The University of Osaka
Institutional Knowledge Archive

Title	位置依存型Pub/Subシステムにおける Top-k検索手法に関する研究
Author(s)	西尾, 俊哉
Citation	大阪大学, 2020, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/76655
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

位置依存型Pub/Subシステムにおける Top-k検索手法に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2020年1月

西 尾　俊 哉

関連発表論文

1. 学会論文誌発表論文

1. 西尾俊哉, 天方大地, 原隆浩 : 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, 情報処理学会論文誌 データベース (TOD), Vol. 10, No. 2, pp. 8–18 (2017).

2. 西尾俊哉, 天方大地, 原隆浩 : ユーザの移動を考慮したキーワード付き空間データに対する Top-k 問合せのためのストリーミングアルゴリズム, 電子情報通信学会和文論文誌 D, Vol. J102-D, No. 1, pp. 1–12 (2019).

3. 加藤慎也, 天方大地, 西尾俊哉, 原隆浩 : ストリーミング時系列データに対するモチーフモニタリング, 情報処理学会論文誌, Vol. 60 No. 7, pp. 1260–1269 (2019).

4. 鶴岡翔平, 西尾俊哉, 天方大地, 原隆浩 : Pub/Sub 環境における kNN データモニタリングの分散処理のためのクエリ割り当てアルゴリズム, 情報処理学会論文誌 データベース (TOD), Vol. 12, No. 4, pp. 1–13 (2019).

5. 加藤慎也, 天方大地, 西尾俊哉, 原隆浩 : ストリーミング時系列データに対するディスコードモニタリング, 情報処理学会論文誌, Vol. 61 No. 2 (2020) (採録済).

2. 国際会議発表論文

1. Nishio, S., Amagata, D., and Hara, T.: Geo-Social Keyword Top-k Data Monitoring over Sliding Window, in *Proc. 28th Int'l Conf. on Database and Expert Systems Applications (DEXA 2017), Part I*, LNCS 10438, pp. 409–424, Springer (2017).

2. Kato, S., Amagata, D., Nishio, S., and Hara, T.: Monitoring Range Motif on Streaming Time-Series, in *Proc. 29th Int'l Conf. on Database and Expert Systems Applications (DEXA 2018), Part I*, LNCS 11029, pp. 251–266, Springer (2018).

3. Kato, S., Amagata, D., Nishio, S., and Hara, T.: Discord Monitoring for Streaming Time-Series, in *Proc. 30th Int'l Conf. on Database and Expert Systems Applications (DEXA 2019), Part I*, LNCS 11706, pp. 79–94, Springer (2019).

3. 国内会議発表論文（査読付）

1. 加藤慎也, 天方大地, 西尾俊哉, 原隆浩：ストリーミング時系列データの効率的なモチーフモニタリングアルゴリズム, 情報処理学会 マルチメディア、分散、協調とモバイル (DICOMO 2018) シンポジウム論文集, pp. 1473–1480 (2018).

2. 西尾俊哉, 天方大地, 原隆浩：位置およびキーワードに基づく近似逆 k 最近傍検索, 情報処理学会 第 27 回マルチメディア通信と分散処理ワークショップ (DPSWS 2019) 論文集, pp. 118–124 (2019).

4. その他の研究会等発表論文

1. 西尾俊哉, 天方大地, 原隆浩, 西尾章治郎：位置，キーワードおよびソーシャル関係に基づく Top-k パブリッシュ/サブスクライブ, 第 8 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2016) 論文集, D3-3 (2016).

2. 西尾俊哉, 天方大地, 原隆浩：スライディングウィンドウ上での位置・キーワード・ソーシャル関係に基づく Top-k パブリッシュ/サブスクライブ, 第 9 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2017) 論文集, E4-5 (2017).

3. 西尾俊哉, 天方大地, 原隆浩：位置・キーワードに基づくムービング Top-k パブリッシュ/サブスクライブ, 第 10 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2018) 論文集, G7-2 (2018).

4. 西尾俊哉, 天方大地, 原隆浩：ユーザの移動を考慮した位置・キーワードに基づく Top-k データモニタリング, 第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2019) 論文集, D5-5 (2019).

5. 加藤慎也, 天方大地, 西尾俊哉, 原隆浩：ストリーミング時系列データの効率的なディスコードモニタリングアルゴリズム, 第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2019) 論文集, D5-4 (2019).

6. 鶴岡翔平, 西尾俊哉, 天方大地, 原隆浩：位置・キーワードに基づく kNN データモニタリングの分散処理のためのクエリ割り当てアルゴリズム, 第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2019) 論文集, D4-2 (2019).

以上

内容梗概

インターネット上では様々なデータが生成されており，近年では，飲食店が発信する広告など，位置情報が付加されたデータが大量に生成されている．これらの大量に生成されるデータ群の中から有用なデータをユーザへ配信するには，ユーザが事前に登録したキーワードなどの条件（サブスクリプションと呼ぶ）に合致するデータを配信するパブリッシュ・サブスクライブモデルが有用であり，このモデルを位置情報付きのデータに応用した位置依存型 Pub/Sub システムが注目を集めている．また，データが大量に生成される環境では，ユーザに対して有用な上位 k 個のデータ（Top-k データと呼ぶ）を検索する Top-k 検索を行い，その結果のみを配信することが実用的である．位置依存型 Pub/Sub システムでは，データが頻繁に生成および削除され，各サブスクリプション（ユーザ）に対する Top-k データは時々刻々と変化するため，システムにおいて継続的に Top-k 検索を行い，最新の Top-k データをユーザへ配信することが有用である．本論文では，位置依存型 Pub/Sub システムにおいて，情報受信側と情報配信側それぞれに対して，新たな付加価値を付けたサービスを提供するための Top-k 検索手法について論じる．

情報受信側のユーザに対して，ユーザ個々のニーズに合わせて Top-k データを計算することで，情報受信側は自身の要求により合致したデータを受信可能になる．これを実現するためには，ソーシャル関係に基づくユーザの潜在的な興味を考慮したり，ユーザの移動に伴うユーザの現在地の変化を考慮した上で Top-k データを計算する必要がある．そこで，本論文ではまず，ソーシャル関係を考慮した上での継続的な Top-k 検索，およびユーザの移動を考慮した上での継続的な Top-k 検索を効率的に行う手法をそれぞれ提案する．

一方，情報提供側に対して，あるデータを Top-k データに含むユーザを検索する逆 Top-k 検索を行えるサービスを提供することで，情報提供側は自身のデータに興味をもつ潜在顧客を把握することが可能になる．潜在顧客を把握し，それらの統計的な特性を分析することで，情報提供側は，ユーザのニーズにより合わせたデータの発信や新たな顧客の獲得などが可能になる．そこで，本論文では，逆 Top-k 検索を効率的に行う手法を提案する．

本論文は，5 章から構成され，各章の内容は次の通りである．まず，第 1 章では，序論として研究の背景と目的について述べる．

第 2 章では，位置，キーワード，およびソーシャル関係に基づく継続的な Top-k 検索手法を提案する．提案手法は，データの生成および削除に対して，各サブスクリプションの Top-k データを高速に更新できる．提案手法の性能評価のために行ったシミュレーション実験の結果を示すことにより，提案手法の有効性を示す．

第 3 章では，ユーザの移動を考慮した位置およびキーワードに基づく継続的な Top-k 検索手法を提案する．提案手法は，ユーザの移動，データの生成および削除に対して，各サブスクリプションの Top-k データを高速に更新できる．提案手法の性能評価のために行ったシミュレーション実験の結果を示すことにより，提案手法の有効性を示す．

第 4 章では，位置およびキーワードに基づく逆 Top-k 検索手法を提案する．提案手法は，逆 Top-k 検索の解に含まれないユーザに対する処理を途中で中断することで，高速に解を計算できる．また，複数の逆 Top-k 検索を同時に行うとき，重複する処理を削減し高速に検索を行うため，バッチ処理およびマルチコア環境でバッチ処理を行う手法を提案する．提案手法の性能評価のために行ったシミュレーション実験の結果を示すことにより，提案手法の有効性を示す．

最後に，第 5 章では，本論文の成果を要約し，今後の研究課題について述べ，本論文のまとめとする．

目 次

第 1 章　序論 **1**

1.1　研究背景 1

1.1.1　パブリッシュ・サブスクライブモデル 2

1.1.2　位置依存型 Pub/Sub システム 3

1.1.3　Top-k 検索の利用 4

1.2　研究動機 7

1.2.1　情報受信側の要求 7

1.2.2　情報提供側の要求 9

1.3　本論文の構成 10

第 2 章　位置・キーワード・ソーシャル関係に基づく継続的な Top-k 検索手法 **13**

2.1　まえがき 13

2.2　問題定義 16

2.3　関連研究 19

2.3.1　複数の属性を考慮した検索 20

2.3.2　Pub/Sub システムにおけるデータモニタリング 21

2.3.3　スライディングウィンドウ上での Top-k データモニタリング 22

2.4　提案手法 23

2.4.1　k-スカイバンド 25

2.4.2　サブスクリプションのインデックス 26

2.4.3　アルゴリズム 31

2.5　評価実験 36

2.5.1 評価環境 37
2.5.2 評価結果 40
2.6 むすび 42

第 3 章 ユーザの移動を考慮した位置・キーワードに基づく継続的な Top-k 検索手法 45
3.1 まえがき 45
3.2 問題定義 47
3.3 提案手法 50
3.3.1 フレームワーク 50
3.3.2 Safe Region 52
3.3.3 データの生成に対する処理 56
3.3.4 データの削除に対する処理 68
3.3.5 議論 72
3.4 評価実験 73
3.4.1 評価環境 73
3.4.2 準備実験 76
3.4.3 評価結果 76
3.5 関連研究 83
3.6 むすび 83

第 4 章 位置・キーワードに基づく逆 Top-k 検索手法 87
4.1 まえがき 87
4.2 問題定義 88
4.3 ART クエリ処理手法 91
4.3.1 ベースラインアルゴリズム 91
4.3.2 PART 93
4.4 バッチ処理手法 99
4.4.1 B-PART 99
4.4.2 PB-PART 102

4.5 評価実験 104
4.5.1 評価環境 105
4.5.2 評価結果 106
4.6 関連研究 112
4.6.1 逆 Top-k 検索 112
4.6.2 近似逆最近傍検索 113
4.7 むすび 113

第 5 章 結論 115

5.1 本論文のまとめ 115
5.2 今後の展望 118

謝辞 119

第1章　序論

1.1　研究背景

情報通信技術（ICT）の発展により，パソコンやスマートフォンなどのモバイル端末を用いて，誰もが場所や時間を問わずに簡単にネットワーク上にある多様なコンテンツにアクセス可能になった．それに伴い，ICT を活用した情報配信が盛んに行われ，インターネット上では様々なデータが大量に生成されている．特に，企業や商品に関する宣伝や広告，ニュース記事など人々の生活に役立つ情報を含むデータが大量に生成されており，Google などの検索エンジンや Facebook などの SNS（Social Networking Service）などを用いて，それらの中から自身にとって有用な情報を含むデータを得ることは，人々が日常生活を送るうえで必要不可欠になっている．例えば，外出先で訪れる飲食店を探す際，飲食店検索サイトによる検索だけでなく，SNS におけるユーザの投稿やアプリによるクーポンなどの配布も，行き先を決定する上で重要な要素になっている．情報配信側は，ユーザにとって有用な情報を含むデータを発信したり，閲覧履歴やユーザの属性情報を活用して嗜好性の高い情報や商品を推薦することにより，売り上げの向上や市場規模の拡大を図っている．

情報配信や情報推薦の分野において，それぞれのユーザが必要とする情報を含むデータをリアルタイムに配信することは非常に重要な課題である．近年では，企業だけでなく，ブログや SNS を用いて一般のユーザも多くの情報を発信しており，多種多様なデータが生成されている．それに伴い，有用な情報を得たいユーザの数も爆発的に増加している．例えば，2019 年には，世界中で月間約 24 億人の人々が Facebook を利用しており[1]，膨大な数の人々が，情報の発信および収集を行っ

[1] https://investor.fb.com/investor-events/default.aspx

ている．それぞれのユーザの検索要求も多様化しているため，データの発信者がすべての要求を把握し，要求を満たすデータを配信することは困難である．そのため，各ユーザの要求を満たすデータを効率的に配信できる情報配信システムが必要である．加えて，データは次々と生成されており，各ユーザにとって有用な情報を含むデータは時々刻々と変化する．そのため，システムは，生成されたデータをそれらに興味のあるユーザへリアルタイムに配信できる必要がある．

大量に生成されるデータ群の中から有用な情報を含むデータをユーザへリアルタイムに配信するための方法の1つとして，ユーザが事前に登録したキーワードなどの条件を満たすデータのみを配信するパブリッシュ・サブスクライブ（Pub/Sub）モデルが挙げられる．以下，パブリッシュ・サブスクライブモデルと，それを位置情報付きのデータの配信に応用した位置依存型 Pub/Sub システムについて述べる．

1.1.1　パブリッシュ・サブスクライブモデル

パブリッシュ・サブスクライブ（Pub/Sub）モデルは，データの発信者（情報提供側）が特定の受信者（情報受信側）を指定せずにデータを発信する非同期型のメッセージングモデルである．このモデルでは，情報提供側から情報受信側へのデータの配信を Pub/Sub システムと呼ばれるシステムが仲介する．情報受信側は，自身の興味のあるトピックやキーワードなどの条件（以降，サブスクリプションと呼ぶ）をシステムに登録しておく．情報提供側が，ニュース記事や広告などのデータを特定の受信者を指定することなく発信すると，システムは，それらに興味のある受信者へデータを配信する．具体的には，登録されているサブスクリプション群の中から，発信されたデータのトピックやキーワードと合致するサブスクリプションを特定するフィルタリングという処理を行う．その後，得られたサブスクリプションを登録したそれぞれの受信者へデータを配信する．

パブリッシュ・サブスクライブモデルでは，情報提供側および情報受信側の双方に利点が存在する．情報提供側は，特定の受信者を指定する必要がないため，ユーザ個々の要求を把握することなくデータを発信できる．情報受信側は，登録したサブスクリプションに合致するデータが生成されたとき，システムがそのデータ

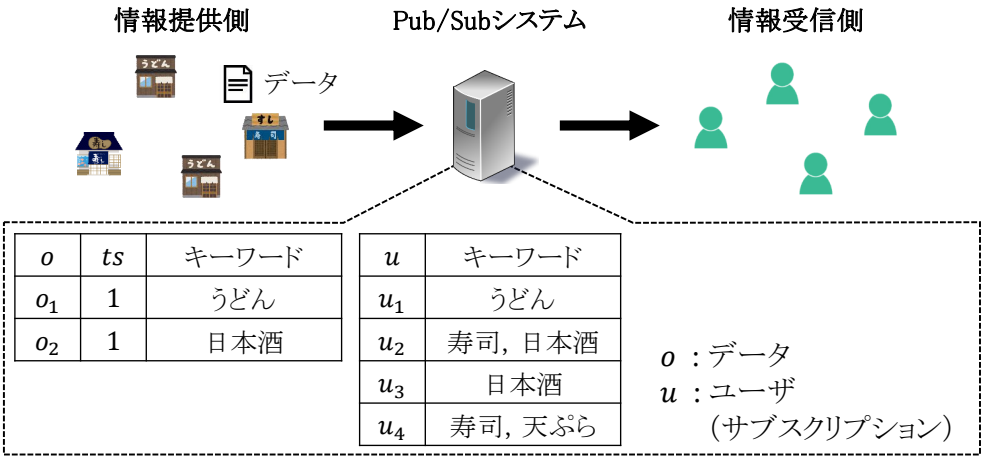


図 1.1: パブリッシュ・サブスクライブモデルの例

をリアルタイムに配信するため，能動的に検索を行う必要はなく，自身の興味のあるデータを受信できる．この際，システムに多数のサブスクリプションが登録されている場合，リアルタイムにデータを配信するためには，システムはフィルタリングを効率的に行う必要がある．

例 1.1. 図 1.1 は，パブリッシュ・サブスクライブモデルの例を示している．情報受信側のユーザは，自身の興味のあるキーワードをサブスクリプションとして登録している．例えば，ユーザ u_1 は “うどん” というキーワードを登録している．そして，ある企業が “うどん” というキーワードに関するデータ o_1 を発信すると，システムがフィルタリングを行い，登録されたサブスクリプションに合致する u_1 に o_1 が配信される．同様に，o_2 は u_2 および u_3 に配信される．

1.1.2　位置依存型 Pub/Sub システム

近年では，レストランなどが発信する広告や SNS におけるユーザの投稿など，位置情報が付加されたデータが大量に生成されている．これらのデータは，位置に関連したイベントや広告など，ユーザにとって有用な情報を多く含んでいるため，位置に依存した情報推薦や情報配信などの対象として注目を集めている [36, 61].

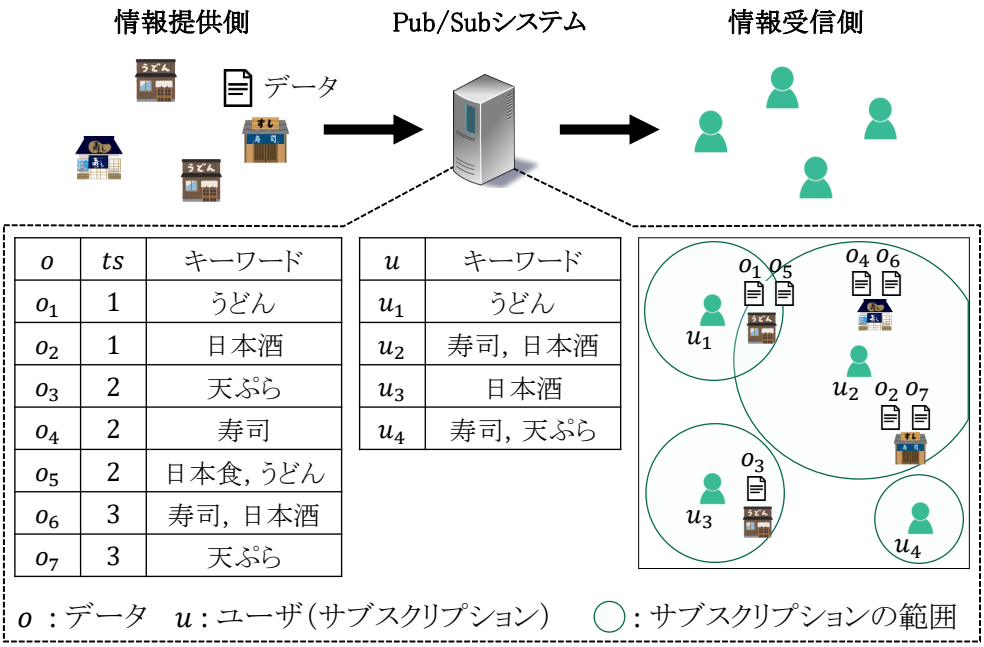
そのため，パブリッシュ・サブスクライブモデルを位置情報付きのデータに応用したシステム（以降，位置依存型 Pub/Sub システムと呼ぶ）に関する研究が盛んに行われている [7, 8, 26, 71, 72, 73]．位置依存型 Pub/Sub システムにおいて，情報受信側のユーザは，興味のあるキーワードに加えて，興味のある位置や範囲をサブスクリプションとしてシステムに登録している．情報提供側の PoI やユーザが，e-クーポンや位置情報付きツイートといったデータを発信すると，システムはフィルタリングを行い，それらが合致するサブスクリプションを登録したユーザへデータを配信する．

位置依存型 Pub/Sub システムでは，情報受信側は興味のあるキーワードだけでなく，興味のある位置や範囲もサブスクリプションとして登録できるため，より自身の興味に合うデータを受信することが可能である．一例として，指定した範囲内のレストランが発信するクーポンの中で，自身の興味に合うものを受信するアプリケーションが考えられる．

例 1.2. 図 1.2 は，位置依存型 Pub/Sub システムの例として，PoI（レストラン）が生成するデータ（クーポン）をそれらに興味のあるユーザに配信する例を表している．各ユーザは，サブスクリプションとして興味のあるキーワードや範囲を指定している．右下の図は，各ユーザが指定する範囲および各データを発信する PoI の位置を表している．時刻 $ts = 1$ では，データ o_1 および o_2 が生成される．o_1 は u_1 が指定した範囲内に含まれキーワードが一致しているため，u_1 に配信される．一方，o_1 は u_2 が指定した範囲内に含まれるがキーワードが一致していないため，u_2 には配信されない．同様に，o_2 は u_2 にのみ配信される．

1.1.3　Top-k 検索の利用

上述した位置依存型 Pub/Sub システムに関する既存研究の中で，文献 [7, 26, 73] では，例 1.2 と同様に，データが生成された位置がサブスクリプションの指定する範囲に含まれ，キーワードが一致するデータを配信する Boolean サブスクリプションを対象としている．しかし，サブスクリプションに合致する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



o	ts	キーワード
o_1	1	うどん
o_2	1	日本酒
o_3	2	天ぷら
o_4	2	寿司
o_5	2	日本食, うどん
o_6	3	寿司, 日本酒
o_7	3	天ぷら

u	キーワード
u_1	うどん
u_2	寿司, 日本酒
u_3	日本酒
u_4	寿司, 天ぷら

図 1.2: 位置依存型 Pub/Sub システムの例

ほとんど受信しないという状況が生じる可能性がある．例えば，図 1.2 において，時刻 $ts = 2$ までに，u_2 は o_2，o_4，および o_5 の 3 つのデータを受信するのに対し，u_3 や u_4 はデータを受信できない．

このような環境では，大量のデータ群の中から，ユーザにとって有用な上位 k 個のデータ（以降，Top-k データと呼ぶ）を検索する Top-k 検索が実用的である．位置依存型 Pub/Sub システムにおいて，ユーザは，サブスクリプションとして位置情報およびキーワードを指定する．そのため，これらに基づいて各データのスコアを計算し，データを順位付けすることが可能である．これにより，各サブスクリプション（ユーザ）に対する Top-k データを計算できる．

例 1.3. 図 1.2 において，各ユーザがそれぞれの位置をサブスクリプションとして指定したと仮定する．時刻 $ts = 2$ までに，5 つのデータが生成されており，u_1 の Top-1 データは，位置が最も近くキーワードが最も類似した o_1 である．同様に，u_2，u_3，および u_4 の Top-1 データは，それぞれ o_4，o_2，および o_3 である．

加えて，近年では，レストランなどの PoI だけでなく一般のユーザも多くの情報

を発信しており，データは頻繁に生成および削除される．それに伴い，各ユーザ（サブスクリプション）に対する Top-k データは時々刻々と変化する．そのため，システムは，各サブスクリプションに対する Top-k データが変化した時点で，最新の Top-k データをリアルタイムに配信できる必要がある．

例 1.4. 図 1.2 において，時刻 $ts = 3$ に，新たにデータ o_6 および o_7 が生成される．このとき，u_2 に注目すると，o_6 は o_4 と比べてよりキーワードが一致しているため，u_2 の Top-1 データは o_6 に変化する．同様に，u_4 に注目すると，o_7 は o_3 と比べてより位置が近いため，u_4 の Top-1 データは o_7 に変化する．

このような背景から，文献 [8, 71, 72] では，位置依存型 Pub/Sub システムにおける Top-k 検索に関する研究に取り組んでいる．リアルタイムに最新の Top-k データをユーザへ配信するためには，システムにおいて，データの生成および削除により，各サブスクリプションに対する Top-k データが変化するかどうか把握する必要がある．これを把握するため，データが生成または削除される度に，システムにおいて各サブスクリプションに対して Top-k 検索を実行する単純な方法は，多大な時間がかかり，サブスクリプションが多く存在する環境に対応できない．そこで，これらの研究では，システムにおいて継続的に Top-k 検索を行い，各サブスクリプションに対する Top-k データを常に管理（モニタリング）する手法が提案されている．具体的には，各サブスクリプションに対する Top-k データを効率的にモニタリングするため，データの生成および削除それぞれに対する処理を高速に行う．データの生成に対して，生成されたデータが Top-k データとなるサブスクリプションの Top-k データを更新する必要がある．これを高速に行うため，サブスクリプションを木構造で管理する．そして，生成されたデータが Top-k データとなる可能性のあるサブスクリプションの候補を高速に限定し，それらの Top-k データのみを更新する．データの削除に対して，削除されたデータを Top-k データに含むサブスクリプションの Top-k データを更新する必要がある．Top-k データを高速に更新するため，k-スカイバンドなどを用いて，Top-k データになる可能性のあるデータのみチェックを行う．

1.2 研究動機

位置依存型 Pub/Sub システムにおいて，情報受信側と情報配信側の双方に対して，新たな付加価値を付けたサービスを提供するため，情報受信側と情報提供側のそれぞれの要求について議論する．

1.2.1 情報受信側の要求

情報受信側であるユーザは，自分自身の興味により合うデータを受信することを要求する．各ユーザの興味により合うデータを配信するためには，各ユーザのニーズに合わせてデータを順位付けし，Top-k データを計算する必要がある．具体的には，事前にサブスクリプションとして指定した位置やキーワードといった条件に加えて，ユーザの潜在的な興味といった明示的に指定できない新たな条件を考慮した上でデータを順位付けしたり，ユーザの現在地のように指定する条件が動的に変化する環境でデータを順位付けする必要がある．以下では，潜在的な興味を考慮した Top-k 検索手法およびユーザの現在地を考慮した Top-k 検索手法について議論する．

