が構築されると，各 PoI p_i に対応するソーシャルスコアリスト（L^i_{ss}）を作成する．L^i_{ss} は，L_N に含まれるノードごとに，ノード id およびそのノードに含まれるクエリのソーシャルスコアの最大値を保存したものである．クエリは事前にシステムに登録されており [3], [14]，四分木の構築，およびソーシャルスコアリストの作成はオフラインで行われる．

データが生成されると，クエリが管理している上位 k 個のデータが更新される可能性があるクエリにのみアクセスするためのチェックを行う．まず，データを生成した PoI の L_{ss} と L_N を組み合わせて，チェックするノードのソーシャルスコアを得るために用いるソーシャルスコアテーブル T_{ss} を作成する．そして，四分木の根ノードから順に，ノードに保存された情報とデータの位置情報およびキーワード集合から，生成されたデータが上位 k 個となりうるためのソーシャルスコアの閾値を計算する．T_{ss} から求められるノードのソーシャルスコア ss_n をこの閾値と比較することにより，生成されたデータが上位 k 個となりえない部分木を枝狩りする．これにより，アクセスするクエリの数を削減できる．

まず，3.1 節において，クエリを管理する四分木の構築方法を紹介する．次に，3.2 節において，各 PoI に対応する L_{ss} の作成方法を紹介する．最後に，3.3 節において，データが生成されたとき，クエリが管理している上位 k 個のデータを更新するアルゴリズムを紹介する．

3.1 四分木の構築

提案アルゴリズムでは，クエリを四分木を用いて管理する．四分木は，各ノードが 4 個までの子ノードを持つ木構造である．さらに，線形四分木を用いることにより，各ノードに対し，根ノードから自身のノードまでの経路を計算できる．四分木のノードの分割を**図 3** のように表現する．ノードを 4 つに分割する際，図 3 (a) に示すように，ノードを 4 つの象限に分割し，分割されたノードをそれぞれ，*NW*，*NE*，*SW*，および *SE* と名付ける．たとえば，図 3 (b) のように，4 つのクエリが存在し，領域が分割されたとする．この場合，四分木の構造は，図 3 (c) のような木構造をとる．図 3 (c) に示すように，本論文では，葉ノードと中継ノードを表現するために四角と丸を用い，クエリが含まれる葉ノードは黒四角とする．

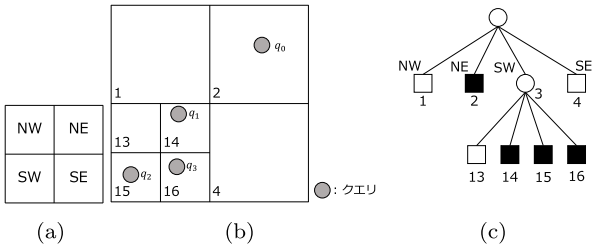
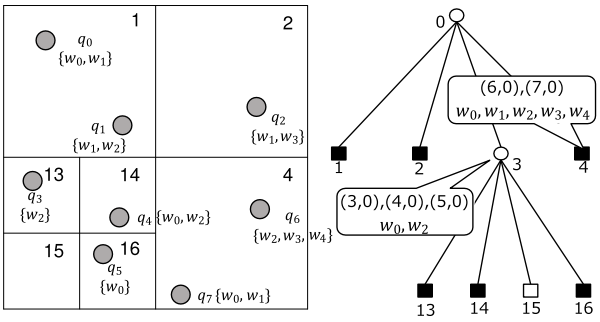


図 3　四分木の表現方法
Fig. 3　Illustration of a Quad-tree.

ここから，提案アルゴリズムにおける四分木の構築方法を紹介する．四分木を構築する際，クエリがどのノードに含まれるかはクエリの位置情報のみに依存する．まず，すべての PoI が存在している領域を根ノードとする．本研究におけるクエリは，すべての PoI が存在している領域内の任意の点を検索点とできるため，すべてのクエリが根ノードに含まれる．次に変数 m を用意し，ノードに含まれるクエリの数が m 個を超える場合は，そのノードを分割し，クエリの位置情報に基づいて各ノードにクエリを振り分ける．分割されたノード内のクエリの数が m 個を超えていれば同じ操作を行い，あるノードに含まれるクエリの数が m 個以下になるまで繰り返す．図 2 におけるユーザが，自身の現在地を検索点とし，何らかのキーワードを指定したクエリ q を発行したと仮定する．$m = 2$ の場合，**図 4** (a) に示す四分木が構築される．

各ノードには，そのノードに含まれるクエリの id，キーワードの集合，およびクエリが管理している k 番目のデータのスコアを保存する．ここで，クエリ q が管理する k 番目のデータのスコアを $q.score_k$ と表し，その初期値は 0 とする．データの生成によりクエリが管理している k 番目のデータが更新されると，そのクエリの情報を持つすべてのノードの情報が更新される．図 4 (a) では，クエリの id およびそのクエリの $q.score_k$ を（id，$q.score_k$）と表している．ノード 4 には，クエリ q_6 および q_7 が含まれているため，それらの id，k 番目のデータのスコア，およびキーワードの集合が保存されている．ノード 3 には，自身の子ノー



(a) 図 2 に基づく四分木

ノードid	クエリid	数字列
1	0,1	0
2	2	1
13	3	02
14	4	12
16	5	32
4	6,7	3

(b) L_N

図 4　四分木の構築（$m = 2$）および L_N
Fig. 4　Construction of Quad-tree ($m = 2$) and L_N.