潜在的な興味を考慮した Top-k 検索手法

近年，Facebook や Twitter 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への関心が高まっている [1, 76]．ソーシャル関係を考慮することで，検索結果をユーザごとの潜在的な興味に合わせたものにすることが可能になる．具体的には，SNS おけるソーシャルリンクを用いたソーシャルフィルタリング [24] によりユーザの潜在的な興味を抽出できる．この潜在的な興味を考慮した上でデータを順位付けすることで，ユーザの嗜好に合う結果を出力できる．例えば，Facebook における “いいね” など，ユーザは興味のある PoI に対してソーシャル関係（リンク）を持つ．このとき，リンクを持つ PoI 群が類似したユーザ同士は，嗜好が類似している可能性が高い．そのため，リンクを持たない PoI に対しても，ユーザ間のリンクの類似度に基づいて潜在的な興味の度合いを計算で

きる．これらを考慮した上でデータを順位付けすることで，ユーザは自身とって既知でない嗜好に合うデータを受信しやすくなる．

いくつかの研究が，位置およびキーワードに加えて，ソーシャル関係を考慮した上での Top-k データを検索する問題に取り組んでいる [1, 76]．しかし，これらの研究は，スナップショットクエリを対象としており，データの生成および削除を考慮していない．Top-k データをモニタリングするためには，データが生成または削除される度に Top-k 検索を実行する必要があり効率的でない．一方，上述した位置依存型 Pub/Sub システムにおける Top-k 検索に関する既存研究 [8, 71, 72] では，ソーシャル関係を考慮していない．考慮した場合への拡張は自明ではなく，データの生成に対して，Top-k データが変化する可能性のあるサブスクリプションの候補を限定できない．

そのため，ソーシャル関係に基づく潜在的な興味を考慮した上で，システムにおいて，データの生成および削除に対して，各サブスクリプションの Top-k データを効率的にモニタリングできる手法を設計する必要がある．具体的には，データの生成に対して，ソーシャル関係を考慮した上で，サブスクリプションの候補を高速に限定でき，データの削除に対して，新たに Top-k データとなるデータを高速に発見できる手法を設計する必要がある．

ユーザの現在地を考慮した Top-k 検索手法

近年，スマートフォンやタブレットなど GPS を搭載した端末を使用し，ユーザが移動しながらアプリケーションを利用する機会が増加している．そのため，ユーザが事前に指定した位置に近いデータだけではなく，現在地に近いデータを配信することが有用となるアプリケーションも存在する．例えば，観光客が歩きながらレストランを探しているとき，近くのレストランが発信するクーポンを受信するようなアプリケーションである．ユーザの移動を考慮し，現在地に基づいてデータを順位付けし Top-k データを計算することで，ユーザは今現在の状況において有用なデータを受信できる．

いくつかの研究が，ユーザの移動に対して，位置およびキーワードに基づく Top-k データをモニタリングする問題に取り組んでいる [33, 77]．しかし，これらの研究

では，データの生成および削除を考慮しておらず，考慮した場合，効率的に Top-k データをモニタリングできない．一方，位置依存型 Pub/Sub システムにおいても，ユーザの移動，つまりサブスクリプションの指定する位置が変化することを考慮した上で，サブスクリプションに合致するデータをモニタリングする手法が提案されている [26, 74]．しかし，これらの研究では，Boolean サブスクリプションを対象としており，Top-k 検索に対応できない．上述した位置依存型 Pub/Sub システムにおける Top-k 検索に関する既存研究 [8, 71, 72] では，ユーザの移動を考慮していない．考慮した場合，サブスクリプションを単純に木構造で管理できず，データの生成に対して，Top-k データが変化する可能性のあるサブスクリプションの候補を限定できない．

そのため，システムにおいて，ユーザの移動，データの生成および削除に対して，各サブスクリプションの Top-k データを効率的にモニタリングできる手法を設計する必要がある．具体的には，ユーザの移動に対して，Top-k データが変化する場合を把握できる手法，データの生成に対して，指定する位置が動的に変化するサブスクリプションを管理し，生成されたデータが Top-k データになる可能性のあるサブスクリプションの候補を高速に限定できる手法，およびデータの削除に対して，新たに Top-k データとなるデータを高速に発見できる手法をそれぞれ設計する必要がある．

1.2.2 情報提供側の要求

情報提供側は，市場分析やマーケティング調査を行い，自身の生成したデータに興味を持つユーザである潜在顧客を把握することを要求する．潜在顧客の統計的な特性を分析することによって，新たな製品の開発，新たな顧客の獲得，およびアップセリングやクロスセリングを行うことが可能となる [14, 70]．例えば，類似した製品に関するそれぞれの広告などのデータに対する潜在顧客の特性を比較することで，新製品の設計方針を決定したり，新生児の衣類に関する情報を含むデータに対する潜在顧客に対して，ミルクやおむつといった関連する商品を推薦するといったことが可能になると考えられる．

あるデータに対する逆 Top-k 検索の解は，そのデータに対する潜在顧客とみなすことができる．逆 Top-k 検索では，データ集合およびサブスクリプション集合が与えられ，クエリとしてあるデータ（クエリオブジェクト）を指定したとき，クエリオブジェクトを Top-k データに含むサブスクリプション（ユーザ）を検索する．

例 1.5. 図 1.2 において，時刻 $ts = 2$ では，u_1 の Top-1 データは o_1 である．つまり，u_1 は o_1 の潜在顧客である．同様に，o_4，o_2，および o_3 の潜在顧客は，それぞれ u_2，u_3，および u_4 である．

Pub/Sub システムにおいて，逆 Top-k 検索を行い，潜在顧客を高速に把握できるサービスを提供することで，情報提供側は，情報配信や情報推薦を効率化でき，売り上げの向上や市場規模の拡大が期待できる．

1.3　本論文の構成

本論文では，位置依存型 Pub/Sub システムにおける Top-k 検索問題について論じる．本論文は，5 章から構成され，本章以降の内容は以下の通りである．

第 2 章では，位置，キーワード，およびソーシャル関係に基づく継続的な Top-k 検索手法について述べる．まず，ソーシャル関係に基づく潜在的な興味を考慮した Top-k データを計算するため，ユーザの PoI に対する興味の度合いを表すソーシャルスコアを定義し，ソーシャルスコアを加えたスコアを定義する．そして，システムにおいて，データの生成および削除に対して，各サブスクリプションの Top-k データを効率的にモニタリングする手法を提案する．データの削除に対して，新たに Top-k データとなるデータを高速に発見するため，提案手法では，各サブスクリプションに対して後に Top-k データになり得るデータの集合である k-スカイバンドを管理する．データの生成に対して，ソーシャルスコアを加えたスコアに基づいて高速にサブスクリプションの候補を限定するため，提案手法では，各サブスクリプションを木構造を用いて管理し，木構造に対応したリストを用いてソーシャルスコアを管理する．提案手法の性能評価のために行ったシミュレーション実験の結果を示し，その有効性について検証する．

第 3 章では，ユーザの移動を考慮した位置およびキーワードに基づく継続的な Top-k 検索手法について述べる．ユーザの移動，データの生成および削除に対して，各サブスクリプションの Top-k データを効率的にモニタリングする新たなシステム Lamps を提案する．ユーザの移動に対して，Top-k データが変化する場合を把握するため，Lamps では，各サブスクリプションに対して Safe Region を計算する．データの生成に対して，ユーザの移動を考慮した上でサブスクリプションを管理し，サブスクリプションの候補を高速に限定するため，Safe Region に基づく新たな木構造 SQ-tree を提案する．データの削除に対して，新たに Top-k データとなるデータを高速に発見するため，Lamps では，Kd 木およびキーワードごとの転置ファイルおよび四分木を組み合わせた新たなインデックス（KIQF）を用いてデータを管理する．Lamps の性能評価のために行ったシミュレーション実験の結果を示し，その有効性について検証する．

第 4 章では，位置およびキーワードに基づく逆 Top-k 検索手法について述べる．まず，近似的な潜在顧客を検索する近似逆 Top-k クエリ（ART クエリ）を新たに定義する．ART クエリの解を求める際，各サブスクリプションに対して，クエリオブジェクトが Top-k データに含まれるかどうかのみ判断できれば良い．この特徴を考慮し，Top-k データに含まれないと判断されたサブスクリプションに対する処理を途中で中断することで高速に解を求める手法を提案する．さらに，複数の ART クエリが与えられたとき，重複する処理を削減し高速に処理を行うため，バッチ処理およびマルチコア環境でバッチ処理を行う手法を提案する．提案手法の性能評価のために行ったシミュレーション実験の結果を示し，その有効性について検証する．

最後に，第 5 章では，本論文の成果を要約し，今後の展望について述べ，本論文のまとめとする．

第 2 章は文献 [53, 54, 55, 56] で公表した結果に，第 3 章は文献 [57, 58, 59] で公表した結果に，第 4 章は文献 [60] で公表した結果に基づき論述する．

第2章　位置・キーワード・ソーシャル関係に基づく継続的なTop-k検索手法

2.1　まえがき

近年では，飲食店や観光スポットなどが発信する広告やSNSにおけるユーザの投稿など，位置情報が付加されたデータが大量に生成されている．これらのデータは，位置に関連したイベントや広告など，ユーザにとって有用な情報を多く含んでいるため，位置に依存した情報推薦や情報配信などの対象として注目を集めている [36, 61]．これら大量に生成されるデータ群の中から有用な情報を含むデータをユーザへリアルタイムに配信する方法の1つとして，1.1.2節で述べた位置依存型Pub/Subシステムが挙げられる．このとき，大量にデータが存在する環境では，大量のデータ群の中から，ユーザにとって有用な上位k個のデータ（以降，Top-kデータと呼ぶ）を検索するTop-k検索が実用的である．

このような背景から，文献 [8, 71, 72] では，位置依存型Pub/SubシステムにおけるTop-k検索に関する研究に取り組んでいる．Pub/Subシステムにおいて，データは頻繁に生成および削除されるため，各ユーザ（サブスクリプション）に対するTop-kデータは時々刻々と変化する．最新のTop-kデータを配信するためには，システムにおいて，データの生成および削除により，各サブスクリプションに対するTop-kデータが変化するかどうか把握する必要がある．これを把握するため，データが生成または削除される度に，システムにおいて各サブスクリプションに対してTop-k検索を実行する単純な方法は，多大な時間がかかり，サブスクリプションが多く存在する環境に適さない．そこで，これらの文献では，システムに

おいて継続的に Top-k 検索を行う手法が提案されている．具体的には，データの生成または削除により，Top-k データが変化するサブスクリプションを高速に限定する．そして，限定されたサブスクリプションの Top-k データのみを更新することで，各サブスクリプションに対する最新の Top-k データを常に管理（モニタリング）する．

位置依存型 Pub/Sub システムにおいて，情報受信側であるユーザは，自分自身の興味により合うデータを受信することを要求する．各ユーザの興味により合うデータを配信するために有効な方法の 1 つとして，事前にサブスクリプションとして指定した位置やキーワードといった明示的に指定された条件に加えて，ユーザの潜在的な興味といった明示的に指定できない新たな条件を考慮した上で Top-k データを計算し，その結果を配信することが挙げられる．ユーザの潜在的な興味を考慮した上でデータを順位付けし，Top-k データを特定することで，ユーザの嗜好に合う結果を出力できる．

ユーザの潜在的な興味を抽出するため，本章では，SNS におけるソーシャル関係に注目する．文献 [24] で提案されているソーシャルフィルタリングなどを用いることにより，ソーシャル関係からユーザの潜在的な興味を抽出できる．ソーシャル関係には，Twitter におけるフォロー関係などのユーザ間の関係に加え，Facebook における “いいね” などユーザと PoI 間の関係も含まれる．本章では，ユーザと PoI 間の関係に注目する．例えば，Facebook における “いいね” では，ユーザは興味のある PoI に対してソーシャル関係（リンク）を持つ．このとき，リンクを持つ PoI 群が類似したユーザ同士は，嗜好が類似している可能性が高い．そのため，リンクを持たない PoI に対しても，ユーザ間のリンクの類似度に基づいて潜在的な興味の度合いを計算できる．計算された興味の度合いを考慮した上で Top-k データを検索することで，ユーザは自身とって既知でない嗜好に合うデータを受信しやすくなる．

例 2.1. 図 2.1 は，ユーザと PoI とのソーシャル関係の例を示している．ソーシャル関係（リンク）はユーザと PoI 間の辺として表現でき，図 2.1 はユーザ u_1 および u_2 のリンクを表現している．ユーザ u_1 は PoI p_1 に対してリンクを持つため，p_1 に興味を持っている．一方，u_2 は p_1 に対してリンクを持たないが，u_1 と u_2 は同

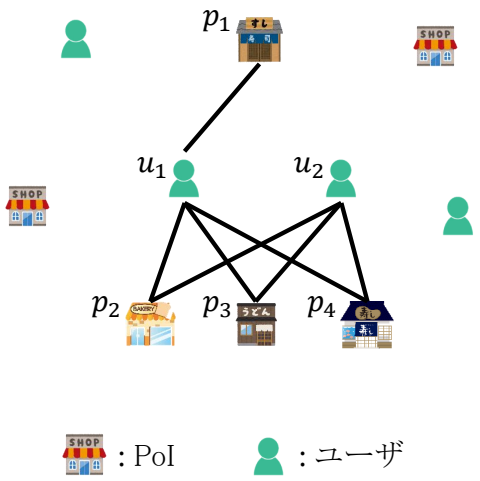


図 2.1: ユーザと PoI とのソーシャル関係の例

じ PoI p_2，p_3，および p_4 に対してリンクを持っており，二人の嗜好は類似していると考えられる．そのため，u_2 の p_1 に対する潜在的な興味の度合いは大きいと言える．

このような背景から，いくつかの研究が，位置およびキーワードに加えて，ソーシャル関係を考慮した上での Top-k データを検索する問題に取り組んでいる [1, 76]．しかし，これらの研究は，データの生成および削除を考慮しておらず，Top-k データを効率的にモニタリングできない．一方，上述した位置依存型 Pub/Sub システムにおける Top-k 検索に関する既存研究 [8, 71, 72] では，ソーシャル関係を考慮しておらず，考慮した場合への拡張も自明ではない．

本章では，位置依存型 Pub/Sub システムにおける位置，キーワード，およびソーシャル関係に基づく継続的な Top-k 検索手法を提案する．具体的には，ソーシャル関係に基づく潜在的な興味を考慮した上で，システムにおいて，各サブスクリプションに対する Top-k データを効率的にモニタリングする手法を提案する．

以下では，2.2 節で，本章で扱う問題を詳細に定義し，2.3 節で関連研究について述べる．2.4 節で提案手法について述べ，2.5 節で評価実験の結果を示す．最後に 2.6 節で本章のまとめを行う．

2.2　問題定義

本節では，まず，位置およびキーワードに基づくデータと位置，キーワード，およびソーシャル関係に基づく Top-k サブスクリプションを定義する．本章では，既存研究 [71, 72] と同様に，時々刻々と生成されるデータ群の中で，直近に生成されたデータ群のみを Top-k データの対象とするため，カウントベースのスライディングウィンドウを用いる．そのため，カウントベースのスライディングウィンドウを定義し，その後，Top-k データを定義する．そして，ソーシャル関係に基づく潜在的な興味を考慮した上で，各サブスクリプションに対する Top-k データを計算するため，潜在的な興味の度合いを表すスコア（ソーシャルスコア）を考慮したスコアを定義する．最後に，本章で扱う問題を詳細に定義する．

データを生成する PoI の集合を P，システムにサブスクリプションを登録するユーザの集合を U とする．まず，位置およびキーワードに基づくデータと位置，キーワード，およびソーシャル関係に基づく Top-k サブスクリプションを定義する．以降，GSK（**G**eo-**S**ocial **K**eyword Top-k）サブスクリプションと呼ぶ．

定義 2.1 (データ)**.** ある PoI $p \in P$ が生成するデータは，$o = (p, l, \mathcal{W}, ts)$ と定義される．ここで，$o.p$ はデータ o を生成した PoI，$o.l$ は緯度と経度によって表される 2 次元平面上の PoI p の位置，$o.\mathcal{W}$ はデータ o が持つキーワードの集合，$o.ts$ はデータ o の生成時刻である．

定義 2.2 (GSK サブスクリプション)**.** あるユーザ $u \in U$ の発行する GSK サブスクリプションは，$s = (u, l, \mathcal{W}, k)$ と定義される．ここで，$s.u$ はサブスクリプション s を発行したユーザ，$s.l$ はサブスクリプション s の位置，$s.\mathcal{W}$ はサブスクリプション s が持つキーワードの集合，および，$s.k$ はユーザが要求するデータの個数である．

今後，特に明記する必要がない場合，$s.k$ を k と表記する．

次に，カウントベースのスライディングおよびあるサブスクリプションに対する Top-k データを定義する．

定義 2.3 (スライディングウィンドウ)**.** 大きさが $|SW|$ のカウントベースのスライディングウィンドウ SW には，直近に生成された $|SW|$ 個のデータが含まれる．

カウントベースのスライディングウィンドウでは，ある PoI が新たにデータを生成すると，ウィンドウがスライドし，最も古いデータが SW から取り除かれ，新たに生成されたデータが SW に挿入される．

定義 2.4 (Top-k データ)**.** スライディングウィンドウ SW が与えられたとき，あるサブスクリプション s の Top-k データ T は次の条件を満たす．(i) $T \subseteq SW$，(ii) $|T| = s.k$，(iii) $\forall o \in T, \forall o' \in SW \backslash T, score(s, o) \leq score(s, o')$．ここで，$score(s, o)$ は s に対するデータ o のスコアである．

次に，あるサブスクリプション s に対するあるデータ o のスコア $score(s, o)$ を，文献 [1] で提案されている定義に基づき，以下のように定義する．スコアは小さいほど優れているものとする．

定義 2.5 ($score(s, o)$)**.** あるサブスクリプション s に対するあるデータ o のスコア $score(s, o)$ は，以下の式で与えられる．

$$score(s, o) = dist(s.l, o.l) + text(s.\mathcal{W}, o.\mathcal{W}) + socio(s.u, o.p)^1 \tag{2.1}$$

ここで，$dist(s.l, o.l)$ は $s.l$ と $o.l$ との空間距離を表すスコア（距離スコア），$text(s.\mathcal{W}, o.\mathcal{W})$ は $s.\mathcal{W}$ と $o.\mathcal{W}$ とのキーワードの類似度を表すスコア（キーワードスコア），$socio(s.u, o.p)$ は $s.u$ の $o.p$ に対する興味の度合いを表すスコア（ソーシャルスコア）である[2]．

距離スコアは，文献 [11, 19, 26, 33, 48, 71, 74] で用いられている式と同様に，ユークリッド距離を用いて計算され，$dist(s.l, o.l) = \frac{Edist(s.l, o.l)}{Maxdist}$ である．ここで，$Edist(s.l, o.l)$ は，$s.l$ と $o.l$ とのユークリッド距離，$Maxdist$ はサブスクリプションと PoI の存在し得る領域 $\mathbb{R}^2$ 内の任意の 2 点間の最大距離である．

[1] 文献 [29, 83, 52] などで，Top-k 検索において，ユーザが各属性の重み付けを行うことは困難であり，自身の望む結果を得ることが困難であるという問題が指摘されている．そのため，本章では重みづけを考慮する必要のない線形和を用いてスコアを計算する．なお，各属性に重みづけをした場合においても，本章の提案手法は軽微な修正で対応可能である．

[2] 本章では，$dist(\cdot, \cdot)$，$text(\cdot, \cdot)$，および $socio(\cdot, \cdot)$ として，$[0, 1]$ に正規化できるものを用いるため，$score(s, o)$ は $[0, 3]$ である．

キーワードスコアは，Dice 類似度を用いて計算され，$text(s.\mathcal{W}, o.\mathcal{W}) = 1 - \frac{2|s.\mathcal{W} \cap o.\mathcal{W}|}{|s.\mathcal{W}|+|o.\mathcal{W}|}$ である[3]．

ソーシャルスコアは，位置，キーワード，およびソーシャル関係に基づいて Top-k データを検索する既存研究 [1, 76] で用いられている定義とは異なり，ユーザと PoI 間のソーシャル関係に基づいて計算される．本章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（リンク）を持つ．この関係は，ソーシャルグラフを用いて図 2.2 のように表現できる．三角がユーザ，四角が PoI を表し，ソーシャルリンクはグラフにおける辺として表現できる．このとき，リンクを持つ PoI 群が類似したユーザ同士は，嗜好が類似している可能性が高い．そこで，あるユーザ u がある PoI p に対してリンクを持たない場合，文献 [66] で用いられているリンクの類似度の定義に基づき，ソーシャルスコアを計算する．具体的には，p に対してリンクを持つユーザの中で，リンクを持つ PoI 群が最も類似したユーザとのリンクの類似度に基づいて，ソーシャルスコアを計算する．ソーシャルグラフの辺の集合を SE，ユーザ u と PoI p 間の辺を $e_{u,p}$ としたとき，ソーシャルスコア $socio(u, p)$ を，以下の式で定義する[4][5]．

$$socio(u,p) = \begin{cases} 0 & (e_{u,p} \in SE) \\ 1 - \max_{u' \in U_p} \frac{2|P_u \cap P_{u'}|}{|P_u|+|P_{u'}|} & (e_{u,p} \notin SE) \end{cases} \tag{2.2}$$

ここで，P_u は u がリンクを持つ PoI の集合，つまり，$P_u = \{\forall p' \in P | \exists e_{u,p'} \in SE\}$ である．また，U_p は p に対してリンクを持つユーザの集合，つまり $U_p = \{\forall u' \in U | \exists e_{u',p} \in SE\}$ である．

例 2.2. 例えば，図 2.2 において，$e_{u_1,p_1} \in SE$ であるため，$socio(u_1, p_1) = 0$ である．次に，$socio(u_6, p_5)$ について考える．ここで，$e_{u_6,p_5} \notin SE$ である．p_5 に対し

[3]本章の提案手法は，Jaccard 類似度や Cosine 類似度といった集合同士の類似度を計算するために用いられる他の指標にも容易に拡張可能である．

[4]実環境では，ソーシャルリンクの更新（辺の追加および削除）が発生することが考えられる．この更新が発生するとソーシャルスコアが変化する可能性がある．しかし，本章ではソーシャルリンクの更新は考えないものとし，更新への対応は今後の課題とする．

[5]ソーシャルスコアの定義として SimRank[34] や Personalized PageRank（PPR）[21, 35] といったものも考えられる．提案手法は，スコアが $[0, 1]$ に正規化できるものであれば，任意の定義を用いることが可能である．

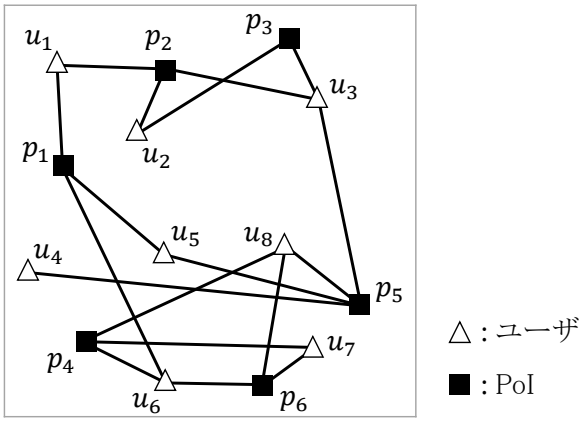


図 2.2: ソーシャルグラフの例

てリンクと持つユーザは u_3，u_4，u_5，および u_8 であるため，それぞれのユーザとのリンクの類似度を計算する．u_6 がリンクを持つ PoI は p_1，p_4 および p_6 であり，u_5 がリンクを持つ PoI は p_1 および p_5 であるため，u_6 と u_5 とのリンクの類似度は，$\frac{2\times1}{3+2} = 0.4$ となる．一方，u_3，u_4，および u_8 との類似度はそれぞれ 0，0，および 0.67 である．よって，$socio(u_6, p_5) = 1 - \max\{0, 0, 0.4, 0.67\} = 0.33$ となる．

問題定義． GSK サブスクリプションの集合 S およびスライディングウィンドウ SW を考える．本問題は，システムにおいて，S に含まれるすべてのサブスクリプションに対して，継続的に Top-k 検索を行うことである．具体的には，データの生成および削除に対して，S に含まれるすべてのサブスクリプションの最新の Top-k データを常に管理（モニタリング）する．本章は，位置およびキーワードに加えてソーシャル関係を考慮した上で，大量のサブスクリプションの Top-k データをモニタリングする問題に取り組んだ初めての研究である．

2.3 関連研究

本節では，複数の属性を考慮した検索に関する従来研究，および Pub/Sub システムにおけるデータモニタリングに関する従来研究について述べる．また，スラ

イディングウィンドウ上でのデータモニタリングに関する従来研究についても述べる．

2.3.1　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盛んに行われている [29, 52, 83, 86, 88]．これらの文献では，位置およびキーワードに基づく上位 k 個のデータ（Top-k データ）を効率的に検索する手法が提案されている．例えば，文献 [86] 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する．そして，クエリが発行されると，そのクエリに対して Top-k データとなり得ない部分木をフィルタリングする．これにより，チェックするデータの数を削減でき，高速に検索を行える．本章の提案手法では，この文献のアイデアに基づき，サブスクリプションを四分木により管理する．そして，新たに生成されたデータに対して，Top-k データの更新をチェックするチェックするサブスクリプションの数を削減する．

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 [3, 42, 45, 50, 79, 92]．文献 [3] 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が提案されている．また，文献 [42, 45, 50, 79, 92] では，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が提案されている．例えば，文献 [92] 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）が提案されている．しかし，これらの文献は，キーワードを考慮しておらず，ソーシャル関係としてユーザ間のソーシャル関係を用いている．そのため，これらの文献で提案された手法は本章における問題に適用できない．

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, 76]．文献 [76] 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 木が提案されている．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およびそれらのデータに興味のあるユーザの情報を保存する．そして，クエリが発行されると Top-k データとなる可能性の大きいデータを管理するノードから順に検索を行う．しかし，これらの文献はスナップショットクエリを対象としており，Top-k データのモニタリングに対応していない．Top-k データのモニタリングを行うためには，データが生成または削除されるたびに Top-k 検索を実行する必要があり効率的でない．