ドであるノード 13，14，15，および 16 に含まれるクエリ q_3，q_4，および q_5 の id，k 番目のデータのスコア，およびキーワードの集合が保存されている．

四分木が構築される際，ノードに含まれるクエリの数が m 個以下となったノード（葉ノード）は，ノードの id，そのノードに含まれるクエリの id，および根ノードからそのノードまでの経路を表す数字列（ds）をノードリスト（L_N）に保存する．根ノードからそのノードまでの経路を表す数字列は，ノード id を用いて計算し，その結果を保存する．線形四分木では，自身のノード id から 1 を引いたものを 4 で除算した商が親ノードのノード id となり，余りが 0 ならば自身は親ノードの NW というように，余りによって自身は親ノードの NW，NE，SW，および SE のいずれであるかが分かる．この計算を繰り返し行うことにより順に求められた余りを 4 進数の数字列として保存する．たとえば，ノード 14 では余りが 1，2 という順に求まるため，12 という数字列を保存する．四分木の構築が完了すると，L_N には，クエリが含まれるすべての葉ノード（図 4 (a) における黒四角）のノード id，そのノードに含まれるクエリの id，および経路を表す数字列が保存されている．図 4 において，ノード 15 にはクエリが含まれていないため，L_N を作成する際，ノード 15 は存在していないものとして扱う．

3.2　ソーシャルスコアリスト

次に，各 PoI p_i に対応するソーシャルスコアリスト L^i_{ss} の作成について説明する．これは，L_N に保存された各ノードに対して，ノード id およびそのノードに含まれるクエリ

ノードid	ソーシャルスコア
1	1
2	0.67
13	0.5
14	1
16	1
4	0.8

(a) L^0_{ss}

ノードid	ソーシャルスコア
1	0.4
14	0.33
16	1
4	1

(b) L^3_{ss}

図 5　ソーシャルスコアリストの例
Fig. 5　Example of a social score list.

のソーシャルスコアの中の最大値を保存したものである．具体的には，ノードに含まれているすべてのクエリに対して，各 PoI とのソーシャルスコアを計算する．そして，計算したソーシャルスコアの中で最大のものを，そのノードのソーシャルスコア ss_n として L^i_{ss} に保存する．また，ソーシャルスコアは PoI ごとに異なるため PoI ごとに L_{ss} を作成する．しかし，各 L_{ss} にすべての ss_n を保持させる場合，管理コストが膨大になってしまうため，ソーシャルスコアが 0 となるノードのノード id およびソーシャルスコアは保存しない．

図 2 および図 4 をもとに p_0 および p_3 に対応する L^i_{ss} を作成すると，**図 5** のようになる．たとえば，ノード 1 にはクエリ q_0 および q_1 が含まれており，p_0 のそれぞれのクエリに対するソーシャルスコアは 1 および 0.5 であるため，L^0_{ss} のノード 1 のソーシャルスコアは 1 となる．また，p_3 の q_2 および q_3 に対するソーシャルスコアは 0 であるため，L^3_{ss} はノード 2 およびノード 13 のノード id およびソーシャルスコアを保存しない．

3.3　Top-k データの更新アルゴリズム

最後に，データが生成されたとき，クエリが管理している上位 k 個のデータを更新するアルゴリズムを説明する．

T_{ss} の作成． 新しくデータが生成されると，L_N，データを生成した PoI の L_{ss}，および四分木を用いて，クエリが管理している上位 k 個のデータが更新される可能性があるクエリのみにアクセスするためのチェックを行う．まず，チェックするノードのソーシャルスコアを得るために用いるソーシャルスコアテーブル T_{ss} を作成する．T_{ss} はデータを生成した PoI の L_{ss} と L_N を組み合わせて作成し，L_N に保存されたすべてのノードに対して，ノード id，ノードのソーシャルスコア ss_n，および経路を表す数字列を保存する．あるノードの ss_n を得る際，データを生成した PoI の L_{ss} にそのノードが含まれている場合は L_{ss} に保存されている ss_n を，含まれていない場合は 0 を，そのノードの ss_n とする．同時に，T_{ss} に保存される最大のソーシャルスコアを計算し，根ノードの ss_n とする．

ノードのチェック． T_{ss} が作成され，根ノードの ss_n が計

Algorithm 1: NodeCheck$(n, ss_n, o, first, last, d, T_{ss})$

Input: n：チェック中のノード，ss_n：n のソーシャルスコア，o：データ，$first \cdot last$：チェックに用いる T_{ss} の範囲，d：チェック中のノードの深さ，T_{ss}

```
1: Calculate threshold
2: if threshold ≤ ss_n then
3:   if first = last //node is a leaf-node then
4:     for ∀q ∈ n do
5:       Update(q, o, T_ss, first)
6:   else
7:     for ∀n_c ∈ [NW, NE, SW, SE] do
8:       count_{n_c} ← 0
9:     for i = first to last do
10:      if d-th digit of T_ss[i].ds = 0 then
11:        x ← NW
12:      else if d-th digit of T_ss[i].ds = 1 then
13:        x ← NE
14:      else if d-th digit of T_ss[i].ds = 2 then
15:        x ← SW
16:      else
17:        x ← SE
18:      count_x ← count_x + 1
19:      if ss_x < T_ss[i].ss_n then
20:        ss_x ← T_ss[i].ss_n
21:    d ← d + 1
22:    for ∀n_c ∈ [NW, NE, SW, SE]
       //NW, NE, SW, SE の順に処理を行う do
23:      if count_{n_c} > 0 then
24:        last ← first + count_{n_c} − 1
25:        NodeCheck(n_c, ss_{n_c}, o, first, last, d, T_ss)
26:        first ← first + count_{n_c}
```

算されると，NodeCheck（Algorithm 1）を用いて，根ノードから順に，クエリにアクセスすべきかどうかを判断していく．アクセスすべきかどうかの判断は，ノードに保存された情報とデータの位置情報およびキーワード集合から計算された，生成されたデータが上位 k 個となり得るためのソーシャルスコアの閾値と ss_n との比較により行われる．ss_n が閾値よりも小さい場合，チェック中のノードに含まれるクエリに対して新しく生成されたデータが上位 k 個のデータとなる可能性がないため，クエリにアクセスしない．以下では，あるノードにアクセスした際の具体的な処理について説明する．

ノードにアクセスすると，まず，閾値を計算する（1 行）．あるデータ o に対するノード n の閾値 threshold は式 (5) で計算される．

$$threshold = score_{min} - dist(n, o) - key(n, o) \qquad (5)$$

ここで，$score_{min}$ は，ノード n に含まれるクエリのうち $q.score_k$ が最小のものである．$dist(n, o)$ は，データの位置とノード n の間の最小距離を $MAXloc$ で割ったものを 1 から引いた値である．ただし，データの位置がノード n の範囲内に含まれる場合は $dist(n, o) = 1$ である．$key(n, o)$ は，$key(q_u, o)$ を計算する場合と異なり，式 (6) によって計算される．

$$key(n, o) = \frac{2|n.key \cap o.key|}{|n.key \cap o.key| + |o.key|} \qquad (6)$$

ここで，$n.key$ はノード n が保持しているキーワード集合である．つまり，$key(n, o)$ は，n が保存しているキーワードのうち，データが持つキーワードのみを n が保持していると仮定して，そのキーワードのみで一致度を計算する．

上記により計算される $dist(n, o)$ および $key(n, o)$ は，チェック中のノードに含まれるすべてのクエリ q_u における $dist(q_u, o)$ および $key(q_u, o)$ がとりうる上界値となるため，あるクエリ $q_{u'}$ の $dist(q_{u'}, o)$ および $key(q_{u'}, o)$ が $dist(n, o)$ および $key(n, o)$ より大きくなることはない．また，Top-k データの更新が起こるためには，クエリの k 番目のデータのスコアよりも，新しく発生したデータのスコアが大きくなければならない．ここで，以下の定理が成り立つ．

定理 1. $threshold > ss_n$ ならば，チェック中のノードに含まれるクエリにおいて，*Top-k* データの更新は起こりえない．

証明. チェック中のノードに含まれるすべてのクエリ q_u に対して，Top-k データの更新が起きるためには $q_u.score_k \leq s(q_u, o)$ とならなければならない．しかし，$threshold > ss_n$ ならば，$score_{min} > dist(n, o) + key(n, o) + ss_n$ である．また，$q_u.score_k \geq score_{min}$，および $dist(n, o) + key(n, o) + ss_n \geq dist(q_u, o) + key(q_u, o) + socio(q_u, p) = s(q_u, o)$ であるため，$threshold > ss_n$ ならば，$q_u.score_k > s(q_u, o)$ である．以上により，定理 1 が証明される．　□