2.3.2 Pub/Sub システムにおけるデータモニタリング

キーワードに基づくデータモニタリング． これまでに様々な研究が Pub/Sub システムおけるデータモニタリング問題に取り組んでいる．文献 [2, 11, 20, 48, 65, 75, 87] では，各サブスクリプションに対して，サブスクリプションのキーワードを含むデータを効率的にモニタリングする手法，文献 [51, 67] では，サブスクリプションとデータのキーワード集合の類似度に基づいて計算された各サブスクリプションの解をモニタリングする手法が提案されている．しかし，これらの研究では位置情報を考慮していないため，本章における問題とは異なる．

位置およびキーワードに基づくデータモニタリング． 文献 [7, 32, 41, 73] では，Pub/Sub システムにおいて，位置およびキーワードに基づくサブスクリプションの解をモニタリングする手法が提案されている．例えば，文献 [73] では，各サブスクリプションに対して，データの位置がサブスクリプションの指定する範囲に含まれ，サブスクリプションのキーワードを含むデータをモニタリングする手法として，サブスクリプションを効率的に管理する AP-tree という新たなインデックスが提案されている．AP-tree は，木構造を構築する際，管理するサブスクリプションの位置情報およびキーワードに基づいて，位置またはキーワードによるノードの分割を柔軟に行う．しかし，これらの文献は Top-k 検索を考えていない．

一方，文献 [8, 71, 72] では，Pub/Sub システムにおいて，位置およびキーワードに基づく Top-k データをモニタリングする手法が提案されている．文献 [8] 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 データをモニタリングする手法，文献 [71, 72] では，スライディングウィンドウ上で，位置およびキーワードに基づいて計算された Top-k データをモニタリングする手法がそれぞれ提案されている．これらの手法では，四分木を用いてサブスクリプションを管理する．そして，生成されたデータが Top-k データになる可能性のあるサブスクリプションの候補を高速に限定する．限定されたサブスクリプションの Top-k データのみを更新することで，データの生成に対して，各サブスクリプションの Top-k データを効率的にモニタリングする．

しかし，本章における問題では，位置およびキーワードに加えてソーシャル関係に基づいて Top-k データを計算する．ソーシャルスコアは，データの生成元である PoI に依存する．そのため，四分木の各ノードでサブスクリプションの候補を限定するために用いるフィルタリングの条件は，PoI ごとに異なる．その結果，ソーシャル関係を考慮した場合，データの生成に対して，サブスクリプションの候補を限定できない．Top-k データをモニタリングするためには，データを生成するすべての PoI に対して異なる四分木を管理する必要があり，これらの手法を単純に適用できない．

2.3.3　スライディングウィンドウ上での Top-k データモニタリング

これまでにスライディングウィンドウ上での Top-k データモニタリングに関する様々な研究が行われている [49, 63, 71, 72, 82]．ウィンドウがスライドし，古いデータがウィンドウから取り除かれると，そのデータを Top-k データに含むサブスクリプションの Top-k データを更新する必要がある．このとき，ウィンドウ内のすべてのデータに対してスコアを計算し，Top-k データを更新する方法は計算コストが非常に大きい．これらの文献では，この問題を解決する手法が提案されている．

文献 [82] では，kmax というアプローチにより，Top-k データを効率的にモニタ

リングする．具体的には，サブスクリプションに対して，上位 k 個のデータではなく，上位 k' 個のデータを管理する．ここで，k' は k からあるパラメータ $kmax$ までの間のランダムな値である．これにより，Top-k データに含まれるデータが削除されたとき，高速に Top-k データを更新できる．しかし，kmax は Top-k データになる可能性のないデータを管理してしまう可能性があり，効率的でない．

一方，文献 [49] では，k-スカイバンドと呼ばれる，データが削除されたとき，後に Top-k データとなる可能性のあるデータを管理しておくことで，Top-k データを効率的にモニタリングする手法が提案されている．しかし，サブスクリプションが多く存在する環境では，各サブスクリプションに対して正確な k-スカイバンドを管理すると管理コストが大きい．この問題を解決するため，文献 [63, 71, 72] では，k-スカイバンドの一部である，確率的な k-スカイバンドやコストベースの k-スカイバンドを保持する手法が提案されている．

2.4 提案手法

文献 [72, 73] と同様に，サブスクリプションはシステムに事前に登録されていると仮定する．カウントベースのスライディングウィンドウでは，ある PoI が新たにデータを生成すると，ウィンドウ SW がスライドし，最も古いデータが SW から取り除かれ（データの削除），新たに生成されたデータが SW に挿入される（データの生成）．ウィンドウがスライドすると，各サブスクリプションの Top-k データが変化する可能性がある．そのため，各サブスクリプションの Top-k データを効率的にモニタリングするためには，データの削除および生成それぞれに対して，高速に処理を行う必要がある．データの削除に対して，削除されるデータを Top-k データに含むサブスクリプションの Top-k データを更新する必要がある．高速に Top-k データを更新するためには，Top-k データになる可能性のあるデータのみチェックを行うことが望ましい．一方，データの生成に対して，生成されたデータが Top-k データになるサブスクリプションの Top-k データを更新する必要がある．高速に Top-k データを更新するためには，生成されたデータが Top-k データとなる可能性のあるサブスクリプションの候補を限定し，それらの Top-k データ

のみを更新することが望ましい．

そこで，データの削除および生成に対して高速に処理を行うため，提案手法を以下のよう設計する．データの削除に対して高速に処理を行うため，提案手法では，既存研究 [71, 72] と同様に，k-スカイバンド [18] を用いる．k-スカイバンドは，古いデータが削除されたとき，新たに Top-k データとなる可能性のあるデータの集合である．各サブスクリプションに対して k-スカイバンドを管理することで，Top-k データに含まれるデータが削除されたサブスクリプションに対して，新たに Top-k データとなるデータを高速に発見できる．一方，データの生成に対して高速に処理を行うため，生成されたデータが Top-k データになる可能性のあるサブスクリプションの候補を高速に限定する手法を提案する．既存研究 [8, 71, 72] では，位置情報に基づいてサブスクリプションを木構造で管理することで，高速にサブスクリプションの候補を限定する手法が提案されている．これらの文献のアイデアに基づき，ソーシャル関係を考慮した上で，サブスクリプションの候補を高速に限定できる手法を提案する．具体的には，既存手法と同様にサブスクリプションを木構造で管理する．そして，ソーシャル関係を考慮した上で正確にフィルタリングを行うため，ソーシャル関係に基づく情報を新たに管理する．木構造の各ノードでは，ソーシャルスコアを含めたスコアに関する情報を新たに管理する．さらに，各 PoI に対して，木構造に対応したリストを用いてソーシャルスコアを管理する．これらを用いることで，生成されたデータが Top-k データとなる可能性のないサブスクリプションをまとめてフィルタリングできる．

以下では，まず，2.4.1 項において，提案手法で適用する k-スカイバンドを説明する．次に，2.4.2 項において，サブスクリプションを管理するインデックスについて説明する．最後に，2.4.3 項において，ウィンドウのスライドに対して，各サブスクリプションの Top-k データを更新するアルゴリズムについて述べる．表 2.1 に，本章で用いる記号の概要を示す．

表 2.1: 記号の概要

記号	意味
o	データ
SW	スライディングウィンドウ
s	GSK サブスクリプション
S	GSK サブスクリプションの集合
w	キーワード
n	四分木のノード
S_n	n または n を根とする部分木で管理されるサブスクリプションの集合
$\mathcal{W}_n$	$s \in S_n$ に含まれるキーワード集合の和集合
SL_p	PoI p に対するソーシャルスコアリスト
ds	根ノードからの経路を表す数字列

2.4.1 k-スカイバンド

ウィンドウ SW のスライドにより，あるデータ o' が SW から取り除かれたとき，o' を Top-k データに含むサブスクリプションの Top-k データを更新する必要がある．高速に Top-k データを更新するため，位置依存型 Pub/Sub システムにおける Top-k 検索に関する既存研究 [71, 72] では，k-スカイバンド [18] を用いている．提案手法でも，同様に k-スカイバンドを用いる．以下に，支配および k-スカイバンドを定義する．

定義 2.6 (支配)**.** あるサブスクリプション s に対して，データ o および o' が以下の条件を満たす場合，s に対して o は o' に支配されるという．

$$(score(s, o) \geq score(s, o')) \wedge (o.ts < o'.ts)$$

定義 2.7 (k-スカイバンド)**.** スライディングウィンドウ SW が与えられたとき，あるサブスクリプション s に対する k-スカイバンドは，SW に含まれるデータ群の中で，最大 $k-1$ 個のデータに支配されるデータの集合である．

あるサブスクリプションに対する k-スカイバンドとは，古いデータが削除されたとき，Top-k データとなり得るデータの集合である．また，あるサブスクリプションの Top-k データは，k-スカイバンドに含まれるデータのうちスコアの小さい上位 k 個のデータである．以降，あるサブスクリプション s に対する k-スカイバンドを SKY とし，SKY に含まれるデータ o に対して，o を支配するデータの数を $o.dom$ と表す．

新たにデータが生成されると，k-スカイバンドに含まれる各データ o の $o.dom$ が変化する可能性がある．しかし，データが生成されるたびに，すべてのサブスクリプションの k-スカイバンドを正確に更新すると多大な時間がかかる．この問題を解決するため，文献 [71, 72] では，Top-k データに影響を及ぼす可能性の高いデータでのみ k-スカイバンドを更新する手法が提案されている．提案手法では，これらの文献で提案された手法のアイデアに基づき，生成されたデータのスコアが j 番目（$j \geq k$）のスコアより小さくなる可能性のあるサブスクリプションのみ k-スカイバンドの更新を行う．

2.4.2　サブスクリプションのインデックス

新たにデータ o が生成されたとき，各サブスクリプションに対する Top-k データおよび k-スカイバンドを高速に更新するため，o のスコアが j 番目（$j \geq k$）のスコアより小さくなる可能性のあるサブスクリプションの候補を高速に限定する必要がある．これを実現するため，既存研究 [8, 71, 72] で提案されている手法のアイデアに基づき，ソーシャル関係を考慮した上で，高速にサブスクリプションの候補を限定できる手法を提案する．この際，ソーシャルスコアの以下の特徴に注目する．あるサブスクリプションに対するソーシャルスコアは，データの生成元である PoI に依存する．そのため，ある PoI が複数のデータを生成したとき，各データのあるユーザに対するソーシャルスコアはすべて同じ値を取る．

提案手法では，既存手法と同様にサブスクリプションを四分木で管理する．そして，ソーシャル関係を考慮した上で正確にフィルタリングを行うため，ソーシャル関係に基づく情報を新たに管理する．四分木の各ノードでは，ソーシャルスコ

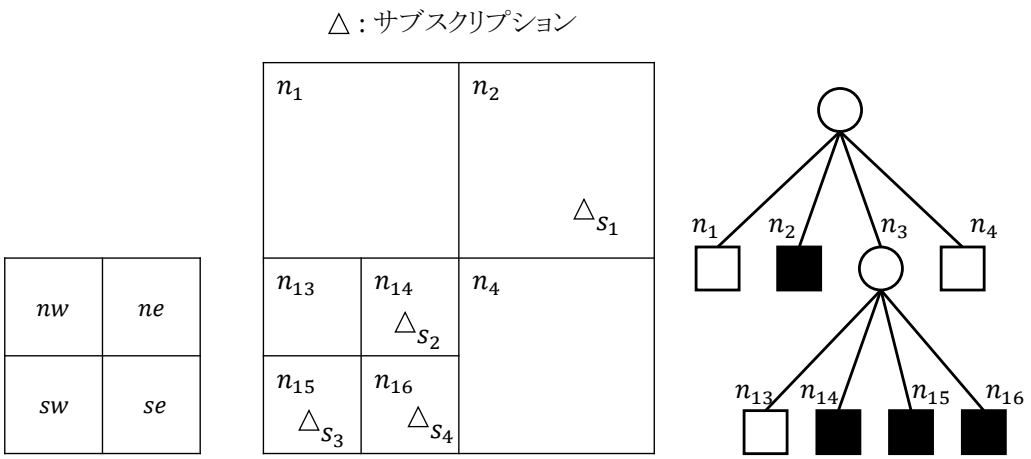


図 2.3: 四分木の表現方法

アを含めたスコアに関する情報を新たに管理する．加えて，各 PoI に対して，ソーシャルスコアを管理するソーシャルスコアリストを作成する．ある PoI に対するソーシャルスコアリストは，四分木の葉ノードごとに，そのノードで管理されるサブスクリプションに対するソーシャルスコアの最小値を管理する．

以下では，提案手法におけるインデックス（四分木およびソーシャルスコアリスト）の詳細を説明し，その後，これらを用いたフィルタリング手法について述べる．

インデックス

四分木． 提案手法では，文献 [86] で用いられている線形四分木のアイデアに基づき，GSK サブスクリプションをそれぞれの位置に基づいて管理する．線形四分木は，各ノードが 4 個までの子ノードを持つ木構造である．四分木のノードの分割を図 2.3 のように表現する．左図に示すように，ノードを分割した際，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．例えば，中央図のように，4 つのサブスクリプションが存在し，領域が分割されたとする．この場合，四分木の構造は，右図のような木構造をとる．本章では，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，サブスクリプションが含まれる葉ノードは黒四角で表現する．また，各ノードの id はモートン順序に基づいて決定する．

ここから，四分木の構築方法を説明する．四分木を構築する際，サブスクリプションがどのノードに含まれるかはサブスクリプションの位置のみに依存する．まず，サブスクリプションおよび PoI が存在し得る領域を四分木の根ノードとする．各ノードで管理できるサブスクリプションの数を m とし，ノードに含まれるサブスクリプションの数が m 個を超える場合は，子ノードを作成し，サブスクリプションを子ノードに分配する．分割されたノード内のサブスクリプションの数が m 個を超えていれば同じ操作を再帰的に行い，各ノードに含まれるサブスクリプションの数が m 個以下になるまで繰り返す．

ソーシャル関係を考慮した上でフィルタリングを行うために，提案手法における四分木の各ノード n では以下の情報を管理する．

- S_n : n が，葉ノードである場合，n で管理するサブスクリプションの集合，中継ノードである場合，n を根とする部分木で管理されるサブスクリプションの集合．
- $\mathcal{W}_n$: S_n に含まれるサブスクリプションのキーワード集合の和集合．
- $jscore_{max}$: S_n に含まれる各サブスクリプション s に対する j 番目のデータのスコア $jscore(s)$ の中で最大の値

さらに，n が葉ノードである場合，S_n に含まれる各サブスクリプション s に対する $jscore(s)$ を管理する．

例 2.3. 図 2.2 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サブスクリプション s を発行し，$m = 2$ であると仮定した場合，図 2.4 の中央に示す四分木が構築される．例えば，中継ノード n_3 は，$S_{n_3} = \{s_4, s_5, s_6\}$ であるため，それらのキーワード集合および $jscore_{max}$ を管理する．同様に，葉ノード n_4 は，$S_{n_4} = \{s_7, s_8\}$ であるため，それらのキーワード集合および $jscore_{max}$ に加えて，$jscore(s_7)$ および $jscore(s_8)$ を管理する．

ソーシャルスコアリスト． 四分木の葉ノードごと把握するため，四分木が構築される際，ノードに含まれるサブスクリプションの数が m 個以下となったノード（葉

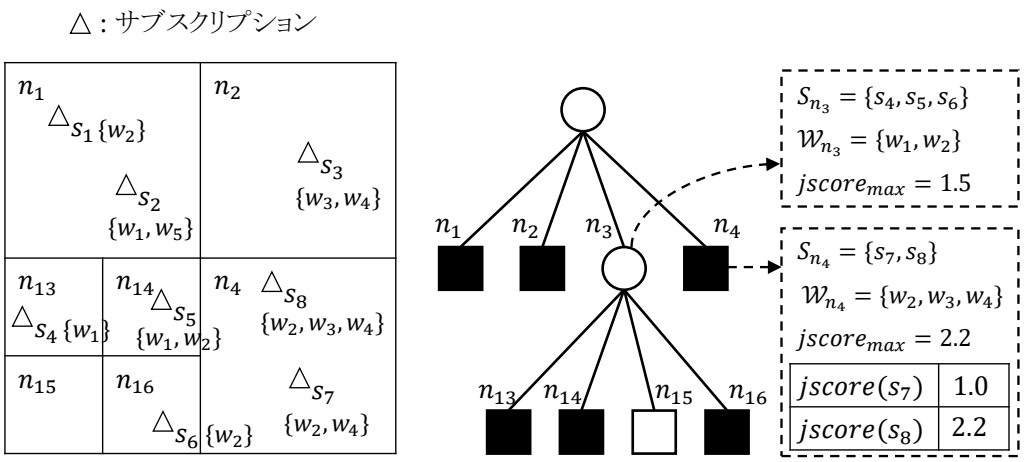


図 2.4: 四分木の構築の例

ノード）に対して，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 NL として管理する．根ノードからそのノードまでの経路を表す数字列は，ノードの id を用いて計算し，その結果を保存する．提案手法における四分木では，モートン順序に基づいてノードの id を決定しているため，自身の id から 1 を引いたものを 4 で除算した商が親ノードの 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．例えば，n_{14} 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，NL には，サブスクリプションが含まれるすべての葉ノード（図 2.4 における黒四角のノード）に対する数字列が管理されている．図 2.5 は，図 2.4 における木構造に対する NL の例を示している．n_{15} はサブスクリプションを管理していないため，NL を作成する際，n_{15} は存在していないものとして扱う．

ここで，ある PoI p に対する四分木のノード n のソーシャルスコア $socio(n, p)$ を以下の式で定義する．

$$socio(n, p) = \min_{s \in S_n} socio(s.u, p) \tag{2.3}$$

各 PoI p に対するソーシャルスコアリスト SL_p は，NL で管理する各ノード（サ

ノード	数字列
n_1	0
n_2	1
n_{13}	02
n_{14}	12
n_{16}	32
n_4	3

図 2.5: NL の例

ノード	$socio(n,p)$
n_1	0
n_2	0
n_{13}	0.5
n_{14}	0.6
n_4	0.67

図 2.6: SL_{p_3} の例

ブスクリプションを管理する葉ノード）に対して，そのノードのソーシャルスコアを管理する．さらに，各 SL_p に対して NL で管理するすべてのノードのソーシャルスコアを管理する場合，管理コストが膨大になってしまうため，$socio(n,p)=1$ となるノードはソーシャルスコアリストに含めない．図 2.6 は，図 2.2 における PoI p_3 に対するソーシャルスコアリスト SL_{p_3} の例を示している．$socio(n_{16},p_3)=1$ であるため，n_{16} は SL_{p_3} に含めない．

フィルタリング手法

ここから，四分木の各ノードで管理されるサブスクリプションをフィルタリングする方法を説明する．ある PoI p がデータ o を生成し，ノード n をチェックしたと仮定する．このとき，S_n に含まれる各サブスクリプション s に対する o のスコアの下界値 $score_{lb}(n,o)$ が以下の式で計算される．

$$score_{lb}(n,o) = dist(n,o.l) + text(\mathcal{W}_n, o.\mathcal{W}) + socio(n,o.p) \tag{2.4}$$

ここで，$dist(n,o.l)$ は n と $o.l$ との距離スコアの最小値，$text(\mathcal{W}_n, o.\mathcal{W})$ は n と o との間のキーワードスコアであり，それぞれ以下の式により計算される．

$$dist(n,o.l) = \begin{cases} 0 & (o.l \in n) \\ dist(l^*, o.l) & (o.l \notin n) \end{cases} \tag{2.5}$$

ここで，$o.l \in n$ は，$o.l$ が n の領域内に含まれることを表し，l^* は，n の境界線上

で $o.l$ に最も近い点である．

$$text(\mathcal{W}_n, o.\mathcal{W}) = 1 - \frac{2|\mathcal{W}_n \cap o.\mathcal{W}|}{|\mathcal{W}_n \cap o.\mathcal{W}| + |o.\mathcal{W}|} \tag{2.6}$$

つまり，$text(\mathcal{W}_n, o.\mathcal{W})$ は，n が管理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 が管理していると仮定した場合のキーワードスコアである．

$dist(n, o.l)$ および $text(\mathcal{W}_n, o.\mathcal{W})$ は，S_n に含まれるすべてのサブスクリプション s に対して，常に $dist(n, o.l) \leq dist(s.l, o.l)$ および $text(\mathcal{W}_n, o.\mathcal{W}) \leq text(s.\mathcal{W}, o.\mathcal{W})$ を満たす．同様に，p に対するソーシャルスコアリスト SL_p から得られる $socio(n, o.p)$ も，$socio(n, o.p) \leq socio(s.u, o.p)$ を満たす．（SL_p から $socio(n, o.p)$ を得る詳細なアルゴリズムは，後述する．）あるサブスクリプションの Top-k データの更新が起こるためには，サブスクリプションの k 番目のデータのスコア $kscore(s)$ よりも，o のスコアが小さくなければならない．そのため，以下の定理が成り立つ．

定理 2.1. $jscore_{max} < score_{lb}(n, o)$ ならば，新たに生成されたデータ o は，S_n に含まれるすべてのサブスクリプションの Top-k データになり得ない．

証明. あるサブスクリプション s に対して，Top-k データの更新が起こるためには $kscore(s) > score(s, o)$ が必要である．しかし，$jscore_{max} < score_{lb}(n, o)$ ならば，$jscore_{max} < dist(n, o.l) + text(\mathcal{W}_n, o.\mathcal{W}) + socio(n, o.p)$ である．また，$kscore(s) \leq jscore(s) \leq jscore_{max}$，および $dist(n, o.l) + text(\mathcal{W}_n, o.\mathcal{W}) + socio(n, o.p) \leq dist(s.l, o.l) + text(s.\mathcal{W}, o.\mathcal{W}) + socio(s.u, o.p) = score(s, o)$ であるため，$\forall s \in S_n, kscore(s) < score(s, o)$ である．よって，定理が成り立つ． □

定理 2.1 より，$jscore_{max} < score_{lb}(n, o)$ ならば，正確性を失うことなく，n を根とする部分木をフィルタリングでき，S_n に含まれるすべてのサブスクリプションの Top-k データおよび k-スカイバンドを更新する必要がない．

2.4.3 アルゴリズム

最後に，データの生成および削除に対して，各サブスクリプションの Top-k データを更新するアルゴリズムについて述べる．以下では，データの生成および削除に対する処理をそれぞれ説明する．

データの生成

新たにデータ o が生成されると，四分木，ノードリスト NL，および o を生成した PoI $o.p$ のソーシャルスコアリスト $SL_{o.p}$ を用いて，o のスコアが $jscore(s)$ より小さくなる可能性があるサブスクリプション，つまり，k-スカイバンドを更新する必要があるサブスクリプションの候補を限定する．

ST の作成． 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，ソーシャルスコアテーブル ST を作成する．ST は，NL と $SL_{o.p}$ を組み合わせて作成し，NL で管理するすべてのノードに対して，ノードの経路を表す数字列およびノードのソーシャルスコアを保存する．あるノード n のソーシャルスコア $socio(n, o.p)$ を得る際，$n \in SL_{o.p}$ ならば，保存されているソーシャルスコア，$n \notin SL_{o.p}$ ならば，$socio(n, o.p) = 1$ とする．同時に，ST に保存される最小のソーシャルスコアを計算し，根ノード n_{root} のソーシャルスコア $socio(n_{root}, o.p)$ とする．

ノードのチェック． ST が作成され，$socio(n_{root}, o.p)$ が計算されると，根ノード n_{root} から順にノードをチェックし，k-スカイバンドを更新すべきサブスクリプションを発見する．アルゴリズム 1 は，あるノード n をチェックするアルゴリズムを示している．

まず，n のソーシャルスコア $socio(n, o.p)$ などに基づいて，$score_{lb}(n, o)$ を計算する（1 行）．定理 2.1 より，$jscore_{max} < score_{lb}(n, o)$ ならば，新たに生成されたデータ o により，チェックを行っているノード n の S_n に含まれるすべてのサブスクリプションの k-スカイバンドを更新する必要がないため，n 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．その際，S_n に含まれる各サブスクリプションに対して，$s.score_f > score_{lb}(n, o)$ ならば $s.score_f$ を更新する（19–22 行）．ここで，$s.score_f$ は s がフィルタリングされたときの $score_{lb}(n, o)$ の最小値を保存したものである．$s.score_f$ の詳細は後述する．$jscore_{max} \geq score_{lb}(n, o)$ であり，n が葉ノードである場合，S_n に含まれる各サブスクリプションに対して，k-スカイバンド，Top-k データ，および四分木に保存されている情報を更新するアルゴリズム UpdateResult を実行する（3–5 行）．（このアルゴリズムの詳細は，後述する．）一方，n が中継ノードである場合，n の子ノー

Algorithm 1: CheckNode($n, o, socio, begin, end, depth$)

```
Input: n, o, socio, begin, end, depth
1  Calculate score_lb(n, o)
2  if jscore_max ≥ score_lb(n, o) then
3      if begin = end //n is a leaf node then
4          for ∀s ∈ S_n do
5              UpdateResult(s, o, ST[begin].ds)
6      else
7          digit ← 0, socio ← ∞, count ← 0
8          for i = begin to end do
9              if depth-th digit of ST[i].ds = digit then
10                 count ← count + 1
11                 if socio > ST[i].socio then
12                     socio ← ST[i].socio
13             else
14                 if count > 0 then
15                     n_c ← n.N_c[digit]
16                     CheckNode(n_c, o, socio, begin, begin + count − 1, depth + 1)
17                 digit ← digit + 1, begin ← begin + count
18                 socio ← ∞, count ← 0
19 else
20     for ∀s ∈ S_n do
21         if s.score_f > score_lb(n, o) then
22             s.score_f ← score_lb(n, o)
```