つまり，$threshold > ss_n$ であるとき，チェック中のノードに含まれるクエリにおいて，Top-k データの更新は起こりえないため，正確性を担保したまま，チェック中のノードを根とする部分木を枝刈りすることができる．

たとえば，図 4 において，$q_6.score_k = 1.8$ および $q_7.score_k = 2.2$ であるとき，図 2 における p_0 がデータ o を生成したとし，$o.key = \{w_1, w_2, w_5\}$ とする．また，図 4 におけるノード 4 をチェックしたとする．ノード 4 が保存している $n.key = \{w_0, w_1, w_2, w_3, w_4\}$ と $o.key = \{w_1, w_2, w_5\}$ との共通部分は w_1 および w_2 である．このとき，ノード 4 が w_1 および w_2 のみを保持していると仮定して，それらのキーワードとデータ o の持つキーワードの一致度からノード 4 における $key(\cdot, o)$ を計算する．この場合，$key(\cdot, o) = \frac{2\times2}{2+3} = 0.8$ となる．また，PoI p_0 とノード 4 の距離のスコアが $dist(n, o) = 0.6$ であるとする．このとき，新しく生成されたデータに対するノード 4 の閾値は，$\min\{1.8, 2.2\} - 0.6 - 0.8 = 0.4$ となる．

閾値が計算されると，その閾値とチェック中のノードのソーシャルスコアが比較される（2 行）．閾値の方が大きい場合は，現在チェックしているノードに含まれるクエリにおいて，新しく生成されたデータが上位 k 個のデータと

なる可能性がないため，チェック中のノードを根とする部分木のチェックを終了し，チェックが終了していないノードのチェックに移る．閾値の方が小さければ，子ノードをチェックしていく．まず，T_{ss} に含まれる葉ノードであり，チェック中のノード n を根とする部分木に含まれる葉ノードが，n の NW，NE，SW，および SE のどのノードの経路上に存在するか計算する．ある葉ノードが n' の経路上に存在するとは，n からその葉ノードにアクセスする際に，n' が中継ノードであることを意味している．この計算は数字列を用いて行い，子ノードの深さ（$depth$）の桁を参照することで計算できる．同時に，n の子ノード（NW，NE，SW，および SE）のソーシャルスコアを計算する（7–18 行）．それらのうち，T_{ss} に含まれる葉ノードである，もしくは子孫に T_{ss} に含まれる葉ノードを 1 つ以上含むノードに対して，NW から順に同様のチェックを行っていく（20–24 行）．チェックしているノードが葉ノードの場合，閾値の方が小さければ，新しいデータがクエリの上位 k 個のデータとなる可能性がある．そのため，ノードに含まれるすべてのクエリに対して，Top-k データの更新，および四分木に保存されている情報の更新を行うアルゴリズム Update を実行する（3–5 行）．このアルゴリズムの詳細は，後に説明する．

先ほどの例の場合，ノード 4 は葉ノードであり，図 5 よりノード 4 は L^0_{ss} に含まれるため，計算された閾値（0.4）とノード 4 のソーシャルスコア（0.8）を比較すると，閾値の方が小さい．そのため，ノード 4 に含まれるすべてのクエリに対して Top-k データの更新を行う．

Top-k データの更新．葉ノードにおいて Top-k データの更新が必要であると判断されると，Update を用いて，そのノードに含まれるすべてのクエリに対して，Top-k データの更新，および四分木に保存されている情報の更新を行う．

まず，データのスコアを計算し，クエリの k 番目のデータのスコアと比較する．新たに生成されたデータのスコアの方が大きければ，生成されたデータと k 番目のデータを置き換える．提案アルゴリズムでは，Top-k データの置き換えが実行されると，四分木に保存されている情報を更新する必要がある．置き換えが実行されたクエリを含むノードは，T_{ss} に保存された現在の葉ノードの数字列から計算できる．計算で得られたノードに対して Top-k データの置き換えが実行されたクエリの k 番目のデータのスコアの情報を更新する．

4. 評価実験

本章では，ベースラインおよび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．

4.1 セッティング

本実験は，Windows 7，3.47 GHz Intel Xeon CPU，および 192 GB RAM を搭載した計算機で行い，すべてのアルゴリズムは C++で実装した．

表 1　データセット
Table 1　Dataset statistics.

	Yelp	Brightkite
ユーザの数	366,715	50,687
PoI の数	60,785	772,631
全チェックインの数	1,521,160	1,072,965

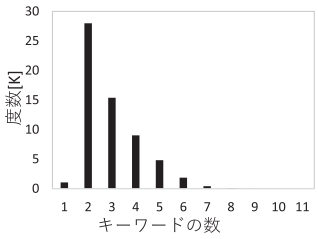


図 6　キーワードの数のヒストグラム
Fig. 6　Histogram of the number of keywords.