ドを再帰的にチェックする．このとき，n を根とする部分木に含まれる葉ノードのソーシャルスコアは，ST の $begin$ から end で管理されている．ここで，ST の $begin$ から end で管理される葉ノードは，n の nw を根ノードとする部分木に含まれる葉ノードから順に，ne，sw，および se の順にソートされてる．そのため，n の子ノードの集合を $N_c[\cdot] = [nw, ne, sw, se]$ とし，nw から順に処理を行う．ST 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

ノード	数字列	$socio(n,p)$
n_1	0	0
n_2	1	0
n_{13}	02	0.5
n_{14}	12	0.6
n_{16}	32	1
n_4	3	0.67

図 2.7: ST の例

するかは，ST で管理する数字列の中で子ノードの深さ（depth）の桁を参照することで計算できる（9–12 行）．ここで，ある葉ノードがノード n' の経路上に存在するとは，ノード 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．このとき，同時に，子ノードのソーシャルスコアを計算する．チェックを行う子ノードを根とする部分木に含まれる葉ノードのソーシャルスコアを管理する ST の範囲およびソーシャルスコアが計算されると，そのノードに対して再帰的にノードのチェックを行う（14–18 行）．

例 2.4. 図 2.2 における PoI p_3 がデータ o を生成し，$o.\mathcal{W} = \{w_1, w_3, w_4\}$ であると仮定する．このとき，図 2.7 に示すソーシャルスコアテーブル ST が作成され，$socio(n_{root}, o.p) = 0$ である．そして，ノード n_3 をチェックしたとし，$dist(n_3, o.l) = 0.6$ であると仮定する．$\mathcal{W}_{n_3} = \{w_1, w_2\}$ と $o.\mathcal{W} = \{w_1, w_3, w_4\}$ の共通部分は w_1 であるため，$text(\mathcal{W}_{n_3}, o.\mathcal{W}) = 1 - \frac{2\times1}{1+3} = 0.5$ である．さらに，$socio(n_3, o.p) = 0.5$ であるため，$score_{lb}(n_3, o) = 0.6 + 0.5 + 0.5 = 1.6$ である．つまり，$n_3.jscore_{max} < score_{lb}(n_3, o)$ が成り立つため，n_3 を根とする部分木をフィルタリングできる．同様に，ノード n_4 をチェックしたとし，$dist(n_4, o.l) = 0.3$ であると仮定する．$text(\mathcal{W}_{n_4}, o.\mathcal{W}) = 1 - \frac{2\times2}{2+3} = 0.2$ および $socio(n_4, o.p) = 0.6$ であるため，$score_{lb}(n_4, o) = 0.3 + 0.2 + 0.6 = 1.1$ である．つまり，$n_4.jscore_{max} < score_{lb}(n_4, o)$ は成り立たないため，n_4 に含まれるすべてのサブスクリプションに対して UpdateResult を実行する．

k-スカイバンドおよび Top-k データの更新． 葉ノードにおいて k-スカイバンドの更新が必要であると判断されると，そのノードに含まれる各サブスクリプションに

Algorithm 2: UpdateResult(s, o, ds)

Input: s, o, ds

1 Calculate $score(s, o)$
2 **for** $\forall o' \in s.SKY$ **do**
3 　**if** $score(s, o) \leq score(s, o')$ **then**
4 　　$o'.dom \leftarrow o'.dom + 1$
5 　　**if** $o'.dom \geq s.k$ **then**
6 　　$s.SKY \leftarrow s.SKY - \{o'\}$
7 $s.SKY \leftarrow s.SKY + \{o\}$
8 **if** $score(s, o) \geq kscore(s)$ **then**
9 　Update $s.T$ from $s.SKY$
10 $N \leftarrow$ A set of nodes that are calculated by ds
11 **for** $\forall n \in N$ **do**
12 　Update $n.jscore_{max}$

対して UpdateResult を実行する．アルゴリズム 2 は，k-スカイバンド，Top-k データ，および四分木に保存されている情報を更新するアルゴリズムを示している．まず，データのスコアを計算し，$s.SKY$ に含まれるデータ o' の $o'.dom$ を更新する（1–4 行）．$o'.dom$ が k 以上となったデータは $s.SKY$ から取り除き，生成されたデータを $s.SKY$ に追加する（5–7 行）．次に，$score(s, o)$ がサブスクリプションの k 番目のスコア $kscore(s)$ より小さければ，o と k 番目のデータを置き換える（8–9 行）．k-スカイバンドが更新された場合，四分木に保存された情報を更新する必要がある．更新されたサブスクリプションを含むノードは ST で管理する数字列から計算できる．これにより得られた各ノード n に対して，必要があれば $n.jscore_{max}$ を更新する（10–12 行）．

データの削除

新たに生成されたデータに対する処理が終了すると，SW から取り除かれるデータ o' に対する処理を行う．アルゴリズム 3 は，o' を k-スカイバンドに含むサブスクリプション s の k-スカイバンドおよび Top-k データを更新するアルゴリズムを示

Algorithm 3: ReevaluateTopk(s, o)

Input: s, o'

1 $s.SKY \leftarrow s.SKY - \{o'\}$
2 **if** $o' \in s.T$ **then**
3 　**if** $|s.SKY| \geq s.k$ **then**
4 　　Extract $s.T$ from $s.SKY$
5 　　**if** $s.score_f \leq kscore(s)$ **then**
6 　　Extract $s.SKY, s.T$ from SW
7 　**else**
8 　　Extract $s.SKY, s.T$ from SW

している．まず，o' を k-スカイバンド $s.SKY$ から取り除く（1 行）．そして，Top-k データ $s.A$ に o' が含まれる場合，o' を取り除き，$s.SKY$ に含まれるデータの中で，スコアの良い上位 k 個のデータを新たな Top-k データとする．ここで，新たに生成されるデータ o に対する処理において，o のスコアが j（$j \geq k$）番目のスコア $jscore(s)$ より小さくなるサブスクリプション s のみ，k-スカイバンドおよび Top-k データの更新を行う．そのため，$|s.SKY| < s.k$ またはスコアが $s.score_f$ 以上のデータが Top-k データとなるとき，SW 内のデータにおいて $s.SKY$ に含まれないデータが Top-k データとなり得る．ここで，$s.score_f$ は，自身を含むノードがフィルタリングされたときの $score_{lb}(n, o)$ の最小値をサブスクリプションごとに保存したものである．つまり，スコアが $s.score_f$ 以上のデータが Top-k データとなるとき，過去に k-スカイバンドに追加しなかったデータが Top-k データになり得る．よって，この場合，SW 内のすべてのデータに対してスコアを計算し，k-スカイバンドおよび Top-k データを更新する（5–8 行）．

2.5　評価実験

本節では，提案手法の性能評価のために行った実験の結果を述べる．提案手法の有効性を検証するため，サブスクリプションが指定する k の最大値，サブスクリプションの数，およびウィンドウサイズに対するスケーラビリティを評価した．

本実験は，Windows 10 Pro，3.20GHz Intel Core i7，および 64GB 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．コンパイルには最適化オプションとして-O2 を使用した．既存の Pub/Sub システムに関する研究 [72, 73] と同様に，リアルタイム性を保証するため，メインメモリ上で各インデックスの管理を行った．

2.5.1 評価環境

位置，キーワード，およびソーシャル関係に基づいて，大量のサブスクリプションの Top-k データをモニタリングする問題に取り組んだ初めての研究であるため，本問題に適用できる既存手法は存在しない．そのため，本実験では，ベースラインとなる手法に提案手法のいくつかの機能を適用した手法との比較を行った．ベースラインとなる手法では，データの生成および削除に対して以下のように処理を行う．データが生成されたとき，すべてのサブスクリプションの Top-k データを更新する．データが削除されたとき，SW 内のすべてのデータをチェックし Top-k データを更新する．提案手法の機能を適用したそれぞれの手法の詳細は，以下の通りである．

- **ベースライン 1（b1）．** k-スカイバンドを用いる手法．データが生成されたとき，すべてのサブスクリプションの k-スカイバンド（Top-k データ）を更新する．データが削除されたとき，k-スカイバンドを用いて Top-k データを更新する．

- **ベースライン 2（b2）．** サブスクリプションのインデックス（四分木およびソーシャルスコアリスト）を用いる手法．データが生成されたとき，インデックスを用いてサブスクリプションの候補を限定し，それらの Top-k データのみを更新する．データが削除されたとき，SW 内のすべてのデータをチェックし Top-k データを更新する．

- **ベースライン 3（b3）．** サブスクリプションのインデックス（四分木およびソーシャルスコアリスト）を用いる手法．データが生成されたとき，インデッ

表 2.2: データセットの詳細

データセット	*YELP*	*BRIGHTKITE*
ユーザの数	366,715	50,687
PoI の数	60,785	772,631
チェックインの数	1,521,160	1,072,965

クスを用いてサブスクリプションの候補を限定し，それらの Top-k データのみを更新する．データが削除されたとき，SW 内のすべてのデータをチェックする際に，ソーシャルスコアリスト SL を利用し，Top-k データを更新する．具体的には，サブスクリプション s に対してあるデータ o のソーシャルスコアを計算する際，s を管理するノードが $SL_{o.p}$ に含まれる場合は保存されたソーシャルスコアを，含まれない場合は 1 とする． そして，計算されたスコアと k 番目のデータのスコアを比較し，Top-k データになり得るかどうかを評価する．o が Top-k データになり得る場合のみ正確なソーシャルスコアを計算し，Top-k データを更新する．

評価結果を示すグラフ中では，提案手法を Proposal と表す．

データセット． 本実験では，PoI およびソーシャル関係（リンク）に関するデータセットとして，2 つの実データ（*YELP*[6]および *BRIGHTKITE*[7]）を用いた．表 2.2 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 間のリンクはチェックイン情報を用いた．

データ． 各データセットにおける PoI が複数のデータを生成すると仮定し，データを作成した．各データの位置は，データを生成した PoI の位置とした．また，それぞれのデータセットはキーワードに関する情報を含んでいないため，別に用意した Yelp のレビューに関する実データ（*REVIEW*）を用いた．各レビューには 1〜11 個のキーワードが含まれており，各レビューに含まれるキーワードの数のヒストグラムを図 2.8 に表す．各データは，1 つのレビューをランダムに選び，その

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

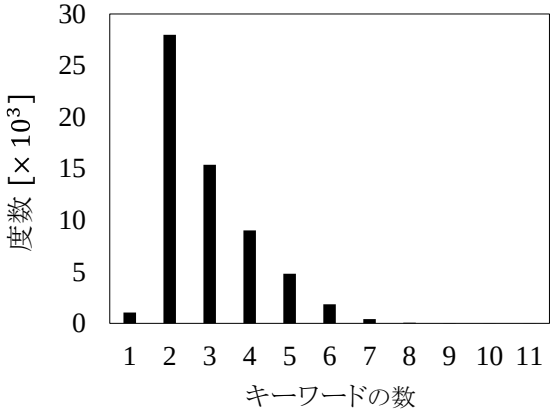


図 2.8: *REVIEW* の 1 つのレビューに含まれるキーワードの数のヒストグラム

表 2.3: パラメータの設定

パラメータ	値
$s.k$ の最大値，k_{max}	5, **10**, 15, 20, 25, 30
サブスクリプションの数，$\|S\|$ [$\times 10^3$]	**50**, 100, 150, 200, 250, 300, 350 （*YELP*） **10**, 20, 30, 40, 50 （*BRIGHTKITE*）
ウィンドウサイズ，$\|SW\|$ [$\times 10^6$]	**0.5**, 1, 1.5, 2, 2.5, 3

レビューに含まれるキーワードをデータのキーワードとする．

サブスクリプション． ユーザは一人ひとつのサブスクリプションを発行するものとし，各データセットに対して，GSK サブスクリプションを作成した．各サブスクリプションは，1 つのレビューに含まれるキーワードの中から 1〜5 個のキーワードをランダムに選び，サブスクリプションのキーワードとする．各データセットにおいてユーザは位置情報を保持していないため，サブスクリプションの位置として，各データセットにおいて PoI が存在する領域内のランダムな点とした．さらに，$s.k$ は 1 から k_{max} の間の一様乱数とした．

パラメータ． 表 2.3 に，本実験で用いたパラメータを示す．太字で表されている値はデフォルトの値である．また，$j = 2k$ をデフォルトの値として用いた．

本実験は，$|SW|$ 個のデータが生成された時点から開始し，新たに 100,000 個の

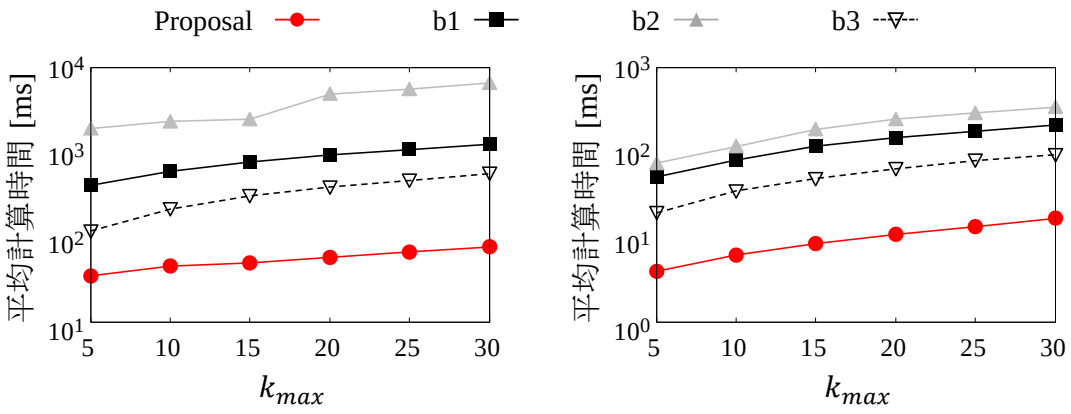
データが生成されるまで行う．そして，各手法における平均計算時間（ウィンドウのスライドに対して，各サブスクリプションの Top-k データの更新が完了するまでの時間）の結果を述べる．

2.5.2　評価結果

k_{max} の影響． 図 2.9 にサブスクリプションの指定する k の最大値 k_{max} を変えたときの結果を示す．提案手法は他の手法と比較して，常に高速に Top-k データを更新できており，計算時間が少なくとも 3 分の 1 に抑えられている．また，k_{max} の増加に伴い，各手法において平均計算時間が増加することがわかる．k_{max} が増加すると，各サブスクリプション s に対する $kscore(s)$ が大きくなるため，新たに生成されたデータが Top-k データになり得るサブスクリプションの数が増加し，計算時間が増加する．ここで，各データセットにおいて，*BRIGHTKITE* を用いた場合のベースライン 2 とベースライン 3 の計算時間の差よりも，*YELP* を用いた場合の差の方が大きいことがわかる．*YELP* はユーザの数に比べて PoI の数が少なく，各ユーザが同じ PoI にチェックインしている可能性が大きい．その結果，ソーシャルスコアの計算により多くの時間が必要であるため，すべてのデータに対して，ソーシャルスコアリストを利用せずにソーシャルスコアを計算するベースライン 2 は大きく性能が低下する．

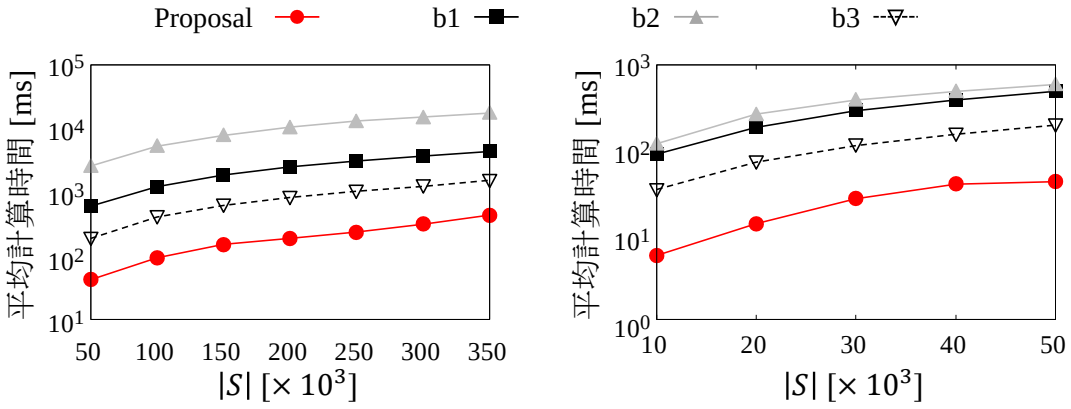
$|S|$ の影響． 図 2.10 にサブスクリプションの数 $|S|$ を変えたときの結果を示す．提案手法は他の手法と比較して，常に高速に Top-k データを更新できることがわかる．また，$|S|$ の増加に伴い，各手法において平均計算時間が増加することがわかる．

$|SW|$ の影響． 図 2.11 にウィンドウサイズ $|SW|$ を変えたときの結果を示す．提案手法は他の手法と比較して，常に高速に Top-k データを更新できることがわかる．また，$|SW|$ の増加に伴い，各ベースライン手法の計算時間は一定または増加するのに対して，提案手法の計算時間は減少する．ここで，計算時間はデータの生成および削除それぞれに対する計算時間（GT および ET）の合計である．$|SW|$ が増加すると，よりスコアの小さいデータのみが Top-k データとなり，新たに生成されたデータが Top-k データになり得るサブスクリプションの数が減少するため，



(a) *YELP* (b) *BRIGHTKITE*

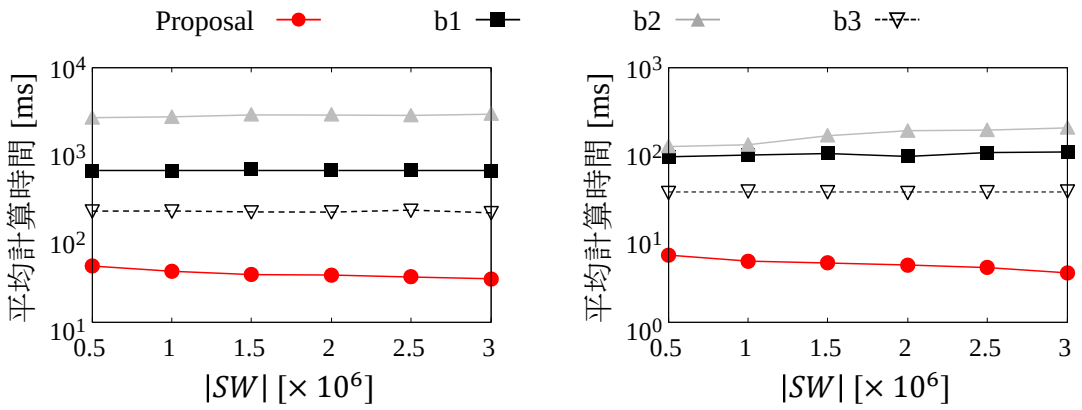
図 2.9: k_{max} の影響



(a) *YELP* (b) *BRIGHTKITE*

図 2.10: $|S|$ の影響

ベースライン 2，ベースライン 3，および提案手法では GT が減少する．一方，ベースライン 1 では常にすべてのサブスクリプションをチェックするため，GT はほぼ一定である．また，$|SW|$ が増加すると，SW から削除されるデータを Top-k データに含むサブスクリプションの数が減少し，Top-k データの更新を行う頻度が小さくなる．しかし，ベースライン 2 およびベースライン 3 において，Top-k データを更新する際，SW 内のすべてのデータを再評価するため，$|SW|$ が増加すると再評価するデータの数が増加し，ET が増加する．一方，ベースライン 1 および提案手



(a) *YELP*　　(b) *BRIGHTKITE*

図 2.11: $|SW|$ の影響

法では，k-スカイバンドを用いており，データの削除に対する Top-k データの更新を高速に行える．以上の理由から，提案手法のみ $|SW|$ の増加に伴い，計算時間が減少する．

2.6　むすび

位置依存型 Pub/Sub システムにおいて，情報受信側であるユーザは，自分自身の興味により合うデータを受信することを要求する．ソーシャル関係に基づくユーザの潜在的な興味を考慮した上で Top-k データを計算することで，各ユーザの興味により合うデータを配信することが可能になる．

本章では，位置，キーワード，およびソーシャル関係に基づき，各サブスクリプションに対して有用な上位 k 個のデータ（Top-k データ）をスライディングウィンドウ上でモニタリングする問題に対し，効率的に Top-k データをモニタリングする手法を提案した．提案手法では，各サブスクリプションに対して，k-スカイバンドを管理することで，あるデータが削除されたとき，そのデータを Top-k データに含むサブスクリプションの Top-k データを高速に更新する．また，サブスクリプションを四分木を用いて管理し，PoI ごとにソーシャルスコアリストを用いてソーシャルスコアを管理する．新たにデータが生成されたとき，四分木およびソー

シャルスコアリストを用いて，生成されたデータが Top-k データとなる可能性のあるサブスクリプションの候補を限定し，それらの k-スカイバンドおよび Top-k データのみを更新する．実データを用いた実験の結果から，データの生成および削除に対して，高速に各サブスクリプションの Top-k データを更新できることを確認した．

本章における今後の課題として，3 つの課題が考えられる．

1 つ目は，PoI の密度分布やサブスクリプションの密度分布の影響の評価である．本章における評価実験では，データセットに含まれる全ての PoI がデータを生成し，サブスクリプションは領域内のランダムな点を指定すると仮定した．提案手法は，サブスクリプションを位置情報に基づき木構造で管理するため，それぞれの密度分布が手法の性能に影響を及ぼすことが予想される．これらの影響を評価し，提案手法の有効性を検証する必要がある．

2 つ目は，タイムベースのスライディングウィンドウへの対応である．本章では，カウントベースのスライディングウィンドウ上で Top-k データをモニタリングすると想定した．しかし，アプリケーションによっては，カウントベースではなくタイムベースのスライディングウィンドウも考えられる．タイムベースのスライディングウィンドウを考慮した場合，複数のデータが同時に生成および削除されることが考えられる．このような環境に効率的に対応するため，手法の拡張を行う必要がある．

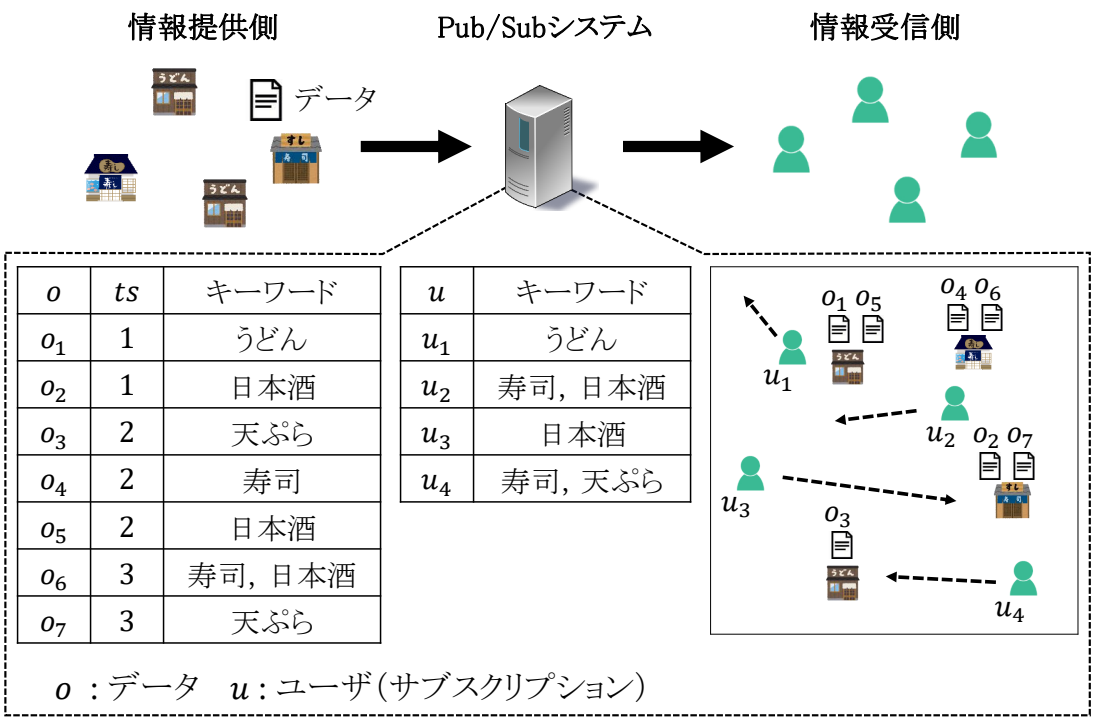
3 つ目は，ソーシャル関係の更新（リンクの追加および削除）への対応である．本章では，ソーシャル関係の更新を想定しなかった．ソーシャル関係を更新を考慮することで，ユーザの最新の興味に合う Top-k データをモニタリングできることが期待されるため，更新に対応できるように手法を拡張する必要がある．

第3章　ユーザの移動を考慮した位置・キーワードに基づく継続的なTop-k検索手法

3.1　まえがき

第2章では，事前にサブスクリプションとして指定した位置やキーワードといった明示的に指定された条件に加えて，ソーシャル関係に基づくユーザの潜在的な興味といった明示的に指定できない新たな条件を考慮した上で，位置依存型Pub/Subシステムにおいて，各サブスクリプションのTop-kデータを効率的にモニタリングする手法を提案した．一方，近年，スマートフォンやタブレットなどGPSを搭載した端末を使用し，ユーザが移動しながらアプリケーションを利用する機会が増加している．そのため，ユーザが事前に指定した位置に近いデータではなく，現在地に近いデータを配信することが有用となるアプリケーションも存在する．例えば，車を運転している際に，近くで空きのある駐車場の情報を受信するようなアプリケーションや観光客が歩きながらレストランを探しているとき，近くのレストランが発信するクーポンを受信するようなアプリケーションである．ユーザの移動を考慮し，現在地に基づいてデータを順位付けしTop-kデータを計算することで，ユーザの現在地に応じて有用なデータを配信できるサービスを提供することが可能になる．

例 3.1. 図3.1は，ユーザの移動を考慮した位置依存型Pub/Subシステムの例を示している．時刻$ts = 2$において，u_1，u_2，u_3，およびu_4のTop-1データは，それぞれo_1，o_4，o_5，およびo_3である．時刻$ts = 3$において，各ユーザが矢印の先へ移



o	ts	キーワード
o_1	1	うどん
o_2	1	日本酒
o_3	2	天ぷら
o_4	2	寿司
o_5	2	日本酒
o_6	3	寿司，日本酒
o_7	3	天ぷら

u	キーワード
u_1	うどん
u_2	寿司，日本酒
u_3	日本酒
u_4	寿司，天ぷら

図 3.1: ユーザの移動を考慮した位置依存型 Pub/Sub システムの例

動すると仮定する．このとき，u_3 に注目すると，移動後の位置では，o_5 より o_2 の方がより位置が近く，キーワードが類似しているため，u_3 の Top-1 データは o_2 に変化する．同様に，u_4 の Top-1 データは，o_7 が生成されても o_3 から変化しない．

このような背景から，いくつかの研究が，ユーザの移動に対して，位置およびキーワードに基づく Top-k データをモニタリングする問題に取り組んでいる [33, 77]．これらの研究では，ユーザが移動しても Top-k データが変化しない領域である Safe Region を計算することで，効率的に Top-k データをモニタリングしている．しかし，これらの研究では，データの生成および削除を考慮していない．データが生成および削除されると，Safe Region が変化する可能性があるため，効率的に Top-k データをモニタリングできない．一方，位置依存型 Pub/Sub システムにおける Top-k 検索に関する既存研究 [8, 71, 72] では，ユーザの移動を考慮していない．ユーザが移動すると，サブスクリプションの位置が変化する．そのため，サブスクリプションを木構造で単純に管理するためには，ユーザが移動する度に，新たなサブ

スクリプションが発行されたと仮定し，木構造に挿入する必要がある．その結果，データの生成に対して，Top-k データが変化する可能性のあるサブスクリプションの候補を効率的に限定できない．

本章では，位置依存型 Pub/Sub システムにおける，ユーザの移動を考慮した位置およびキーワードに基づく継続的な Top-k 検索手法を提案する．具体的には，ユーザの移動，データの生成および削除に対して，各サブスクリプションの Top-k データを効率的にモニタリングする新たなシステム Lamps（**L**ocation-**A**ware **M**oving Top-k **P**ub/**S**ub）を提案する．

以下では，3.2 節で，本章で扱う問題を詳細に定義する．3.3 節で提案手法について述べ，3.4 節で評価実験の結果を示す．3.5 節で関連研究について述べ，最後に 3.6 節で本章のまとめを行う．

3.2 問題定義

本節では，まず，位置およびキーワードに基づくデータおよびムービング Top-k サブスクリプションを定義する．その後，あるサブスクリプションに対する Top-k データを定義し，Top-k データを計算するために必要なスコアについて述べる．最後に，本章で扱う問題を詳細に定義する．

まず，位置およびキーワードに基づくデータおよびムービング Top-k サブスクリプションを定義する．以降，MkSK（**M**oving Top-***k*** **S**patial **K**eyword）サブスクリプションと呼ぶ．

定義 3.1 (データ)．ある PoI が生成するデータは，$o = (l, \mathcal{W})$ と定義される．ここで，$o.l$ は緯度と経度によって表される 2 次元平面上のデータ o の位置，$o.\mathcal{W}$ はデータ o が持つキーワードの集合である．

定義 3.2 (MkSK サブスクリプション)．あるユーザの発行する MkSK サブスクリプションは，$s = (l, \mathcal{W}, k, \alpha)$ と定義される．ここで，$s.l$ はサブスクリプション s を発行したユーザの現在地，$s.\mathcal{W}$ はサブスクリプション s の持つキーワードの集合 $(1 \leq |s.\mathcal{W}| \leq \mathcal{W}_{max})$，$s.k$ はユーザが要求するデータの個数，および $s.\alpha$ は s に

対するデータのスコアを計算するためのスコアリング関数において用いられる重み係数 $(0 < s.\alpha < 1)$ である．

今後，特に明記する必要がない場合，$s.k$ および $s.\alpha$ をそれぞれ k および α と表記する．

生成されたデータの集合を O とし，あるサブスクリプションに対する Top-k データを定義する．

定義 3.3 (Top-k データ)**.** データ集合 O が与えられたとき，あるサブスクリプション s の Top-k データ T は次の条件を満たす．(i) $T \subseteq O$，(ii) $|T| = s.k$，(ii) $\forall o \in T, \forall o' \in O \backslash T, score(s, o) \leq score(s, o')$．ここで，$score(s, o)$ は s に対するデータ o のスコアである．

本章では，あるサブスクリプション s に対するあるデータ o のスコア $score(s, o)$ として，文献 [33, 72] で用いられている以下の式を用いる．

$$score(s, o) = s.\alpha \cdot dist(s.l, o.l) + (1 - s.\alpha) \cdot text(s.\mathcal{W}, o.\mathcal{W})^1 \tag{3.1}$$

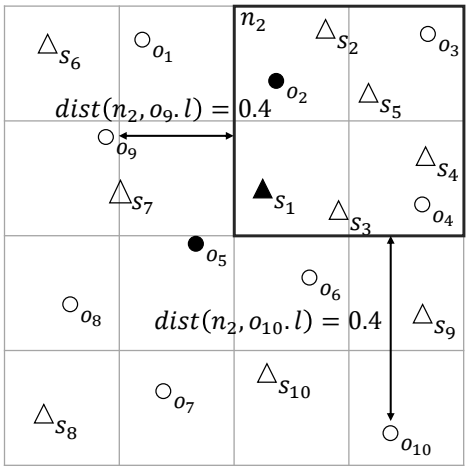
ここで，$dist(s.l, o.l)$ は $s.l$ と $o.l$ との空間距離を表すスコア（距離スコア），$text(s.\mathcal{W}, o.\mathcal{W})$ は $s.\mathcal{W}$ と $o.\mathcal{W}$ とのキーワードの類似度を表すスコア（キーワードスコア）である[2]．

距離スコアは，第2章と同様にユークリッド距離を用いて計算され，$dist(s.l, o.l) = \frac{Edist(s.l, o.l)}{Maxdist}$ である．キーワード集合同士の類似度を計算するために広く用いられるものとして，Jaccard，Dice，および Cosine 類似度が存在する [17]．本章では，キーワードスコアとして Jaccard 類似度を用いる．つまり，$text(s.\mathcal{W}, o.\mathcal{W}) = 1 - \frac{|s.\mathcal{W} \cap o.\mathcal{W}|}{|s.\mathcal{W} \cup o.\mathcal{W}|}$ である．また，3.4.3 項において，Dice および Cosine 類似度を用いた際の Lamps の性能を評価する．

問題定義． MkSK サブスクリプションの集合 S およびデータの集合 O を考える．PoI は，新たなデータを O に挿入することができ（データの生成），自身が生成し

[1] Lamps は，第 2 章におけるソーシャル関係を考慮したスコアを用いた場合にも対応できる．3.3.5 項において，ソーシャル関係を考慮したときの処理について議論する．

[2] 本章では，$dist(\cdot, \cdot)$ および $text(\cdot, \cdot)$ として，$[0, 1]$ に正規化できるものを用いるため，$score(s, o)$ は $[0, 1]$ である．



s	キーワード
s_1	w_1, w_2
s_2	w_3
s_3	w_2, w_4
s_4	w_3, w_4, w_5, w_6
s_5	w_2, w_5
s_6	w_1, w_5
s_7	w_2, w_4, w_5
s_8	w_1
s_9	w_2, w_3
s_{10}	w_7

o	キーワード
o_1	w_3, w_4
o_2	w_1, w_2, w_3
o_3	w_1, w_4, w_5
o_4	w_1, w_2, w_3
o_5	w_1, w_4
o_6	w_1, w_6, w_7
o_7	w_1, w_2
o_8	w_2
o_9	w_1, w_2, w_3, w_5
o_{10}	w_2, w_7, w_8

新たに生成される.

図 3.2: サブスクリプションおよびデータの位置およびキーワードの例

たデータを O から取り除くことができる（データの削除）．また，ユーザは，任意の時間にランダムに移動することができる（ユーザの移動）．本問題は，システムにおいて，S に含まれるすべてのサブスクリプションに対して，継続的に Top-k 検索を行うことである．具体的には，データの生成，データの削除，およびユーザの移動に対して，S に含まれるすべてのサブスクリプションの Top-k データを常に管理（モニタリング）する．本章は，ユーザの移動を考慮した上で，大量のサブスクリプションの Top-k データをモニタリングする問題に取り組んだ初めての研究である．

例 3.2. 図 3.2 は，本章を通して用いる例を示している．この例では，10 個のサブスクリプション $\{s_1, \cdots, s_{10}\}$ が存在し，8 個のデータ $\{o_1, \cdots, o_8\}$ がこれまでに生成されている．さらに，2 つのデータ o_9 および o_{10} が新たに生成され，$s_1.k = 2$ および $s_1.\alpha = 0.5$ であると仮定する．このとき，s_1 の Top-k データは，距離が近くキーワードが類似した o_2 および o_5 である．

3.3　提案手法

3.3.1　フレームワーク

図3.3は，Lampsのフレームワークを示している．文献[72, 73]と同様に，サブスクリプションはLampsに事前に登録されていると仮定する[3]．Lampsへの入力として，データの生成，データの削除，およびユーザの移動が含まれる．Lampsでは，ユーザの移動に対して，Top-kデータをモニタリングするため，既存研究[33, 77]と同様に，Safe Region[64]を用いる．ここから，それぞれの入力に対する処理について簡単に述べる．高速に処理を行うため，Lampsは，サブスクリプションを管理するサブスクリプションインデックス，Top-kデータを管理するTop-kインデックス，およびデータを管理するデータインデックスを保持する．

データの生成

新たにデータoが生成されたとき，oによってTop-kデータが変化するサブスクリプションのTop-kデータおよびSafe Regionを更新する必要がある．まず，データインデックスを更新した後，サブスクリプションインデックスを用いてTop-kデータが変化する可能性のあるサブスクリプションの候補を限定する．その後，各候補のTop-kデータおよびSafe Regionを更新する．Top-kデータおよびSafe Regionが変化した場合，その結果をユーザに配信すると同時に，サブスクリプションおよびTop-kインデックスを更新する．

データの削除

あるデータo'が削除されたとき，o'をTop-kデータに含むサブスクリプションのTop-kデータを更新する必要がある．まず，データインデックスを更新した後，Top-kインデックスを用いてo'をTop-kデータに含むサブスクリプションの集合を発見する．その後，各サブスクリプションに対して，データインデックスを用

[3]Lampsはサブスクリプションの更新（発行や削除）に対応できる．3.3.5項において，サブスクリプションが新たに発行または削除されたときの処理について議論する．

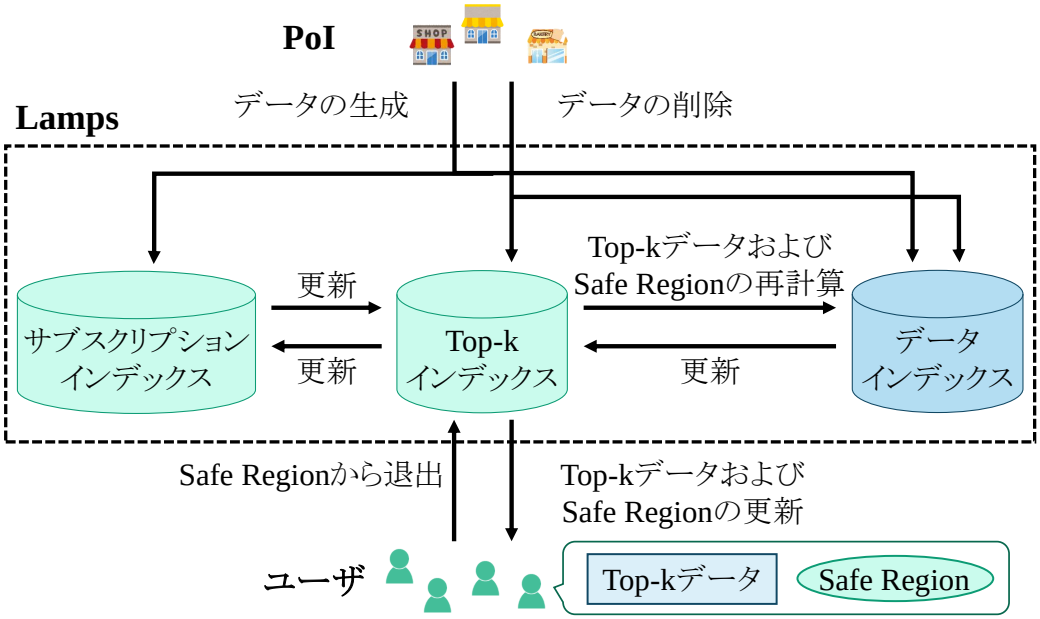


図 3.3: Lamps のフレームワーク

いて新たな Top-k データおよび Safe Region を再計算する．それらをユーザに配信すると同時に，サブスクリプションおよび Top-k インデックスを更新する．

ユーザの移動

各ユーザは，自身の Top-k データに加えて，Safe Region を管理している．そのため，各ユーザは自身が移動したとき，Safe Region の外に出たかどうかを判断できる．あるユーザが移動し Safe Region の外に出たとき，Lamps に現在地を送信する．このとき，Lamps はデータの削除の際と同様に，新たな Top-k データおよび Safe Region を再計算する．

以下では，まず，Lamps で用いる Safe Region について述べる．その後，データの生成および削除に対する処理の詳細について述べる．（上述した通り，あるユーザが Safe Region の外に出たとき，Lamps ではデータの削除と同様の処理を行う．）表 3.1 は，本章で用いる記号の概要を示す．

表 3.1: 記号の概要

記号	意味
o	データ
O	これまでに生成されたデータの集合
s	MkSK サブスクリプション
S	MkSK サブスクリプションの集合
w	キーワード
$\mathcal{W}_{max}$	$\|s.\mathcal{W}\|$ の最大値
R	Safe Region
c	R の重心
$\lambda(s, o\|\cdot)$	s に対する距離スコアの閾値
n	SQ-tree のノード
S_n^r	n を根とする部分木に含まれるサブスクリプションの集合
S_n	n で管理されるサブスクリプションの集合
$\mathcal{W}_n^r$	$s \in S_n^r$ に含まれるキーワード集合の和集合
I_n	転置ファイル
$\Lambda_n[\cdot], \Lambda_n^r[\cdot]$	S_n, S_n^r に対する距離スコアの閾値

3.3.2　Safe Region

本項では，まず，Safe Region および関連する概念である Dominant Region を定義する．その後，Safe Region を効率的に計算する既存手法，および既存手法を拡張し，より効率的に Safe Region を計算する提案手法について述べる．

まず，Safe Region および関連する概念である Dominant Region を定義する．

定義 3.4 (Safe region)**.** データの集合 O，サブスクリプション $s = (l, \mathcal{W}, k, \alpha)$，および s の Top-k データ T が与えられたとき，s の Safe Region R を次の条件を満たす領域として定義する．

$$R = \{l' | \forall o^* \in T, \forall o' \in O \backslash T, score(s', o^*) \leq score(s', o')\} \tag{3.2}$$

ここで，s' は s が新たな位置 l' に移動した後のサブスクリプションを示し，$s' = (l', \mathcal{W}, k, \alpha)$ である．

つまり，s の Safe Region は，s の Top-k データが変化しない領域を表す．

定義 3.5 (Dominant Region)**.** サブスクリプション $s = (l, \mathcal{W}, k, \alpha)$ および 2 つのデータ o^*，o' が与えられたとき，o^* の o' に対する Dominant Region $D_{o^*,o'}$ を次の条件を満たす領域として定義する．

$$D_{o^*,o'} = \{l' | score(s', o^*) \leq score(s', o')\} \tag{3.3}$$

ここで，s' は s が新たな位置 l' に移動した後のサブスクリプションを示し，$s' = (l', \mathcal{W}, k, \alpha)$ である．

つまり，o^* の o' に対する Dominant Region は，$score(s, o^*) \leq score(s, o')$ を満たす領域を表す．これらの定義から，以下の定理が成り立つ [33]．

定理 3.1. データの集合 O, サブスクリプション s, および s の Top-k データ T が与えられたとき, s の Safe Region R について次の式が成り立つ．$R = \cap_{o^* \in T}(\cap_{o' \in O \setminus T} D_{o^*,o'})$.

Local Safe Region

定理 3.1 より，Safe Region を正確に計算するためには，Top-k データに含まれる各データに対して，Top-k データに含まれない各データに対する Dominant Region を計算し，それらの共通部分を計算する必要があるため，計算コストが非常に大きい．そこで，文献 [33] では，Safe Region を効率的に計算するため，Safe Region の部分領域である Local Safe Region（LSR）を計算する手法が提案されている．ユーザ（サブスクリプション）が LSR の外に出たときのみ，Top-k データを再計算することで，正確な Top-k データをモニタリングできる．各データ o に対して，$o.l$ を中心，$r_o = \frac{1-\alpha}{\alpha} \cdot text(s.\mathcal{W}, o.\mathcal{W})$ を半径とする円を C_o とする．このとき，$score(s, o) = \alpha(dist(s.l, o.l) + r_o)$ と表現できる．LSR は以下の 3 ステップで計算される．

ステップ 1. 各データ $o^* \in T$ に対して，以下の条件を満たす楕円領域 $\mathcal{E}_{o^*}$ を計算し，それらの共通部分 $\mathcal{E} = \cap_{o^* \in T} \mathcal{E}_{o^*}$ を計算する．

$$\mathcal{E}_{o^*} = \{l' | dist(s'.l', o^*.l) + dist(s.l, s'.l') \leq \gamma - r_{o^*}\} \tag{3.4}$$

ここで，$\gamma = \max\{dist(s.l, o^*.l) + r_{o^*} | o^* \in T\} + \Delta$ であり，Δ は近似率を決める変数である．また，$\mathcal{E}_{o^*}$ は $s.l$ と $o^*.l$ を焦点とする楕円領域を表す．

ステップ 2. $s.l$ を中心，γ を半径とする円を C_s とする．$s'.l' \in \mathcal{E}$ であるとき，あるデータ o に対して C_o が C_s に含まれないならば，o は s' の Top-k データになり得ない．C_o が C_s に含まれないならば，$\gamma \leq dist(s.l, o.l) + r_o$ であるため，各データ $o^* \in T$ に対して $score(s', o^*) < score(s', o)$ である．ここで，s の Top-k データに含まれないデータの中で，C_o が C_s に含まれるデータの集合を O_s とする．各データ $o' \in O_s$ は，s' の Top-k データになり得るため，o' により s の Top-k データが変化しない領域を計算する必要がある．そこで，各データ $o' \in O_s$ に対して，$D_{o^*,o'}$ を計算し，それらの共通部分 $D = \cap_{o^* \in T}(\cap_{o' \in O_s} D_{o^*,o'})$ を計算する．

ステップ 3. $\mathcal{E}$ と D の共通部分 $\mathcal{E} \cap D$ を計算する．

例 3.3. 図 3.4 は，LSR の例を示している．ステップ 1 において，$s_1.k = 2$ および s_1 の Top-k データは o_2 および o_5 であるため，$\mathcal{E}_{o_2}$ と $\mathcal{E}_{o_5}$ の共通部分を計算する．ステップ 2 において，C_{o_1} および C_{o_3} は C_{s_1} に含まれないため，o_1 および o_3 は s'_1 の Top-k データになり得ない．O_{s_1} には o_4 および o_6 のみ含まれるため，o_4 および o_6 に対して Dominant Region を計算し，それらの共通部分 D を計算する．ステップ 3 において，それぞれの共通部分を計算するため，s_1 の LSR は，$\mathcal{E}_{o_2} \cap \mathcal{E}_{o_5} \cap D$ である．

定理 3.2. LSR の計算の時間計算量は $\mathcal{O}(\mu k |O_s|)$ である．ここで，μ は，2 つの多角形（Dominant Region）の共通部分を計算するコストである．

証明. ステップ 1 は，k 個の楕円領域およびそれらの共通部分を計算するため，時間計算量は $\mathcal{O}(k^2)$ である．次に，頂点の数がそれぞれ v_1 および v_2 である 2 つの多角形が与えられたとき，それらの共通部分を計算する時間計算量は $\mathcal{O}((a+b)\log_2(a+b))$

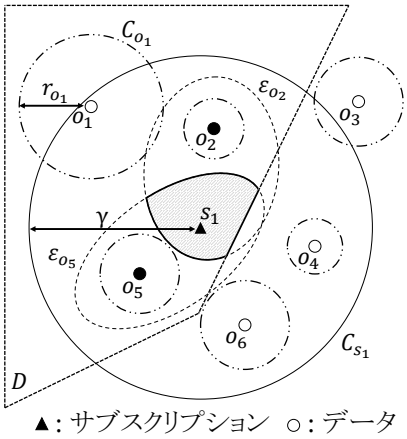


図 3.4: LSR の例．s_1 の LSR は黒く塗りつぶされた領域 $\mathcal{E}_{o_2} \cap \mathcal{E}_{o_5} \cap D$ である．

である．ここで，$a = v_1 + v_2$ であり，b は多角形の辺同士の交点の数である [85]．よって，$\mu = (a+b)\log_2(a+b)$ とすると，ステップ 2 の時間計算量は $\mathcal{O}(\mu k|O_s|)$ である．ステップ 3 の時間計算量は $\mathcal{O}(\mu)$ である．したがって，定理が成り立つ．□

Approximate Safe Region

LSR の計算には，O_s に含まれる各データ o' に対して，Dominant Region $D_{o^*,o'}$ を計算する必要がある．システムに多くのデータが存在する場合，O_s に多くのデータが含まれるため，LSR の計算コストは非常に大きくなる．この問題を解決するため，既存手法を拡張し，データの分布によらず近似的な Safe Region（ASR) を計算する手法を新たに提案する．LSR と同様に，ASR は正確な Safe Region の部分領域であるため，ユーザ（サブスクリプション）が ASR の外に出たときのみ，Top-k データを再計算することで，正確な Top-k データをモニタリングできる．

s の Safe Region を計算するとき，s に対する $(k+1)$ 番目のデータを o_{k+1} とする．ASR の計算では，$o^* \in T$ に対して，楕円領域 $\mathcal{E}_{o^*}$ を計算する際，$\gamma = dist(s.l, o_{k+1}.l) + r_{o_{k+1}}$ とする．このとき，以下の定理が成り立つ．

定理 3.3. $\gamma = dist(s.l, o_{k+1}.l) + r_{o_{k+1}}$ ならば，$O_s = \varnothing$ である．

証明. 各データ $o' \in O \backslash T$ に対して，$score(s, o_{k+1}) \leq score(s, o')$ であるため，$\frac{score(s,o_{k+1})}{\alpha} = dist(s.l, o_{k+1}.l) + r_{o_{k+1}} \leq \frac{score(s,o')}{\alpha} = dist(s.l, o'.l) + r_{o'}$ が成り立つ．よって，$C_{o'}$ は C_s に含まれない．したがって，定理が成り立つ． □

定理3.3より，ASRを計算する際，Dominant Regionを計算する必要がない．つまり，楕円領域およびそれらの共通部分を計算するだけでSafe Regionを計算できるため，Safe Regionの計算コストを削減できる．さらに，ASRは常にLSRの部分領域であるとは限らない．例えば，$\max\{dist(s.l, o^*.l) + r_{o^*} | o^* \in T\} + \Delta < dist(s.l, o_{k+1}.l) + r_{o_{k+1}}$ ならば，ASRはLSRの上位領域（スーパーセット）である．詳細は後述するが，ASRを高速に計算するため，Top-kデータの更新と同時に o_{k+1} を検索しておく．今後，Safe RegionはASRを指すものとする．つまり，$R = \mathcal{E}$ である．

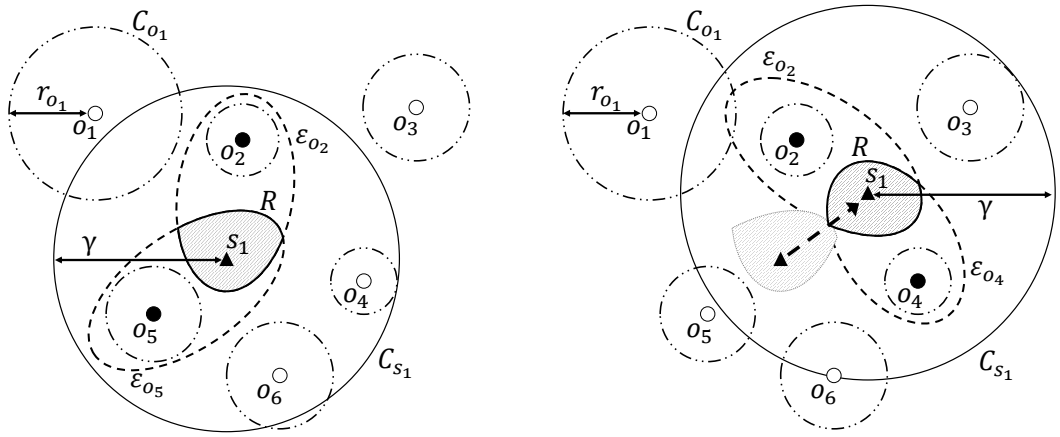
例 3.4. 図3.5は，s_1 のASRの例を示している．時刻 $ts = 1$ において，s_1 のTop-kデータは o_2 および o_5 であり，o_{k+1} は o_4 である．よって，$\gamma = dist(s_1.l, o_4.l) + r_{o_4}$ として，$s_1.R$ を計算する．次に，$ts = 2$ において，s_1 が $s_1.R$ の外に出るため，$s_1.R$ を再計算する必要がある．各データのスコアが変化するため，Top-kデータは o_2 および o_4，o_{k+1} は o_3 に変化する．よって，o_2，o_4，および o_3 に基づいて $s_1.R$ を再計算する．