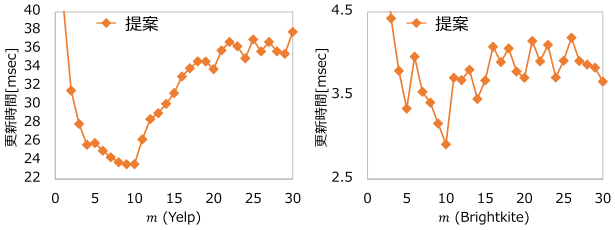
表 2　パラメータの設定
Table 2　Configuration of parameters.

パラメータ	値（デフォルト）
各ノードに含まれるクエリの数の最大値，m	1〜30
$q_u.k$ の最大値，k_{max}	5〜30（10）
キーワードの数の最大値，$\|q_u.key\|_{max}$	1〜11（11）
クエリの数，$\|Q\|$　$[K]$	50〜350（50）（Yelp） 5〜50（10）（Brightkite）

データセット．本実験では，2 つの実データ（Yelp[*6] および Brightkite[*7]）を使用した．**表 1** にデータセットの詳細を示す．それぞれのデータセットにおいて，PoI は位置情報を保持しており，ユーザと PoI のソーシャル関係はチェックイン情報を用いた．それぞれのデータセットはキーワードを含んでいないため，文献 [15] の評価実験と同様に，別途用意した Yelp のレビューのデータセットからキーワードセットを抽出し，そのいずれかを $o.key$ として用いた．このキーワードセットは 1〜11 個のキーワードが含まれている．キーワードの数のヒストグラムを**図 6** に表す．

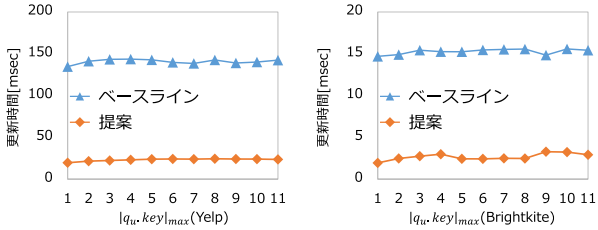
パラメータ．本実験で用いたパラメータを**表 2** に示す．m のデフォルトの値は，各データセットにおいて最初に最適な値を求め，その値を使用した．ユーザは 1 人 1 つのクエリを発行するものとし，$q_u.loc$ は各データセットにおいてすべての PoI が存在する領域内の任意の点とした．$q_u.k$ は 1 から k_{max} の間でランダムな値を選び，$q_u.key$ は Yelp のキーワードセットからキーワードの数が $|q_u.key|_{max}$ 以下であるもののいずれかを用いた．

*6　http://www.yelp.com/dataset_challenge/
*7　http://snap.stanford.edu/data/loc-brightkite.html



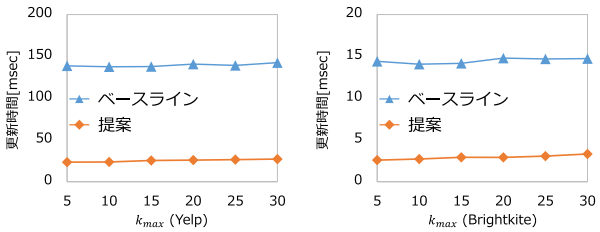
(a) Yelp　(b) Brightkite

図 7　m の影響
Fig. 7　Effect of m.



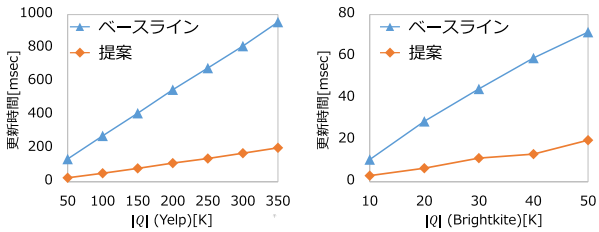
(a) Yelp　(b) Brightkite

図 9　$|q_u.key|_{max}$ の影響
Fig. 9　Effect of $|q_u.key|_{max}$.



(a) Yelp　(b) Brightkite

図 8　k_{max} の影響
Fig. 8　Effect of k_{max}.



(a) Yelp　(b) Brightkite

図 10　$|Q|$ の影響
Fig. 10　Effect of $|Q|$.