定理 3.4. ASRの計算の時間計算量は $\mathcal{O}(k^2) \ll \mathcal{O}(\mu k|O_s|)$ である．

証明. ASRの計算は，k 個の楕円領域およびそれらの共通部分を計算するのみである．よって，定理が成り立つ． □

3.3.3　データの生成に対する処理

新たなデータ o が生成されたとき，o がTop-kデータになるサブスクリプションのTop-kデータを更新する必要がある．Top-kデータを高速に更新するためには，o によりTop-kデータが変化する可能性のあるサブスクリプションの候補を限定し，それらのTop-kデータおよびSafe Regionのみを更新することが望ましい．



▲: サブスクリプション ○: データ

(a) $ts = 1$

▲: サブスクリプション ○: データ

(b) $ts = 2$

図 3.5: ASR の例．各時刻において，s_1 の ASR は黒く塗りつぶされた領域である．

これを実現する方法として，既存手法では，R 木や四分木のような空間インデックスでサブスクリプションを管理する手法が提案されている．しかし，MkSK サブスクリプションは現在地が常に変化するため，これらのインデックスで単純に MkST サブスクリプションを管理できない．

この問題を解決するため，MkSK サブスクリプションの現在地を把握することなくサブスクリプションの候補を限定できる手法を提案する．これを実現するため，MkST サブスクリプションを管理できる新たなインデックス（SQ-tree）を提案する．SQ-tree は，MkST サブスクリプションを Safe Region に基づいて管理する新たな四分木である．新たにデータ o が生成されたとき，o により Top-k データが変化しないサブスクリプションをそれらの現在地を要求することなくまとめてフィルタリングできる．

以下では，まず，新たなフィルタリング手法を提案する．その後，SQ-tree の詳細，データの生成に対する詳細なアルゴリズムの順に述べる．最後に，SQ-tree の更新方法について述べる．

フィルタリング手法

新たにデータ o が生成されたとき，各サブスクリプションに対して o により Top-k データが変化するかどうか判断する必要がある．あるサブスクリプション s の k 番目のデータのスコアを $kscore(s)$ と表記する．このとき，o により s の Top-k データが更新されるためには，$kscore(s) > score(s, o)$ が必要である．ここで，位置およびキーワードを考慮した検索において，キーワードを優先したフィルタリングは，高速化に大きく貢献することが知られている [9]．そこで，$s.\mathcal{W}$ と $o.\mathcal{W}$ の間で一致するキーワードの数に基づいた新たなフィルタリング手法を提案する．

まず，$s.l$ が既知であると仮定する．（この仮定は後に取り除く．）また，$s.\mathcal{W}$ と $o.\mathcal{W}$ の間で一致するキーワードの数を i とする．つまり，$|s.\mathcal{W} \cap o.\mathcal{W}| = i$ である．このとき，$|s.\mathcal{W} \cap o.\mathcal{W}| = i$ であるような s に対して，$text(s.\mathcal{W}, o.\mathcal{W})$ の下界値 $text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i)$ が計算できる．

補助定理 3.1. $text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i) = 1 - \frac{i}{|s.\mathcal{W}|}$

証明. $s.\mathcal{W}$ と $o.\mathcal{W}$ の間で一致するキーワードの数が i 個であるとき，$text(s.\mathcal{W}, o.\mathcal{W})$ が最も小さな値を取るのは，$|o.\mathcal{W}| = i$ を満たすときである．$text(s.\mathcal{W}, o.\mathcal{W}) = 1 - \frac{i}{|s.\mathcal{W}|+|o.\mathcal{W}|-i}$ より，定理が成り立つ． □

補助定理3.1および式（3.1）より，$|s.\mathcal{W} \cap o.\mathcal{W}| = i$ であるような s に対して，Top-k データが更新されるための距離スコアの閾値 $\lambda(s, o|i)$ が計算できる．

$$\lambda(s, o|i) = \frac{kscore(s)}{\alpha} - \frac{1-\alpha}{\alpha} \cdot text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i) \tag{3.5}$$

式（3.5）より，以下の定理が成り立つ．

定理 3.5. $\lambda(s, o|i) < dist(s.l, o.l)$ ならば，新たに生成されたデータ o は s の Top-k データになり得ない．

証明. o が s の Top-k データとなるためには，$kscore(s) > score(s, o)$ が必要である．しかし，$\lambda(s, o|i) < dist(s.l, o.l)$ ならば，$kscore(s) < \alpha \cdot dist(s.l, o.l) + (1-\alpha) \cdot text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i)$ である．また，$text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i) \leq text(s.\mathcal{W}, o.\mathcal{W})$ より，$\alpha \cdot dist(s.l, o.l) + (1-\alpha) \cdot text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i) \leq \alpha \cdot dist(s.l, o.l) + (1-\alpha) \cdot text(s.\mathcal{W}, o.\mathcal{W}) = score(s, o)$ である．よって，定理が成り立つ． □

定理 3.5 より，$\lambda(s, o|i) < dist(s.l, o.l)$ ならば，正確性を失うことなく s をフィルタリングできる．

ここまで，$s.l$ が既知であると仮定していた．しかし，本章では，ユーザは常に移動しているため，$s.l$ は常に変化する．このとき，$s.l$ により $ksocre(s)$ が変化し，$\lambda(s, o|i)$ が変化するため，正確にフィルタリングできない．この問題を解決するため，$s.l$ が R 内の任意の位置であるとき，正確にフィルタリングが可能になるようにフィルタリング手法を拡張する．まず，$s.l$ が R 内の任意の位置であるとき，$kscore(s)$ の上界値 $kscore_{ub}(s)$ は以下で与えられる．

補助定理 3.2. $kscore_{ub}(s) = \frac{\gamma \cdot \alpha + kscore(s)}{2}$

証明. R は，s の各 Top-k データ $o^* \in T$ に対する $\mathcal{E}_{o^*}$ の共通領域であり，$\mathcal{E}_{o^*}$ は $s.l$ と $o^*.l$ を焦点とする楕円領域である．$\mathcal{E}_{o^*}$ と $s.l$ および $o^*.l$ の 2 点を通る直線が与えられたとき，$\mathcal{E}_{o^*}$ と直線の 2 つの交点のうち，o^* から遠い方の交点を p' とする．$s.l = l'$ であるとき，$score(s, o^*)$ は最も大きな値を取るため，$score(s, o^*)$ の上界値 $score_{ub}(s, o^*)$ は以下の式で与えられる．

$$\begin{aligned} score_{ub}(s, o^*) &= \alpha \cdot dist(p', o^*.l) + (1 - \alpha) \cdot text(s.\mathcal{W}, o^*.\mathcal{W}) \\ &= \alpha \cdot (dist(p', o^*.l) + r_{o^*}) \end{aligned}$$

一方, $dist(s.l, p') = dist(p', o^*.l) - dist(s.l, o^*.l)$ であるため, 式(3.4)より $dist(p', o^*.l) = \frac{\gamma - r_{o^*} + dist(s.l, o^*.l)}{2}$ である．よって，以下の式が成り立つ．

$$\begin{aligned} score_{ub}(s, o^*) &= \alpha \cdot (\frac{\gamma - r_{o^*} + dist(s.l, o^*.l)}{2} + r_{o^*}) \\ &= \frac{\gamma \cdot \alpha + score(s, o^*)}{2} \end{aligned}$$

$kscore_{ub}(s) = \max\{score_{ub}(s, o^*) | o^* \in T\}$ であるため，補助定理が成り立つ． □

式（3.5）および補助定理 3.2 より，$s.l$ が R 内の任意の位置にいるとき，Top-k データが更新されるための距離スコアの閾値が計算できる．

$$\lambda(s, o|i) = \frac{kscore_{ub}(s)}{\alpha} - \frac{1 - \alpha}{\alpha} \cdot text_{lb}(s.\mathcal{W}, o.\mathcal{W}|i) \tag{3.6}$$

式（3.6）より，以下の定理が成り立つ．

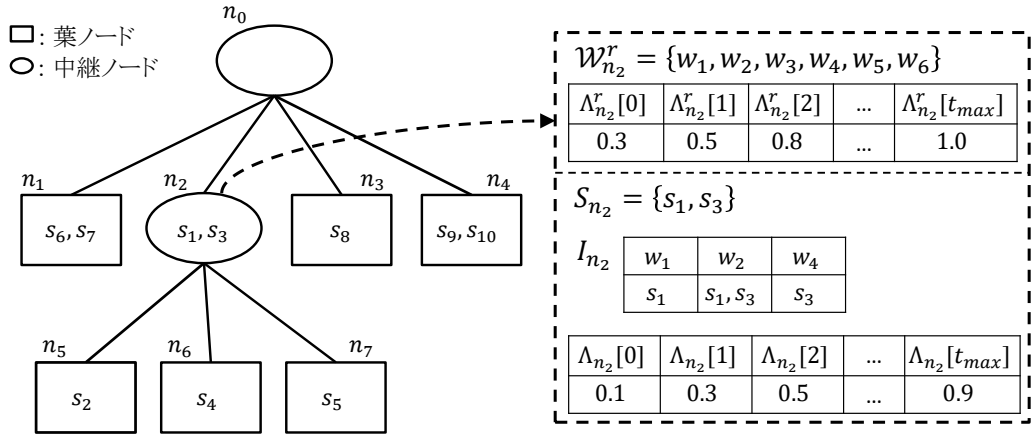


図 3.6: SQ-tree の例

定理 3.6. $\lambda(s, o|i) < dist(s.R, o.l)$ ならば，新たに生成されたデータ o は s の Top-k データになり得ない．ここで，$dist(s.R, o.l)$ は $o.l$ と $s.R$ 内の任意の点との距離スコアの最小値である．

証明. 定理 3.5 と同様. □

定理 3.6 より，$\lambda(s, o|i) < dist(s.R, o.l)$ ならば，正確性を失うことなく s をフィルタリングでき，$s.R$ の更新も考えなくてよい．

SQ-tree

SQ-tree の構造. SQ-tree は，図 3.6 に示す通り，MkST サブスクリプションを Safe Region に基づいて管理する新たな四分木である．上述したフィルタリング手法を適用することで，新たに生成されたデータ o により，Top-k データが変化しないサブスクリプションをそれらの現在地を要求することなくまとめてフィルタリングできる．SQ-tree の各ノードは，Safe Region の重心がノードの領域内に含まれるサブスクリプションを管理する．さらに，SQ-tree は，既存の空間インデックスとは異なり，葉ノードだけでなく中継ノードでもサブスクリプションを管理する．

SQ-tree のあるノード n は以下の情報を管理する．

- S_n：n で管理するサブスクリプションの集合.

- I_n：S_n に含まれるサブスクリプションのキーワード集合の転置ファイル．
- $\Lambda_n[\cdot]$：S_n に含まれるサブスクリプションの集合をフィルタリングするための距離スコアの閾値（式（3.9））．

n を根とする部分木で管理されるサブスクリプションの集合を S_n^r とする．n が中継ノードである場合，n は以下の情報も管理する．

- W_n^r：S_n^r に含まれるサブスクリプションのキーワード集合の和集合．
- $\Lambda_n^r[\cdot]$：S_n に含まれるすべてのサブスクリプションをフィルタリングするための距離スコアの閾値（式（3.10））．

例 3.5. 図3.2に示すようにサブスクリプションが存在し，三角形がSafe Regionの重心を表すと仮定すると，図3.6に示すSQ-treeが構築される．例えば，$S_n = \{s_1, s_3\}$ であるため，n_2 はそれらのキーワード集合の転置ファイルおよび $\Lambda_n[\cdot]$ を管理する．さらに，$S_{n_2}^r = \{s_1, s_2, s_3, s_4, s_5\}$ であるため，n_2 はそれらのキーワード集合の和集合および $\Lambda_n^r[\cdot]$ を管理する．

ここから，距離スコアの閾値 $\Lambda_n[i]$ および $\Lambda_n^r[i]$ の詳細について説明する．まず，$o.l$ と n との距離スコアの最小値 $dist(n, o.l)$ は以下の式で計算される．

$$dist(n, o.l) = \begin{cases} 0 & (o.l \in n) \\ dist(l^*, o.l) & (o.l \notin n) \end{cases}$$

ここで，$o.l \in n$ は，$o.l$ が n の領域内に含まれることを表し，l^* は，n の境界線上で $o.l$ に最も近い点である．

新たにデータが生成されたとき，$dist(n, o.l) \leq dist(s.R, o.l)$ が成り立つことに基づいてフィルタリングを行う．しかし，あるサブスクリプション s が n で管理される場合，R が完全に n に含まれる（$R \subseteq n$），または，R の一部が n に含まれる（$R \nsubseteq n$）という2つの場合が考えられる．R の重心を c とすると，図3.7の右図の場合では，$R \nsubseteq n$ であり，このとき，$dist(n, o.l) \leq dist(R, o.l)$ が常に成り立

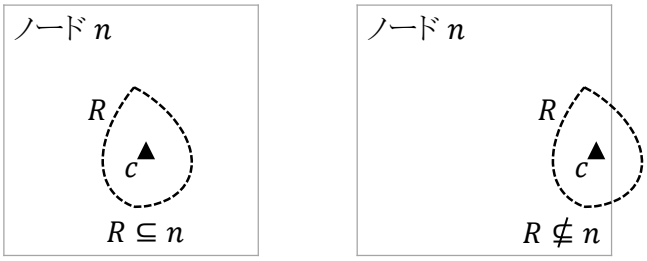


図 3.7: n で s を管理するときの例

つとは限らない．この問題を解決するため，R が n からどれだけはみ出ているかを表す追加の距離スコア $add(R,n)$ を考える．$add(R,n)$ は以下の式で与えられる．

$$add(R,n) = \begin{cases} 0 & (R \subseteq n) \\ dist_f(c,R) - dist_n(c,n) & (R \nsubseteq n) \end{cases} \tag{3.7}$$

ここで, $dist_f(c,R)$ は c から R の境界線上の最も遠い点までの距離スコア, $dist_n(c,n)$ は c から n の境界線上の最も近い点までの距離スコアである．このとき，以下の不等式が成り立つ．

$$dist(n,o.l) \leq dist(R,o.l) + add(R,n) \tag{3.8}$$

式（3.8）より，$\Lambda_n[i]$ および $\Lambda_n^r[i]$ はそれぞれ以下の式で定義される．

$$\Lambda_n[i] = \max\{\lambda(s,o|i) + add(s.R,n) | s \in S_n\} \tag{3.9}$$

$$\Lambda_n^r[i] = \max\{\lambda(s,o|i) + add(s.R,n) | s \in S_n^r\} \tag{3.10}$$

ここで，$|s.\mathcal{W}| < i$ ならば，$\lambda(s,o|i) = \lambda(s,o||s.\mathcal{W}|)$ である．

SQ-tree の構築. スコアは，位置，およびキーワードに基づいて計算されるため，両者を考慮して，サブスクリプションを分散するのが望ましい．しかし，R 木や四分木といった空間インデックスは，位置に基づいて領域を分割していく木構造であるため，キーワードの特徴に基づいてサブスクリプションを分散できない．この問題を解決するため，SQ-tree は，各サブスクリプションのキーワードに関する

特徴を考慮し，すべてのサブスクリプションを葉ノードで管理するのではなく，中継ノードでもサブスクリプションを管理する．ある中継ノード n でサブスクリプションを管理すると，n の各子ノード n' に対して， n' を根とする部分木に含まれるサブスクリプションの数 $|S^r_{n'}|$ が減少する．その結果，n' が管理する距離スコアの閾値 $\Lambda^r_n[\cdot]$ がタイトになり，n' をフィルタリングできる可能性が大きくなる．

ここから，サブスクリプションを管理するノードを決定するアルゴリズムについて述べる．あるサブスクリプション s をノード n で管理すべきかについて考える．ここで，あるデータ o が生成されたとき，n に対して計算される $dist(n, o.l)$ の期待値を $E[dist(n, o.l)]$[4] とする．このとき，s の Top-k データが更新されるために一致する必要のあるキーワードの数 $pnum(s, n)$ は，以下のように計算される．

$$pnum(s, n) = \lfloor (1 - \frac{score_{ub}(s) - \alpha \cdot E[dist(n)]}{1 - \alpha}) \cdot |s.\mathcal{W}| \rfloor \tag{3.11}$$

さらに，データに含まれるキーワードの分布が時間の経過とともに変化しないと仮定すると，$i_n = |\mathcal{W}^r_n \cap o.\mathcal{W}|$ の期待値 $E[i_n]$ は，以下の式で与えられる．

$$E[i_n] = \sum_{w \in \mathcal{W}^r_n} \frac{|O_w|}{|O|} \tag{3.12}$$

ここで，O_w はキーワード w を含むデータの集合である．また，$E[i_n] \geq \mathcal{W}_{max}$ ならば，$E[i_n] = \mathcal{W}_{max}$ とする．あるデータが生成されたとき，$E[i_n] < pnum(s, n)$ ならば，s がフィルタリングされる可能性が大きい．そのため，この条件を満たすサブスクリプションは，n が葉ノードでない場合においても，n で管理する．3.4.3 項において，$E[i_n]$ の正確性について評価を行う．

アルゴリズム 4 は，あるサブスクリプション s を SQ-tree のノード n に挿入するアルゴリズムを示している[5]．まず，S^r_n，$\mathcal{W}^r_n$，$E[i_n]$，および $\Lambda^r_n[\cdot]$ を更新する（1–2 行）．n が葉ノードである場合，S_n，I_n，および $\Lambda_n[\cdot]$ を更新する（3–5 行）．さらに，$|S_n|$ が SQ-tree の葉ノードが管理できるサブスクリプションの数の最大値を超えていれば，子ノードを作成し，$pnum(s, n) \geq E[i_n]$ を満たすサブスクリプションを子ノードに分

[4] $E[dist(n, o.l)] = \frac{1}{A} \iint_{\mathbb{R}^2} dist(n, o.l) dxdy$（$A$：領域 $\mathbb{R}^2$ の面積）

[5] SQ-tree を 1 から構築するとき，サブスクリプションおよびデータの存在し得る領域を SQ-tree の根ノード n_{root} とする．そして，各サブスクリプション $s \in S$ に対して，InsertSubscription(n_{root}, s) を実行する．

Algorithm 4: InsertSubscription(n, s)

Input: n and s

1 $S_n^r \leftarrow S_n^r \cup \{s\}$
2 Update T_n^r, $E[i_n]$, and $\Lambda_n^r[\cdot]$
3 **if** *n is a leaf node* **then**
4 　$S_n \leftarrow S_n \cup \{s\}$
5 　Update I_n and $\Lambda_n[\cdot]$
6 　**if** *$|S_n|$ is larger than the node capacity* **then**
7 　　Create child nodes, distribute subscriptions to child nodes, and recompute $\Lambda_n[\cdot]$.
8 **else**
9 　Distribute subscriptions with $pnum(s, n) \geq E[i_n]$ to child nodes.
10 　**if** $E[i_n] < pnum(s, n)$ **then**
11 　　$S_n \leftarrow S_n \cup \{s\}$
12 　　Update I_n and $\Lambda_n[\cdot]$
13 　**else**
14 　　**for** $\forall n' \in N$ *//N is n's children* **do**
15 　　**if** $s.c \in n'$ **then**
16 　　InsertSubscription(n', s)

配する．その後，分配されなかったサブスクリプションに対して $\Lambda_n[\cdot]$ を再計算する（6–7行）．一方，n が中継ノードである場合，$E[i_n]$ の更新により，$pnum(s, n) \geq E[i_n]$ となったサブスクリプションを子ノードに分配する（9行）．$E[i_n] < pnum(s, n)$ ならば，S_n，I_n，および $\Lambda_n[\cdot]$ を更新する．一方，$E[i_n] \geq pnum(s, n)$ ならば，$s.c$ が含まれる n の子ノード n' に対して，InsertSubscription(n', s) を再帰的に実行する（10–16行）．

SQ-tree によるフィルタリング手法

ここから，SQ-tree の各ノードで管理するサブスクリプションをフィルタリングする方法について述べる．このフィルタリングは，ノードフィルタリングおよび

サブスクリプションフィルタリングの 2 段階で行われる．あるデータ o が生成され，ノード n をチェックしたと仮定する．

サブスクリプションフィルタリング． まず，n に含まれるサブスクリプション中で Top-k データが変化しないサブスクリプションをフィルタリングする方法について述べる．$\Lambda_n[i] \geq dist(n, o.l)$ を満たす i の最小値を i_{min} とする．このとき，i_{min} は，o が S_n に含まれる各サブスクリプションの Top-k データになるために少なくとも一致する必要のあるキーワードの数を表す．そのため，以下の定理が成り立つ．

定理 3.7. 新たに生成されたデータ o および i_{min} が与えられたとき，$|s.\mathcal{W} \cap o.\mathcal{W}| < i_{min}$ ならば，o は $s \in S_n$ の Top-k データになり得ない．

証明. $|s.\mathcal{W} \cap o.\mathcal{W}| = j < i_{min}$ ならば，$\Lambda_n[j] < dist(n, o.l)$ である．また，$\lambda(s, o|j) + dist_{add}(s.R, n) \leq \Lambda_n[j]$ であるため，$\lambda(s, o|j) + dist_{add}(s.R, n) < dist(n, o.l)$ である．さらに，式（3.8）より，$\lambda(s, o|j) + dist_{add}(s.R, n) < dist(s.R, o.l) + dist_{add}(s.R, n)$ である．したがって，$\lambda(s, o|j) < dist(s.R, o.l)$ であるため，定理 3.6 より，定理が成り立つ． □

定理 3.7 より，正確性を失うことなく，$|s.\mathcal{W} \cap o.\mathcal{W}| < i_{min}$ を満たすサブスクリプションをフィルタリングでき，それらの Top-k データおよび Safe Region を更新する必要がない．

例 3.6. 図 3.2 および 3.6 において，o_9 が生成され，n_2 をチェックしたと仮定する．$\Lambda_{n_2}[1] < dist(n_2, o_9.l) < \Lambda_{n_2}[2]$ であるため，$i_{min} = 2$ である．したがって，S_n に含まれる各サブスクリプション s_1 および s_3 の Top-k データは，$|s.\mathcal{W} \cap o.\mathcal{W}| \geq 2$ である場合のみ，o_9 により更新される可能性がある．図 3.2 より，$|s_1.\mathcal{W} \cap o_9.\mathcal{W}| = 2$ および $|s_3.\mathcal{W} \cap o_9.\mathcal{W}| = 1$ であるため，s_1 のみ Top-k データを更新するサブスクリプションの候補に追加する．

ノードフィルタリング． 次に，n を根とする部分木に含まれるすべてのサブスクリプションをフィルタリングする方法について述べる．$i_n = \min\{|\mathcal{W}_n^r \cap o.\mathcal{W}|, \mathcal{W}_{max}\}$ が与えられたとき，以下の定理が成り立つ．

定理 3.8. $\Lambda_n^r[i_n] < dist(n, o.l)$ ならば，新たに生成されたデータ o は，S_n^r に含まれるすべてのサブスクリプションの Top-k データになり得ない．

証明. 各サブスクリプション $s \in S_n^r$ に対して $|s.\mathcal{W} \cap o.\mathcal{W}| = j$ ならば，$j \leq i_n$ である．よって，$\lambda(s, o|i)$ は，i の増加に伴い単調に増加するため，$\lambda(s, o|j) \leq \lambda(s, o|i_n)$ が成り立つ．$\Lambda_n^r[i_n] < dist(n, o.l)$ ならば，定理3.7の証明と同様にして，$\lambda(s, o|i_n) < dist(s.R, o.l)$ が導かれる．したがって，$\lambda(s, o|j) < dist(s.R, o.l)$ であるため，定理3.6より，定理が成り立つ．　□

定理3.8より，$\Lambda_n^r[i_n] < dist(n, o.l)$ ならば，正確性を失うことなく，n を根とする部分木をフィルタリングでき，S_n^r に含まれるすべてのサブスクリプションの Top-k データおよび Safe Region を更新する必要がない．

例 3.7. 図3.2および3.6において，o_{10} が生成され，n_2 をチェックしたと仮定する．$\mathcal{W}_{n_2}^r$ と $o_{10}.\mathcal{W}$ との間で一致するキーワードは w_2 のみであるため，$i_{n_2} = 1$ である．また，$dist(n_2, o_{10}.l) = 0.6$ より，$\Lambda_{n_2}^r[i_n] < dist(n_2, o_{10}.l)$ である．したがって，n_2 を根とする部分木はフィルタリングされ，$S_{n_2}^r$ に含まれるすべてのサブスクリプションの Top-k データおよび Safe Region は，o_{10} により更新されない．

アルゴリズム

アルゴリズム5は，SQ-tree に基づいたデータの生成に対するアルゴリズムを示している．このアルゴリズムは，まず，SQ-tree を用いて，新たに生成されたデータが Top-k データになり得るサブスクリプションの候補を発見し（1–15行），その後，各候補の Top-k データおよび Safe Region を更新する（16行）．具体的には，サブスクリプションの候補 S_g およびフィルタリングされていないノードを管理するヒープ H を初期化し，H に SQ-tree の根ノードを挿入する（1–2行）．その後，以下の処理を H が空になるまで繰り返し実行する．まず，i_n を計算する（5行）．n がノードフィルタリングによりフィルタリングされず（6行），中継ノードであった場合，n の各子ノード n' の中で $S_{n'}^r \neq \varnothing$ を満たすノードを H に挿入する（7–10行）．さらに，$S_n \neq \varnothing$ である場合，S_n に含まれるサブスクリプションに