4.2　評価結果

本実験では，各アルゴリズムにおける更新時間（1 つのデータの発生に対して Top-k データの更新が完了するまでの平均時間 [msec]）の結果を示す．

m の影響． **図 7** に m を変えたときの結果を示す．m が小さいと，1 つのノードに含まれるクエリの数が少なくなり四分木の高さが高くなる．四分木の高さが高くなると，閾値を計算して比較を行う回数が多くなるため，更新時間が長くなる．m が大きくなると四分木の高さが低くなり更新時間は短くなるが，1 つのノードに含まれるクエリの数が多くなり，ノードで計算される閾値が小さくなる．閾値が小さくなると枝狩りが起きにくくなり，アクセスするクエリの数を削減できないため，m がある値よりも大きくなると再び更新時間が長くなる．以降，Yelp，Brightkite ともに $m = 10$ とした．

k_{max} の影響． **図 8** に k_{max} を変えたときの結果を示す．まず，ベースラインアルゴリズムに比べて提案アルゴリズムの方が高速に Top-k データを更新している．また，ベースラインアルゴリズムは，すべてのクエリに対してスコアを計算するため，k_{max} によらず一定時間で Top-k データを更新している．一方，k_{max} が大きくなると，よりスコアの低いデータが Top-k データとなるため，提案アルゴリズムではノードで計算される閾値が小さくなる．そのため，枝狩りが起きにくくなり，更新時間がわずかに長くなる．

$|q_u.key|_{max}$ の影響． **図 9** に $|q_u.key|_{max}$ を変えたときの結果を示す．提案アルゴリズムでは，クエリのキーワードの数が増加すると，四分木の各ノードに保存されるキーワードの数が増加する．そのため，チェック中のノードにおけるキーワードのスコア $key(n, o)$ を計算する時間がわずかではあるが長くなる．

$|Q|$ の影響． **図 10** に $|Q|$ を変えたときの結果を示す．$|Q|$ を増加させたとき，まだクエリを発行していないユーザが新たにクエリを発行する．ベースラインアルゴリズムに比べて提案アルゴリズムの方が高速に Top-k データを更新している．また，クエリの数が多くなると Top-k データの更新が起こりえないクエリも多くなり，提案アルゴリズムでは，アクセスしないクエリが多くなる．そのため，クエリの数が多くなると，どちらのアルゴリズムでも更新時間が増加するが，提案アルゴリズムはベースラインアルゴリズムよりも増加率が小さい．この結果から，提案したアルゴリズムが効果的であることが分かる．

5.　関連研究

本章では，複数の属性を考慮した検索に関する従来研究，また，Pub/Sub モデルにおけるデータのモニタリングに関する従来研究について紹介する．まず，**表 3** において本研究の位置づけを整理する．

5.1　複数の属性を考慮した検索

位置およびキーワードを考慮した検索． 位置およびキーワードを考慮した検索に関する研究が近年さかんに行われている [17], [19]．これらの文献 [17], [19] では，位置およびキーワードに基づく上位 k 個のデータを効率的に検索するための手法を提案している．たとえば，文献 [17] では，データを四分木により管理し，キーワードの重みの情報を四分木の各ノードに保存している．クエリが発行される

表 3　先行研究との比較
Table 3　The comparison of existing work.

文献	位置	キーワード	ソーシャル関係	Top-k クエリ	データストリーム
[17]	✓	✓		✓	
[2], [21]	✓		✓		
[1], [15]	✓	✓	✓	✓	
[6], [14]	✓	✓			✓
[3]	✓	✓		✓	✓
本研究	✓	✓	✓	✓	✓

と，そのクエリに対して上位 k 個のデータとなりえない部分木を枝刈りする．これにより，アクセスするデータの数を削減し，高速に検索を行うことができる．本研究では，多数クエリが存在している環境において新たにデータが発生したとき，各クエリの Top-k データを高速に更新するため，Top-k データの更新しうるクエリにのみアクセスすることが目的である．そのため，本研究の提案アルゴリズムでは，クエリを四分木により管理することで，新たに発生したデータによりクエリの Top-k データの更新が起こりえない部分木を枝刈りする．

位置およびソーシャル関係を考慮した検索．位置およびソーシャル関係を考慮した検索についても様々な研究が行われている [2], [21]．文献 [2] では，位置およびソーシャル関係を考慮した検索に対して，位置情報に関する処理とソーシャル関係に関する処理を分離することにより，データの管理およびアルゴリズムのデザインに柔軟に対応できるフレームワークを提案している．文献 [21] は，位置およびソーシャル関係を考慮したユーザのグループを検索するクエリを提案している．具体的には，ソーシャルネットワークと複数のクエリポイントが与えられたとき，各ユーザに関連付けられた範囲（一定時間内に移動可能な範囲など）がすべてのクエリポイントを包含するユーザのグループで，そのユーザのグループにより構成されるソーシャルネットワークが k コアであるような最小のグループを検索するクエリ（GSKCG クエリ）を提案している．