Algorithm 5: HandleGeneratedData(o)

Input: o

1 $S_g \leftarrow \varnothing, H \leftarrow \varnothing$ //S_g is a set of candidate subscriptions
2 Push root node of SQ-tree into H
3 **while** $H \neq \varnothing$ **do**
4 $\quad n \leftarrow$ the node popped from H
5 $\quad i_n \leftarrow \min(|\mathcal{W}_n^r \cap o.\mathcal{W}|, \mathcal{W}_{max})$
6 $\quad$ **if** $\Lambda_n^r[i_n] \geq dist(n, o.l)$ **then**
7 $\quad\quad$ **if** *n is not a leaf node* **then**
8 $\quad\quad\quad$ **for** $\forall n' \in N$ *//N is n's children* **do**
9 $\quad\quad\quad\quad$ **if** $S_{n'}^r \neq \varnothing$ **then**
10 $\quad\quad\quad\quad\quad$ Push n' into H
11 $\quad\quad$ **if** $S_n \neq \varnothing$ **then**
12 $\quad\quad\quad i_{min} \leftarrow \min\{i | \Lambda_n[i] \geq dist(n, o.l)\}$
13 $\quad\quad\quad$ **if** $i_{min} \leq i_n$ **then**
14 $\quad\quad\quad\quad$ **for** $\forall s \in S_n$ *where* $|s.\mathcal{W} \cap o.\mathcal{W}| \geq i_{min}$ **do**
15 $\quad\quad\quad\quad\quad S_g \leftarrow S_g \cup \{s\}$
16 UpdateTopk(S_g)

対してサブスクリプションフィルタリングを実行する（12–15 行）．まず，i_{min} を計算する（12 行）．$i_n < i_{min}$ ならば，S_n に含まれる各サブスクリプションに対して $|s.\mathcal{W} \cap o.\mathcal{W}| < i_{min}$ が成り立つため，S_n に含まれるすべてのサブスクリプションをフィルタリングできる．そのため，$i_{min} \leq i_n$ を満たす場合のみ，S_n に含まれるサブスクリプション s の中で，$|s.\mathcal{W} \cap o.\mathcal{W}| \geq i_{min}$ を満たすものを S_g に加える（13–15 行）．H が空になった後，S_g に対して UpdateTopk(S_g) を実行する（16 行）．具体的には，S_g に含まれる各サブスクリプションを発行したユーザに現在地を要求する．そして，それぞれの現在地に基づいて，現在の Top-k データのスコアの更新および $score(s, o)$ の計算を行う．$score(s, o) < kscore(s)$ ならば，s の Top-k データが変化し（o が新たな Top-k データとなる），変更前の k 番目のデータが o_{k+1} となる．一方，$score(s, o) \geq kscore(s)$ ならば，s の Top-k データは変化が，Safe

Regionを更新する必要がある．しかし，oはo_{k+1}であるとは限らないため，データのインデックスを用いてo_{k+1}を検索する．（詳細は後述する．）その後，o_{k+1}に基づいて新たなSafe Regionを計算する．

SQ-treeのメンテナンス

あるサブスクリプションsのTop-kデータおよびSafe Regionが更新されたとき，sを管理するノードに保存された情報が変化する可能性があるため，SQ-treeを更新する必要がある．更新されたsのSafe Regionの重心cがsを管理するノードnに含まれている場合，単純にnおよびnの親ノードで管理する情報を更新する．一方，cがnに含まれない場合，sを一旦SQ-treeから取り除き，更新後のSafe Regionに基づいてSQ-treeに再度挿入する．

3.3.4　データの削除に対する処理

あるデータo'が削除されたとき，o'をTop-kデータに含むサブスクリプションのTop-kデータおよびSafe Regionを再計算する必要がある．あるサブスクリプションsのTop-kデータおよびSafe Regionを再計算するとき，sに対するk番目および$k+1$番目のデータ（o_k，o_{k+1}）を検索する必要がある．ここで，あるデータoがsのTop-kデータであるとき，oは次の条件のいずれかを満たす．(i) $o \in O_{kNN}$，(ii) $o \in O_{s.\mathcal{W}}$．ここで，O_{kNN}は$s.l$から近いk個のデータ集合，$O_{s.\mathcal{W}}$は$s.\mathcal{W} \cap o.\mathcal{W} \neq \varnothing$を満たすデータの集合である．

これらの条件を満たすデータを効率的に検索するため，KIQFと呼ぶ新たなインデックスを提案する．以下では，KIQFの詳細およびKIQFを用いてo_kを検索するアルゴリズムの詳細について述べる．

KIQF

KIQFは，図3.8に示すように，**K**d木，転置ファイル（**I**nverted file），および頻出なキーワードに対する四分木（**Q**uad-tree for each **F**requent keyword）を組

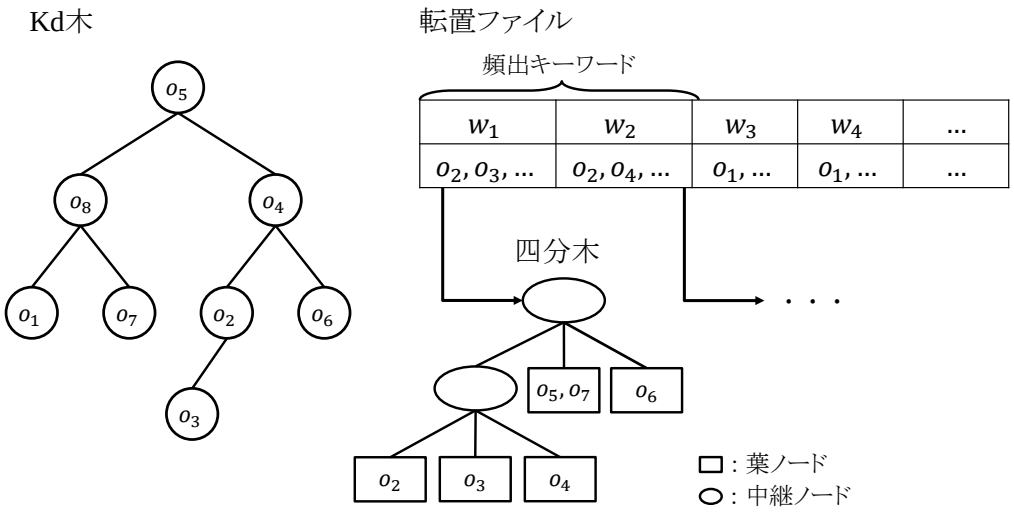


図 3.8: KIQF の例

み合わせたインデックスである．条件（i）を満たすデータを高速に検索するため，Kd 木を用いてすべてのデータの位置情報を管理する．条件（ii）を満たすデータを効率的に検索するため，転置ファイルを用いてキーワードごとにデータを管理する．しかし，あるキーワード $w \in s.\mathcal{W}$ が多くのデータに含まれている場合，w のポスティングリスト $PL[w]$ に含まれるすべてのデータをチェックするには多大な計算コストが必要である．この問題を解決するため，文献 [86] で提案されている Inverted linear Quad-tree のアイデアを適用する．Inverted linear Quad-tree は，各キーワードに対して，そのキーワードを含むデータの位置を管理する四分木を作成する．この手法は，位置およびキーワードに基づく Top-k 検索における最新の手法である．しかし，すべてのキーワードに対して四分木を作成すると膨大なメモリコストが必要である．そのため，KIQF では，すべてのキーワードのうち，出現頻度の大きいキーワード（頻出キーワード）w に対してのみ四分木を作成する．ここで，頻出キーワードは，すべてのキーワードのうち，出現頻度が上位 x% 以上のキーワードと定義する．（x が大きい場合，頻出でないキーワード w' に対して四分木を作成するため，メモリコストが大きくなる．そのため，評価実験では事前実験の結果に基づいて x を決定する．）

アルゴリズム

KIQF を用いて o_k を効率的に検索する方法を説明する．（ASR を計算する際，o_{k+1} が必要であるが，o_k と同様の方法で検索すればよい．）

$s.\mathcal{W}$ に含まれるキーワードを1つ以上含むデータの集合を $O_{s.\mathcal{W}}$ とする．各データ $o \in O_{s.\mathcal{W}} \backslash s.T$ に対して，$|s.\mathcal{W} \cap o.\mathcal{W}| = i$ ならば，$score(s, o)$ の下界値 $score_{lb}(s, o, |i, o_{nn})$ が以下の式で計算される．

$$score_{lb}(s, o|i, o_{nn}) = \alpha \cdot dist(s.l, o_{nn}.l) + (1 - \alpha) \cdot (1 - \frac{i}{|s.\mathcal{W}|}).$$

ここで，o_{nn} は，$O \backslash s.T$ に含まれるデータの中で $s.l$ に最も近いデータである．このとき，これまでにチェックされたデータの中で最もスコアの良いデータを o_b をすると，以下の定理が成り立つ．

定理 3.9. $score_{lb}(s, o|i, o_{nn}) \geq score(s, o_b)$ ならば，$O_{s.\mathcal{W}} \backslash s.T$ に含まれるデータの中で $|s.\mathcal{W} \cap o.\mathcal{W}| \leq i$ を満たすデータは，o_k になり得ない．

証明. $O_{s.\mathcal{W}} \backslash s.T$ に含まれるデータの中で $|s.\mathcal{W} \cap o.\mathcal{W}| \leq i$ を満たすデータ o に対して，$dist(s.l, o.l) \geq dist(s.l, o_{nn}.l)$ および $text(s.\mathcal{W}, o.\mathcal{W}) \geq 1 - \frac{i}{|s.\mathcal{W}|}$ より，$score(s, o) \geq score_{lb}(s, o|i, o_{nn})$ が成り立つ．よって，$score_{lb}(s, o|i, o_{nn}) \geq score(s, o_b)$ ならば，$score(s, o) > score(s, o_b)$ であるため，定理が成り立つ．　□

定理 3.9 より，$score_{lb}(s, o|i, o_{nn}) \geq score(s, o_b)$ ならば，正確性を失うことなく，$|s.\mathcal{W} \cap o.\mathcal{W}| < i$ を満たすデータをフィルタリングできる．

アルゴリズム 6 は，o_k を検索するアルゴリズムを表している．まず，Kd 木を用いて，$O \backslash s.T$ の中で $s.l$ に最も近いデータ返す RetrieveNearestData により，o_{nn} を検索する．そして，暫定的に o_{nn} を o_k とする（1–2 行）．$|s.\mathcal{W} \cap o.\mathcal{W}|$ の大きいデータが o_k になる可能性が大きいため，$O_{s.\mathcal{W}} \backslash s.T$ に含まれるデータを $|s.\mathcal{W} \cap o.\mathcal{W}| = i$ の降順にチェックする．$i > 1$ ならば，$|s.\mathcal{W} \cap o.\mathcal{W}| = i$ を満たすデータ $o \in O_{s.\mathcal{W}} \backslash s.T$ に対して，$score(s, o) < score(s, o_k)$ ならば，o を o_k とする（8–10 行）．一方，$i = 1$ ならば，各キーワード $w \in s.\mathcal{W}$ に対して，w が頻出キーワードであれば，SearchQF を実行し，そうでなければ，$PL[w]$ に含まれるすべてのデータ o をチェックし，

Algorithm 6: RetrieveData(s)

```
Input: s, O
1  o_nn ← RetrieveNearestData(s.l, s.T)
2  o_k ← o_nn
3  for ∀o ∈ O_{s.W} \ s.T do
4      Compute |s.W ∩ o.W| by using Inverted file
5  i ← |s.W|
6  while i > 0 ∧ score_lb(s, o, |i, o_nn) < score(s, o_k) do
7      if i > 1 then
8          for ∀o ∈ O_{s.W} \ s.T such that |s.W ∩ o.W| = i do
9              if score(s, o) < score(s, o_k) then
10                 o_k ← o
11     else
12         for ∀w ∈ s.W do
13             if w is a frequent keyword then
14                 SearchQF(w, s, o_k)
15             else
16                 for ∀o ∈ PL[w] \ s.T do
17                     if score(s, o) < score(s, o_k) then
18                         o_k ← o
19     i ← i − 1
20 return o_k
```

$score(s, o) < score(s, o_k)$ ならば，o を o_k とする（11–18 行）．ここで，SearchQF は，w に対する四分木に含まれるデータ o を $s.l$ から近い順にチェックし，$o \notin s.T$ かつ $score(s, o) < score(s, o_k)$ ならば，o を o_k とする関数である．チェック済みでないデータのスコアが $score(s, o_k)$ を下回ることがないと判断された時点で，検索を終了する．

例 3.8. 図 3.2 に示すようにデータが存在し，w_1 および w_2 が頻出キーワードであると仮定する．そのとき，図 3.8 に示すように KIQF が構築される．さらに，o_1

が削除され，s_2 の Top-k データを再計算すると仮定する．まず，Kd 木を用いて，$O \backslash s_2.T$ に含まれるデータの中で s_2 に最も近いデータ o_3 を得る．$s_2.\mathcal{W} = \{w_1, w_4\}$ であるため，まず，w_1 および w_4 を共に含むデータである o_3 や o_5 のチェックを行う．これらのデータをチェックした後，$score_{lb}(s_2, o|1, o_3) < score(s_2, o_k)$ ならば，w_1 または w_4 のどちらか一方を含むデータをチェックする必要がある．w_1 を含むデータをチェックするとき，w_1 に対する四分木に含まれるデータを $s_2.l$ から近い順にチェックする．一方，w_4 を含むデータをチェックするとき，$PL[w_4]$ に含まれるすべてのデータをチェックする．最後に，チェックされたデータの中で最もスコアの良いデータを，s_2 の新たな Top-k データとする．

3.3.5　議論

キーワードスコア． 本章では，キーワードスコアとして Jaccard 類似度を用いた．Dice および Cosine 類似度を用いる場合，補助定理 3.1 をぞれぞれの定義に従って変更することにより対応可能である．3.4.3 項において，それぞれの類似度関数を評価する．

サブスクリプションの発行及び削除に対する処理． 新たにサブスクリプション s が発行されたとき，アルゴリズム 6 と同様の方法で，s の Top-k データおよび o_{k+1} を計算できる．その後，o_{k+1} に基づいて，Safe Region およびその重心を計算し，最後に，SQ-tree に挿入する．あるサブスクリプション s' が削除されたとき，s' を SQ-tree から取り除き，必要があれば，s' を管理していたノード n および n の親ノードで管理する情報を更新する．

ソーシャル関係の考慮． 本章において，第 2 章におけるスコアを用いる場合，まず，スコアの定義に従って Safe Region を計算する．そして，SQ-tree のノードのうち，サブスクリプションを管理するノードに対してソーシャルスコアリストでソーシャルスコアを管理する．あるデータが生成され，SQ-tree のあるノードをチェックしたとき，ソーシャルスコアリストに基づいてそのノードのソーシャルスコアを計算し，アルゴリズム 5 の 6 行目および 12 行目においてソーシャルスコアを考慮した上でフィルタリングを行うことで，ソーシャル関係の考慮に対応可能である．

3.4 評価実験

本節では，Lampsの性能評価のために行った実験の結果を述べる．Lampsの有効性を検証するため，LSRとASRの計算時間，期待値$E[i_n]$の精度，各パラメータに対するスケーラビリティ，およびキーワードスコアの影響を評価した．スケーラビリティに関する実験では，サブスクリプションが指定するkの最大値，サブスクリプションの数，事前に生成されるデータの数，データの更新頻度，およびデータの更新の中でデータの削除が占める割合について評価した．

本実験は，Windows 10 Pro，3.20GHz Intel Core i7，および64GB 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．コンパイルには最適化オプションとして-O2を使用した．Pub/Subシステムに関する既存研究[72, 73]と同様に，リアルタイム性を保証するため，メインメモリ上で各インデックスの管理を行った．

3.4.1 評価環境

本章は，位置およびキーワードに基づく大量のMkSKサブスクリプションに対して，各サブスクリプションのTop-kデータをモニタリングする問題に取り組んだ初めての研究であるため，本問題に適用できる既存手法は存在しない．そのため，本実験では，ナイーブな手法(Naive)およびLampsのいくつかの機能を適用した3つの手法：Lamps w/o ASR，Lamps w/o SQ-tree，Lamps w/o KIQFとの比較を行った．

Naiveでは，ユーザの移動，データの生成および削除に対してそれぞれ以下のように処理を行う．ユーザの移動に対して，各サブスクリプションに対してLSRを管理し，LSRから出たユーザのTop-kデータおよびSafe Regionを更新する．データの生成に対して，MkSKサブスクリプションを管理する既存のインデックスは存在しないため，すべてのサブスクリプションのTop-kデータおよびSafe Regionを更新する．データの削除に対して，IQ-treeを用いて，新たにTop-kデータになるデータを発見する．IQ-treeとは，IR-tree[91]のように，転置ファイルと四分木を組み合わせたインデックスである．

表 3.2: データセットの詳細

データセット	**TWEETS**	**PLACES**
データの数	20M	9.4M
キーワードの種類	2.1M	54K
1つのデータに含まれるキーワードの数の平均	5.2	2.9

Lamps w/o ASR，Lamps w/o SQ-tree，および Lamps w/o KIQF は，ASR，SQ-tree，および KIQF をそれぞれ用いない手法であり，用いない部分は Naive と同様の処理を行う．

データセット． 本実験では，生成されるデータとして 2 つの実データを用いた．*TWEETS* は，アメリカ合衆国において投稿された 20 万個の位置情報付きツイートである [8]．*PLACES* は，アメリカ合衆国における 9.4 万個の公共施設のエントリであり，各エントリは位置情報とキーワード集合を含んでいる [48]．表 3.2 に，データセットの詳細を示す．

また，ユーザのトラジェクトリとして，実トラジェクトリ *OSM* および人工トラジェクトリ *SUS* を用いた．*OSM*[6]は，Open Street Map の GPS レコードを用いた．*SUS* は，アメリカ合衆国における道路ネットワークのデータセット[7]に基づいて，1,000,000 個のトラジェクトリを作成した．各トラジェクトリは，1 秒を 1 タイムスタンプ（ts）として，*OSM* は，100 個の連続する点を含むトラジェクトリを抽出し，*SUS* は，1,000 個の連続する点を含むトラジェクトリを作成した．

サブスクリプション． データのデータセットに基づいて，$|S|$ 個の MkST サブスクリプションを作成した．各サブスクリプションは，1 つのツイートまたはエントリの中に現れるキーワードの中から i 個のキーワードをランダムに選び，サブスクリプションのキーワードとする（$1 \leq i \leq 5$）[8]．トラジェクトリとして，抽出または

[6] `https://planet.openstreetmap.org/gps/`

[7] `https://www.cs.utah.edu/~lifeifei/SpatialDataset.htm`

[8] 文献 [8, 72] と同様に，Top-k データがキーワードに基づく解であることを保証するため，サブスクリプション s の Top-k データになり得るデータは $s.\mathcal{W}$ に含まれるキーワードを少なくとも一つ含む．

表 3.3: パラメータの設定

パラメータ	値
$s.k$ の最大値，k_{max}	5, **10**, 15, 20
サブスクリプションの数，$\|S\|$ [$\times 10^6$]	**0.25**, 0.5, 0.75, 1
事前に生成されるデータの数，$\|O_{init}\|$ [$\times 10^6$]	**1**, 2, 3, 4, 5
データの更新頻度，f	**100**, 200, 300, 400, 500
データの削除の割合，ϵ	**0.1**, 0.2, 0.3, 0.4, 0.5

作成したトラジェクトリの中からランダムに選び，サブスクリプション（ユーザ）のトラジェクトリをする．さらに，各サブスクリプション s に対して，$s.\alpha$ は 0 から 1 の間の一様乱数とし，$s.k$ は 1 から k_{max} の間の一様乱数とした．

データ． 本実験は，$|O_{init}|$ 個のデータが生成された時点から開始する．各タイムスタンプにおいて，f 個のデータの更新（データの生成または削除）が発生する．また，各タイムスタンプにおけるデータの削除の割合を ϵ とする．つまり，各タイムスタンプにおいて，$f \cdot (1-\epsilon)$ 個のデータが生成され，$f \cdot \epsilon$ 個のデータが削除される．ここで，3.1 節において述べたような実アプリケーションでは，データの削除の頻度は，データの生成の頻度に比べて小さい [68]．そのため，ϵ は小さな値を取り，$\epsilon = 0.1$ をデフォルトの値とする．

パラメータ． 表 3.3 により，本実験で用いたパラメータを示す．太字で表されている値はデフォルトの値である．

本実験では，各手法における平均計算時間（各 ts におけるデータの生成/削除およびユーザの移動に対して，各サブスクリプションの Top-k データの更新が完了するまでの時間）の結果について述べる．平均計算時間には，データの生成に対する処理時間（GT），データの削除に対する処理時間（ET），およびユーザの移動に対する処理時間（MT）が含まれる．

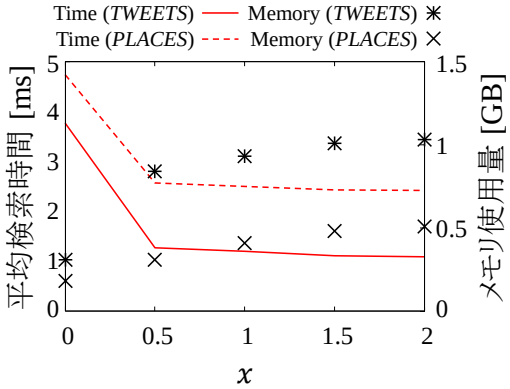


図 3.9: x の影響

3.4.2　準備実験

まず，x の値を決めるために行った準備実験の結果について述べる．ここで，x はデータに含まれるすべてのキーワードのうち頻出キーワードとするキーワードのパーセンテージである．3.3.4 項で述べた通り，Lamps におけるデータのインデックス（KIQF）では，頻出キーワード（出現頻度が上位 x%のキーワード）に対して位置情報を管理する四分木を作成する．

図 3.9 に，x を変えたときの平均検索時間（あるサブスクリプションに対して，o_k を検索する時間）およびメモリ使用量の結果を示す．x の増加に伴い，メモリ使用量が増加するのに対し，$x \geq 0.5$ では，平均計算時間はほぼ一定である．そこで，平均検索時間とメモリ使用量のトレードオフが取れているため，以降の実験では，x の値として 0.5 を用いる．

3.4.3　評価結果

LSR と ASR の計算時間の比較． 図 3.10 に，各データセットを用いた際の LSR および ASR の計算時間（ts あたりの，Safe Region の外に出たユーザの Safe Region を計算する時間）の結果を示す．ASR は LSR より小さくなる傾向にあるため，ts あたりに Safe Region の外に出るサブスクリプションの数は多くなる．しかし，ASR は，Dominant Region およびそれらの共通領域を計算する必要がないため，LSR と比較して非常に高速に Safe Region を計算できる．そのため，Safe Region の外に出

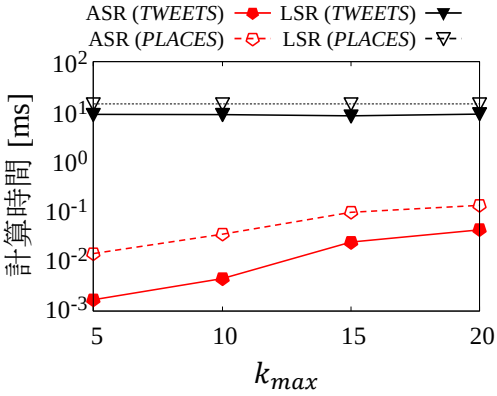


図 3.10: LSR と ASR の計算時間の比較

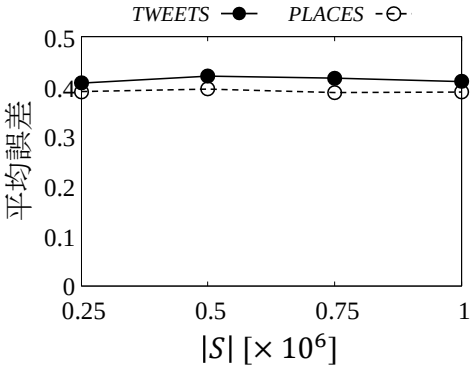


図 3.11: $E[i_n]$ の精度

るサブスクリプションの数が増加しても，ASR を用いた場合，高速に Safe Region を計算可能である．

$E[i_n]$ の精度． 図 3.11 に，$|S|$ を変えたときの平均誤差（ノードフィルタリングにおいて計算される $|E[i_n] - i_n|$ の平均）の結果を示す．$|S|$ に関わらず，平均誤差が 1 を下回っており，式（3.12）により，$E[i_n]$ を十分な精度をもって求めることができている．

各手法の性能の比較

表 3.4 に，各手法における平均計算時間の結果を示す．使用したデータおよびトラジェクトリのデータセットによらず，すべての手法の中で，Lamps は最も高速に Top-k データを更新できることがわかる．また，*TWEETS* を使用した場合よりも *PLACES* を使用した場合の方が平均計算時間が大きくなる．これは，*TWEETS* に比べて *PLACES* の方がキーワードの種類が少なく，各データおよびサブスクリプションが同じキーワードを含みやすくなる．その結果，新たに生成されたデータにより Top-k データが変化する可能性のあるデータの数や KIQF の各キーワードに対するポスティングリストのサイズが大きくるため，平均計算時間が大きくなる．また，上述した通り，ASR は LSR と比較してより優れた性能を持つため，Lamps w/o ASR と比較して，Lamps はより高速に Top-k データを更新できる．さらに，Lamps は，新たにデータが生成されたとき，SQ-tree を用いて Top-k データ

表 3.4: 各手法における平均計算時間 [ms]