位置，キーワード，およびソーシャル関係を考慮した検索．位置，キーワード，およびソーシャル関係を考慮した検索についての研究も行われている [1], [15]．文献 [15] では，上記の 3 つの属性に基づく Top-k 検索を効率的に行うインデックスとして，SNIR 木を提案している．SNIR 木の各ノードには，自身を根とする部分木に含まれるデータの位置情報とデータに興味のあるユーザを保存し，クエリが発行されると Top-k データになりうる部分を順に探索していく．しかし，本研究では，ソーシャル関係の定義が異なり，クエリを SNIR 木で管理した場合，各ノードでソーシャルスコアを計算できない．さらに，このデータ構造は Top-k データのモニタリングに対応しておらず，解の更新を行うためにはすべてのデータを再検索する必要がある．同様に，文献 [1] では，データモニタリングではなくスナップショットクエリを考えており，本研究の問題とは異なる．

5.2　Pub/Sub モデルにおけるデータモニタリング

これまでに様々な研究が Pub/Sub モデルおけるデータモニタリング問題に取り組んでいる．文献 [10], [18] はキーワードがマッチするデータをモニタリングするクエリの効率的な処理手法を，文献 [8], [12] はキーワードの一致度に基づいて計算されたクエリの解をモニタリングするための手法を提案している．しかし，これらの手法は位置情報を考慮していない．文献 [6], [14] では，位置およびキーワードを考慮した Pub/Sub におけるデータモニタリング手法を提案している．たとえば，文献 [14] では，位置情報およびキーワードを考慮し，位置情報がクエリの指定する範囲に含まれ，キーワードがマッチするデータをモニタリングするクエリの効率的な処理手法として，AP 木という新たなデータ構造を提案している．AP 木は，コストモデルに基づき，位置情報およびキーワードによるノードの分割を効果的に行う木構造である．しかし，これらの文献は Top-k 検索を考えていない．また，文献 [3] では，Pub/Sub 環境における Top-k データモニタリング手法を提案している．この文献では，位置，キーワード，および時間に基づいて計算された Top-k クエリの解をモニタリングするためのアルゴリズムを提案している．具体的には，四分木の各セルに，新たに発生したデータの位置がそのセルの範囲内に含まれる場合，解の更新が起こりうるクエリの位置のスコアの上界値を保存している．また，各クエリの発生時間，キーワード集合など位置に依存しない情報は，四分木とは別に保存している．データが発生すると，キーワードのスコアの上界値を計算し，保存された情報から得られるキーワードのスコアの閾値と比較する．そして，キーワードのスコアの上界値が閾値より大きいクエリのみ解の更新を行う．しかし，本研究では，Top-k データを計算する際に時間ではなくソーシャル関係を用いる点，およびデータを生成する PoI ごとにソーシャル関係が変化し，各 PoI ごとに情報を保存する必要がある点から，このアルゴリズムを適用できない．

6. 終わりに

PoI（Point of Interest）からのパブリッシュ/サブスクライブモデルに基づいたデータの発信，および位置情報サービスやソーシャルネットワークサービスの普及により，位置やキーワード，ソーシャル関係を用いた検索への関心が高まっている．本論文では，パブリッシュ/サブスクライブモデルで生成されたデータの中から，位置，キーワード，およびソーシャル関係に基づき，ユーザにとって有用な上位 k 個のデータをモニタリングする問題に取り組んだ．提案アルゴリズムでは，クエリを四分木により管理し，発生したデータが上位 k 個となりうるクエリにのみアクセスすることにより，リアルタイムモニタリングを実現している．評価実験の結果から，提案アルゴリズムの有効性を確認した．

本論文では，データの発生にのみ着目したが，データの削除が起きるアプリケーションも存在する．そのため，データの削除および 2.2 節で述べたソーシャル関係の更新への対応が今後の課題としてあげられる．

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）（JP26240013）および JST 国際科学技術共同研究推進事業（戦略的国際共同研究プログラム）の研究助成によるものである．ここに記して謝意を表す．

参考文献

[1] Ahuja, R., Armenatzoglou, N., Papadias, D. and Fakas, G.J.: Geo-Social Keyword Search, *Proc. Int'l Symposium on Spatial and Temporal Databases*, pp.431–450 (2015).
[2] Armenatzoglou, N., Papadopoulos, S. and Papadias, D.: A general framework for geo-social query processing, *Proc. VLDB Endowment*, Vol.6, No.10, pp.913–924 (2013).
[3] Chen, L., Cong, G., Cao, X. and Tan, K.-L.: Temporal spatial-keyword top-k publish/subscribe, *Proc. IEEE Int'l Conf. on Data Engineering*, pp.255–266 (2015).
[4] Groh, G. and Ehmig, C.: Recommendations in taste related domains: Collaborative filtering vs. social filtering, *Proc. ACM Int'l Conf. on Supporting Group Work*, pp.127–136 (2007).
[5] He, Z. and Lo, E.: Answering why-not questions on top-k queries, *IEEE Trans. Knowledge and Data Engineering*, Vol.26, No.6, pp.1300–1315 (2014).
[6] Hu, H., Liu, Y., Li, G., Feng, J. and Tan, K.-L.: A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions, *Proc. IEEE Int'l Conf. on Data Engineering*, pp.711–722 (2015).
[7] Li, Z., Lee, K.C., Zheng, B., Lee, W.-C., Lee, D. and Wang, X.: Ir-tree: An efficient index for geographic document search, *IEEE Trans. Knowledge and Data Engineering*, Vol.23, No.4, pp.585–599 (2011).
[8] Mouratidis, K. and Pang, H.: Efficient evaluation of continuous text search queries, *IEEE Trans. Knowledge and Data Engineering*, Vol.23, No.10, pp.1469–1482 (2011).
[9] Nanongkai, D., Sarma, A.D., Lall, A., Lipton, R.J. and Xu, J.: Regret-minimizing representative databases, *Proc. VLDB Endowment*, Vol.3, No.1-2, pp.1114–1124 (2010).
[10] Sadoghi, M. and Jacobsen, H.-A.: Be-tree: An index structure to efficiently match boolean expressions over high-dimensional discrete space, *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pp.637–648 (2011).
[11] Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J.X. and Weikum, G.: Efficient top-k querying over social-tagging networks, *Proc. ACM SIGIR Int'l Conf. on Research and Development in Information Retrieval*, pp.523–530 (2008).
[12] Shraer, A., Gurevich, M., Fontoura, M. and Josifovski, V.: Top-k publish-subscribe for social annotation of news, *Proc. VLDB Endowment*, Vol.6, No.6, pp.385–396 (2013).
[13] Tsatsanifos, G. and Vlachou, A.: On Processing Top-k Spatio-Textual Preference Queries, *Proc. Int'l Conf. on Extending Databese Technology*, pp.433–444 (2015).
[14] Wang, X., Zhang, Y., Zhang, W., Lin, X. and Wang, W.: Ap-tree: Efficiently support continuous spatial-keyword queries over stream, *Proc. IEEE Int'l Conf. on Data Engineering*, pp.1107–1118 (2015).
[15] Wu, D., Li, Y., Choi, B. and Xu, J.: Social-aware top-k spatial keyword search, *Proc. IEEE Int'l Conf. on Mobile Data Management*, pp.235–244 (2014).
[16] Yiu, M.L. and Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data, *Proc. Int'l Conf. on Very Large Data Bases*, pp.483–494 (2007).
[17] Zhang, C., Zhang, Y., Zhang, W. and Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search, *Proc. IEEE Int'l Conf. on Data Engineering*, pp.901–912 (2013).
[18] Zhang, D., Chan, C.-Y. and Tan, K.-L.: An efficient publish/subscribe index for e-commerce databases, *Proc. VLDB Endowment*, Vol.7, No.8, pp.613–624 (2014).
[19] Zhang, D., Tan, K.-L. and Tung, A.K.: Scalable top-k spatial keyword search, *Proc. Int'l Conf. on Extending Database Technology*, pp.359–370 (2013).
[20] Zheng, K., Su, H., Zheng, B., Shang, S., Xu, J., Liu, J. and Zhou, X.: Interactive top-k spatial keyword queries, *Proc. IEEE Int'l Conf. on Data Engineering*, pp.423–434 (2015).
[21] Zhu, Q., Hu, H., Xu, J. and Lee, W.-C.: Geo-social group queries with minimum acquaintance constraint, *Computing Research Repository, abs/1406.7367* (2014).

西尾 俊哉　（学生会員）

2016 年大阪大学工学部電子情報工学科卒業後，同大学大学院情報科学研究科博士前期課程在学中．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．

天方 大地　（正会員）

2012 年大阪大学工学部電子情報工学科卒業．2014 年同大学大学院情報科学研究科博士前期課程修了．2015 年同大学院情報科学研究科博士後期課程修了後，同年同大学院情報科学研究科マルチメディア工学専攻助教となり，現在に至る．情報科学博士．データベース，ネットワーク環境におけるデータ検索技術に関する研究に従事．IEEE，ACM，日本データベース学会各会員．

原 隆浩　（正会員）

1995 年大阪大学工学部情報システム工学科卒業．1997 年同大学大学院工学研究科博士前期課程修了．同年同大学院工学研究科博士後期課程中退後，同大学院工学研究科助手，2002 年同大学院情報科学研究科助手，2004 年同大学院情報科学研究科准教授．2015 年より同大学院情報科学研究科教授となり，現在に至る．工学博士．1996 年本学会山下記念研究賞受賞．2000 年電気通信普及財団テレコムシステム技術賞受賞．2003 年本学会研究開発奨励賞受賞．2008 年，2009 年本学会論文賞，2015 年日本学術振興会賞受賞．モバイルコンピューティング，ネットワーク環境におけるデータ管理技術に関する研究に従事．IEEE，ACM，電子情報通信学会，日本データベース学会各会員．

（担当編集委員　榊 剛史）