手法	*TWEETS*		*PLACES*	
	OSM	*SUS*	*OSM*	*SUS*
Naive	6657.9	5858.8	11242.1	9666.6
Lamps w/o ASR	2124.8	1746.1	2616.7	3399.3
Lamps w/o SQ-tree	2436.6	2424.7	2366.2	2320.5
Lamps w/o KIQF	155.3	158.0	215.1	210.1
Lamps	80.0	75.2	156.3	141.2

表 3.5: 各手法におけるメモリ使用量 [GB]

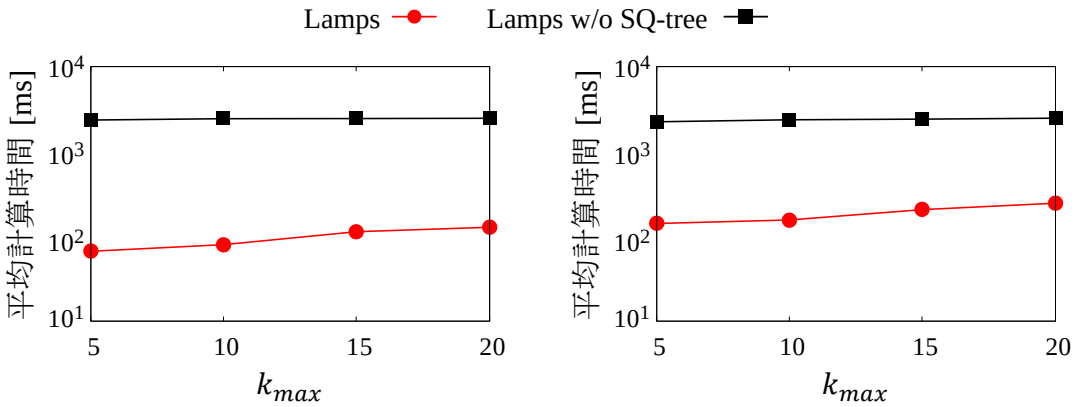
method	*TWEETS*	*PLACES*
Lamps w/o SQ-tree	0.84	0.31
Lamps w/o KIQF	1.63	0.57
Lamps	1.03	0.42

が変化しないサブスクリプションをフィルタリングするため，Lamps w/o SQ-tree より高速に Top-k データを更新できる．

Lamps は Lamps w/o KIQF と比較して高速に Top-k データを更新できるだけでなく，メモリ使用量の観点でも優れている．表 3.5 に，*OSM* を用いたときの各手法におけるメモリ使用量の結果を示す．Lamps は，Lamps w/o KIQF と比較してメモリ使用量を抑えることができていることがわかる．これは，Lamps w/o KIQF は KIQF の代わりに IQ-tree を用いているためである．IQ-tree は，各ノードにおいて，そのノードを根とする部分木に含まれるデータのキーワード集合の転置ファイルを管理するため，より多くのメモリを必要とする．

各パラメータの影響

以降の実験では，Lamps と Lamps w/o SQ-tree との比較を行った．また，データの更新頻度 f およびデータの削除の割合 ϵ の影響を評価する際，Lamps w/o KIQF



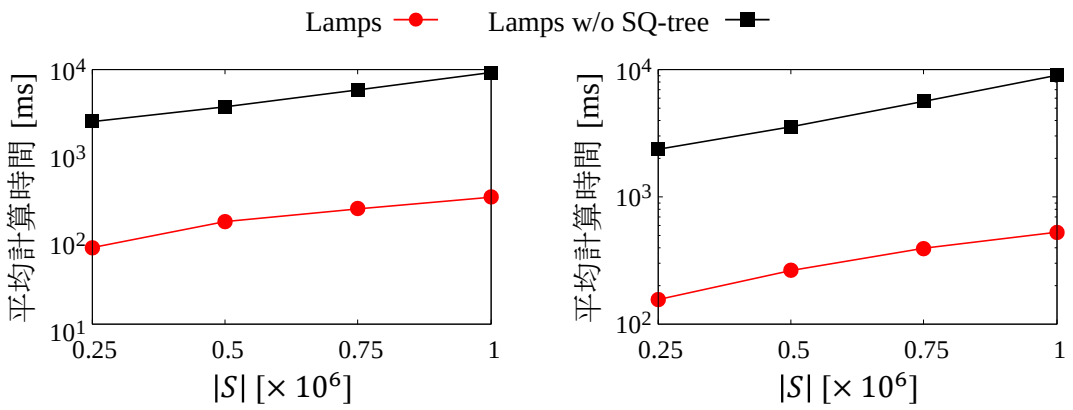
(a) *TWEETS* (b) *PLACES*

図 3.12: k_{max} の影響

との比較も行った．さらに，各トラジェクトリにおける平均計算時間に大きな差は存在しないため，トラジェクトリとして *OSM* を用いた．

k_{max} の影響． 図 3.12 にサブスクリプションの指定する k の最大値 k_{max} を変えたときの結果を示す．Lamps は Lamps w/o SQ-tree と比較して，常に高速に Top-k データを更新できることがわかる．また，k_{max} の増加に伴い，両手法において平均計算時間が増加することがわかる．k_{max} が増加すると，あるデータが削除されたとき，そのデータを Top-k データに含むサブスクリプションの数が増加するため，ET が増加する．また，k_{max} が増加すると，各サブスクリプション s に対する $kscore(s)$ が大きくなるため，新たに生成されたデータが Top-k データになり得るサブスクリプションの数が増加するため，Lamps における GT は増加する．一方，Lamps w/o SQ-tree は，常にすべてのサブスクリプションの Top-k データの更新をチェックするため，Lamps w/o SQ-tree における GT は，k_{max} によらず一定である．その結果，両手法の平均計算時間の差は，k_{max} の増加に伴い僅かに小さくなる．

$|S|$ の影響． 図 3.13 にサブスクリプションの数 $|S|$ を変えたときの結果を示す．Lamps は Lamps w/o SQ-tree と比較して，常に高速に Top-k データを更新できることがわかる．また，$|S|$ の増加に伴い，両手法において平均計算時間が増加し，さらに，両手法の平均計算時間の差が大きくなることがわかる．新たにデータが生



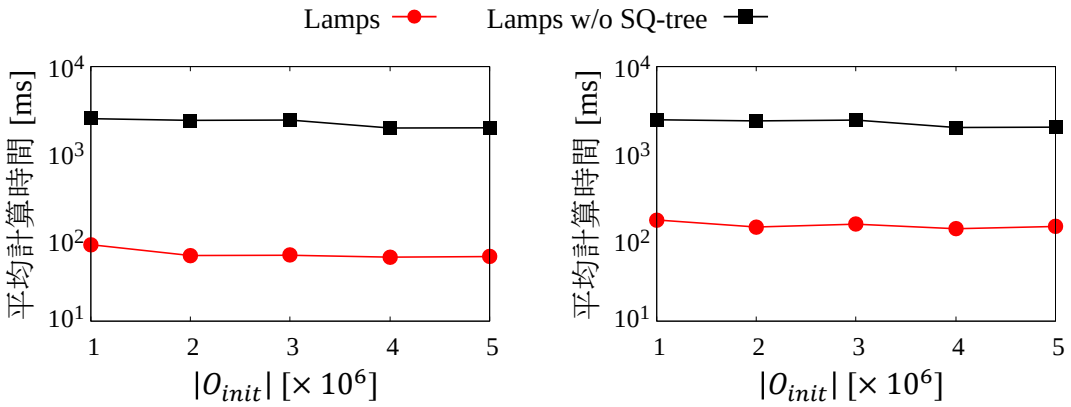
(a) *TWEETS*　　(b) *PLACES*

図 3.13: $|S|$ の影響

成されたとき，Lamps w/o SQ-tree は，常にすべてのサブスクリプションの Top-k データの更新をチェックするのに対し，Lamps は，Top-k データが変化しないサブスクリプションをフィルタリングし，チェックを行わない．その結果，$|S|$ の増加に伴い，Top-k データの更新をチェックするサブスクリプションの数の差が大きくなるため，平均計算時間の差が大きくなる．

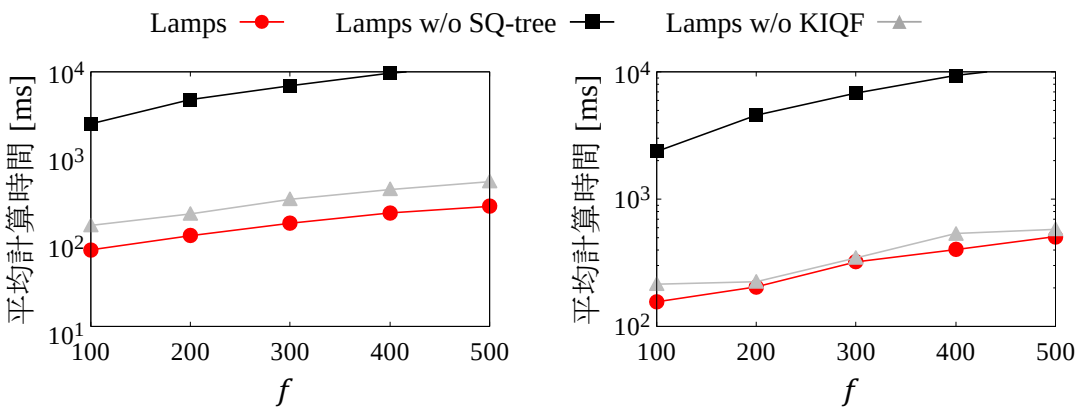
$|O_{init}|$ の影響． 図 3.14 に事前に発生するデータの数 $|O_{init}|$ を変えたときの結果を示す．Lamps は Lamps w/o SQ-tree と比較して，常に高速に Top-k データを更新できることがわかる．また，$|O_{init}|$ の増加に伴い，両手法において平均計算時間が僅かに減少することがわかる．$|O_{init}|$ が増加すると，Safe Region の外に出るユーザの数や Top-k データおよび Safe Region を再計算する際にチェックするデータの数が増加する．一方，新たに生成されたデータが Top-k データになり得るサブスクリプションの数や，削除されたデータを Top-k データに含むサブスクリプションの数が減少する．後者の影響の方が大きいため，平均計算時間が僅かに減少する．

f の影響． 図 3.15 にデータの更新頻度 f を変えたときの結果を示す．Lamps は他の手法と比較して，常に高速に Top-k データを更新できることがわかる．また，f の増加に伴い，各手法において平均計算時間が増加することがわかる．これは，各手法は新たに生成または削除されるデータに対して順に処理を行うため，各手法における GT および ET が線形に増加するためである．



(a) *TWEETS* (b) *PLACES*

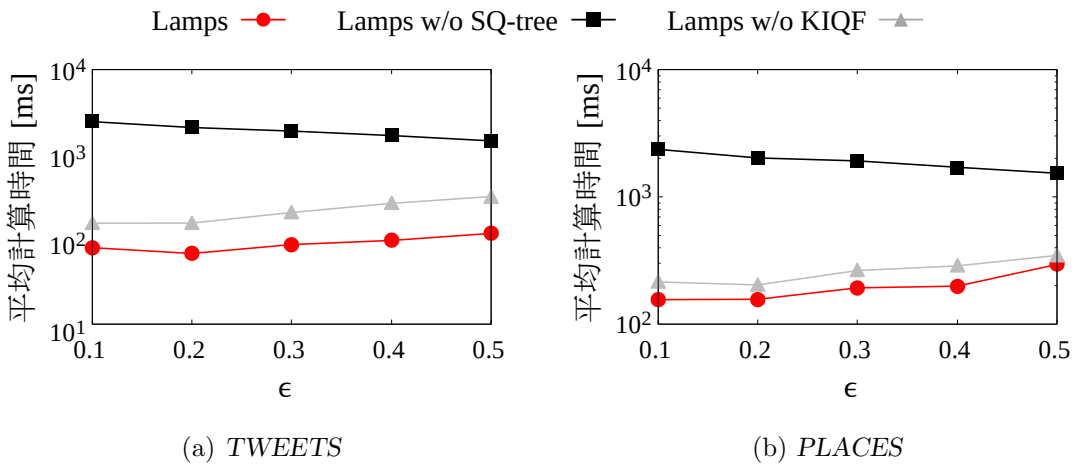
図 3.14: $|O_{init}|$ の影響



(a) *TWEETS* (b) *PLACES*

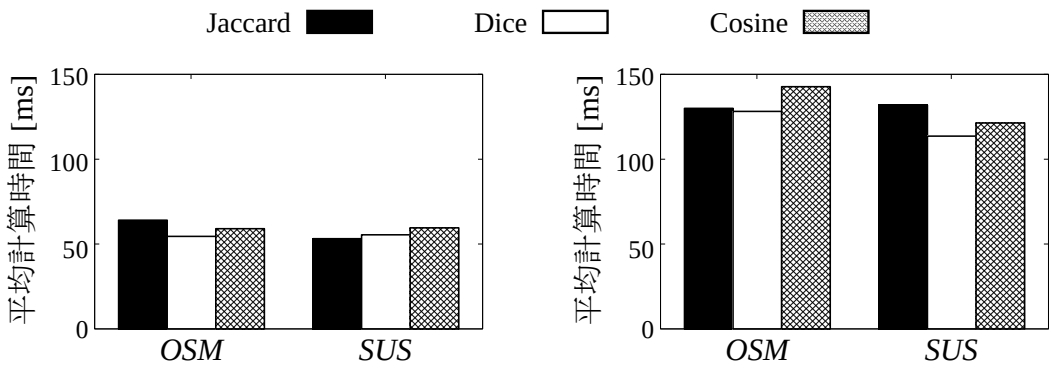
図 3.15: f の影響

ϵ の影響． 図 3.16 にデータの削除の割合 ϵ を変えたときの結果を示す．本実験では，各タイムスタンプにおいて，f 個のデータの更新が発生し，$f \cdot (1-\epsilon)$ 個のデータが生成され，$f \cdot \epsilon$ 個のデータが削除される．Lamps は他の手法と比較して，常に高速に Top-k データを更新できることがわかる．Lamps および Lamps w/o KIQF は SQ-tree を用いてデータの生成に対して効率的に Top-k データを更新できるため，それぞれの手法における GT は ET より小さい．そのため，ϵ の増加に伴い，ET が増加するため平均計算時間が増加する．一方，Lamps w/o SQ-tree は，常に



(a) *TWEETS*　　(b) *PLACES*

図 3.16: ϵ の影響



(a) *TWEETS*　　(b) *PLACES*

図 3.17: キーワードスコアの影響

すべてのサブスクリプションの Top-k データの更新をチェックするため，Lamps w/o SQ-tree における GT は ET より大きい．そのため，ϵ の増加に伴い，GT が減少するため平均計算時間が減少する．

キーワードスコアの影響． 図 3.17 に各キーワードスコア（Jaccard，Dice，および Cosine 類似度）を用いたときの結果を示す．いずれの類似度関数を用いた場合でも，平均計算時間に大きな差はなく，Lamps の性能に影響を及ぼさないことを確認した．

3.5 関連研究

2.3 節において，Pub/Sub システムにおけるデータモニタリングに関する従来研究について述べた．本節では，ムービングサブスクリプションに関する従来研究について述べる．

これまでに様々な研究が位置およびキーワードに基づくムービングサブスクリプションの解をモニタリングする問題に取り組んでいる [26, 33, 39, 40, 73, 77]．文献 [26, 33, 39, 40, 77] では，Safe Region と呼ばれるユーザが移動しても解が変化しない領域を計算することで，効率的にムービングサブスクリプションの解をモニタリングする手法を提案している．しかし，文献 [33, 39, 40, 77] では，新たにデータが生成および削除されることを考慮していない．新たにデータが生成または削除されると，Safe Region が変化する可能性がある．そのため，データが生成および削除されるたびに Safe Region を再計算する必要があり非常に計算コストが大きい．また，文献 [26, 73] では，データが新たに生成および削除される環境において，ムービングサブスクリプションの解をモニタリングする問題に取り組んでいる．例えば，文献 [26] では，ユーザが Safe Region 内の任意の位置にいるとき，新たに生成されるデータが解となる可能性がある領域（Impact Region）を計算することで，効率的にムービングサブスクリプションの解をモニタリングする手法を提案している．しかし，これらの文献では，サブスクリプションの指定した範囲内に含まれるデータをモニタリングするレンジクエリを対象としており，本章における問題と異なる．文献 [33, 77] では，ムービング Top-k サブスクリプションの解をモニタリングする問題に取り組んでいる．しかし，上述した通り，これらの文献では新たにデータが生成および削除されることを考慮していない．

3.6 むすび

位置依存型 Pub/Sub システムにおいて，情報受信側であるユーザは，自分自身の興味により合うデータを受信することを要求する．ユーザの移動に伴うユーザの現在地を考慮した上で Top-k データを計算することで，各ユーザの現在地に応じて有用なデータを配信することが可能になる．

本章では，Pub/Subシステムにおいて，サブスクリプション（ユーザ）の移動を考慮しながら，位置およびキーワード基づくTop-kデータをモニタリングする問題に対して，ユーザの移動，データの生成および削除に対して，効率的に各サブスクリプションのTop-kデータをモニタリングするシステムLampsを提案した．Lampsでは，ユーザの移動に対して，Top-kデータが変化する場合を把握するため，各サブスクリプションに対してSafe Regionを計算する．このとき，既存手法を拡張し，近似的なSafe Region（ASR) を計算する手法を提案した．データの生成に対して，Top-kデータが変化するサブスクリプションのTop-kデータおよびSafe Regionを高速に更新するため，計算されたASRに基づいてサブスクリプションを管理する新たなインデックス（SQ-tree）を提案し，SQ-treeを用いて，生成されたデータによりTop-kデータが変化する可能性があるサブスクリプションの候補を限定する手法を提案した．データの削除に対して，サブスクリプションのTop-kデータおよびSafe Regionを再計算する際，新たにTop-kデータとなり得るデータを効率的に発見できるインデックスKIQFを提案した．実データを用いた実験の結果から，Lampsは，ユーザの移動，データの生成および削除に対して，高速に各サブスクリプションのTop-kデータを更新できることを確認した．

本章における今後の課題として，サブスクリプションが指定するキーワードの動的な変化への対応が考えられる．本章では，各ユーザが自身の興味のある位置やキーワードなどの条件を指定し，有用なデータを受信しようとするが，検索対象となるデータは大量に存在するため，自身の要求を満たす検索条件を指定することは難しい．この問題を解決するため，文献 [10, 12, 13, 44] では，位置およびキーワードに基づくWhy-Notクエリの解を計算する手法を提案している．Why-Notクエリは，あるサブスクリプション，サブスクリプションの解，および解に含まれなかったデータの集合Oが与えられたとき，Oに含まれるデータが解に含まれるように指定する条件を変化させた新たなサブスクリプションを計算する．各サブスクリプション（ユーザ）に対して，これらの研究で提案されている手法を適用することで，ユーザは自身の要求を満たすサブスクリプションを登録することが可能になる．しかし，本章の提案手法は，サブスクリプションの指定するキーワードが動的に変化することを想定しておらず，キーワードが変化する度に，新たに

サブスクリプションが発行されたとして処理を行う必要があるため効率的でない．このような環境において，各サブスクリプションの Top-k データを効率的にモニタリングできる手法を実現する必要がある．

第4章　位置・キーワードに基づく逆 Top-k検索手法

4.1　まえがき

情報配信や情報推薦の分野において，情報提供側が自身が生成したデータに興味を持つユーザ（潜在顧客）を把握することは，市場分析やマーケティング調査を行う上で重要な課題である [14, 70]．潜在顧客の統計的な特性を分析することによって，新たな製品の開発，新たな顧客の獲得，およびアップセリングやクロスセリングを行うことが可能となる．ここで，Pub/Sub システムにおいて，あるデータに対する逆 Top-k 検索の解は，そのデータに対する潜在顧客とみなすことができる．逆 Top-k 検索では，データ集合およびサブスクリプション集合が与えられ，クエリとしてあるデータ（クエリオブジェクト）を指定したとき，クエリオブジェクトを上位 k 個のデータ（Top-k データ）に含むサブスクリプション（ユーザ）を検索する．Pub/Sub システムにおいて，逆 Top-k 検索を行い，潜在顧客を高速に把握できるサービスを提供することで，情報提供側は，情報配信や情報推薦を効率化でき，売り上げの向上や市場規模の拡大が期待できる．

市場分析などへの応用では，潜在顧客を厳密に把握する必要のない場合も多く存在する．あるサブスクリプション s に対する k 番目のデータ o と $k+1$ 番目のデータ o' のスコアが近い値を取る場合，それらの有用度は同程度であると考えられる．このとき，s は o' に対する逆 Top-k 検索において近似的な解とみなすことができる．このようなサブスクリプションを検索可能になれば，より高度な市場分析への応用が期待できる．

本章では，近似逆 Top-k クエリ（ART クエリ）を新たに提案する．ART クエリの解を求める最も単純な方法は，クエリの解となる可能性のあるすべてのサブスク

リプションに対して Top-k 検索を行うものである．これは，サブスクリプションが多く存在する環境では多大な時間がかかる．Pub/Sub システムにおける逆 Top-k 検索の応用では，ある時刻に複数の PoI が同時にクエリを発行することが考えられる．例えば，複数のレストランが，正午時点での潜在顧客を把握するために，逆 Top-k 検索を行うことが考えられる．そのため，各クエリの解を高速に検索できることが望ましい．

そこで，ART クエリの解を高速に検索する手法を提案する．さらに，複数の ART クエリが与えられたとき，重複する処理を削減し高速に処理を行うため，バッチ処理およびマルチコア環境でバッチ処理を行う手法を提案する．

以下では，4.2 節で，本章で扱う問題を詳細に定義する．4.3 節で ART クエリに対する処理手法について述べ，4.4 節で ART クエリの集合に対するバッチ処理手法について述べた後，4.5 節で評価実験の結果を示す．4.6 節で関連研究について述べ，最後に 4.7 節で本章のまとめを行う．

4.2　問題定義

本節では，まず，位置およびキーワードに基づくデータおよびユーザがシステムに登録するサブスクリプションを定義する．そして，あるデータが各サブスクリプションに対して有用であるかどうか判断するため，サブスクリプションに対して有用な k 個のデータとして Top-k データを定義する．その上で，位置およびキーワードに基づく逆 Top-k クエリおよび近似逆 Top-k クエリを新たに定義する．最後に，本章で扱う問題を詳細に定義する

まず，位置およびキーワードに基づくデータおよびサブスクリプションを定義する．

定義 4.1 (データ)．ある PoI が生成するデータは，$o = (l, \mathcal{W})$ と定義される．ここで，$o.l$ は緯度と経度によって表される 2 次元平面上のデータ o の位置，$o.\mathcal{W}$ はデータ o が持つキーワードの集合である．

定義 4.2 (サブスクリプション). あるユーザの発行するサブスクリプションは，$s = (l, \mathcal{W})$ と定義される．ここで，$s.l$ はサブスクリプション s の位置，$s.\mathcal{W}$ はサブスクリプション s が持つキーワードの集合 $(1 \leq |s.\mathcal{W}| \leq \mathcal{W}_{max})$ である．

データの集合を O，サブスクリプションの集合を S とし，あるサブスクリプションに対する Top-k データを定義する．

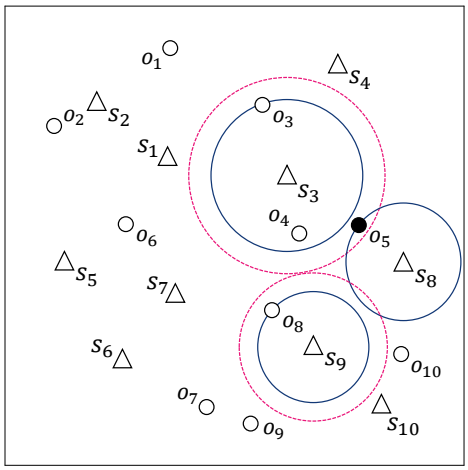
定義 4.3 (Top-k データ). データの集合 O が与えられたとき，あるサブスクリプション $s \in S$ の Top-k データ T は次の条件を満たす．(i) $T \subseteq O_{s.\mathcal{W}}$，(ii) $|T| = k$，(iii) $\forall o \in T, \forall o' \in O_{s.\mathcal{W}} \backslash T, Edist(s.l, o.l) \leq Edist(s.l, o'.l)$．ここで，$O_{s.\mathcal{W}}$ は，O に含まれるデータの中で $s.\mathcal{W}$ に含まれるキーワードを 1 つ以上含むデータの集合である．

次に，位置およびキーワードに基づく逆 Top-k クエリおよび近似逆 Top-k クエリを新たに定義する．以降，これらをそれぞれ RT（**R**everse spatio-textual **T**op-k）クエリおよび ART（**A**pproximate **R**everse spatio-textual **T**op-k）クエリと呼ぶ．

定義 4.4 (RT クエリ). データ集合 O およびサブスクリプション集合 S が与えられたとき，RT クエリ $q = \{o_q, k\}$（$o_q \in O$）は，クエリオブジェクト o_q を Top-k データに含むサブスクリプションを検索する．つまり，q の解であるサブスクリプションの集合 RT は次の条件を満たす．$q.RT = \{s \in S | o_q \in s.T\}$．

定義 4.5 (ART クエリ). データ集合 O およびサブスクリプション集合 S が与えられたとき，ART クエリ $q = \{o_q, k, \delta\}$（$o_q \in O, \delta \geq 1$）の解であるサブスクリプションの集合を ART とする．RT クエリ $q' = \{o_q, k\}$ の解であるサブスクリプションの集合を RT とすると，各サブスクリプション $s \in S$ は，以下の 3 つの場合に分類される．

1. $s \in RT$ ならば，$s \in ART$ である．
2. $s \notin RT$ かつ $Edist(s.l, o_q.l) \leq \delta \cdot Edist(s.l, o_k.l)$ ならば，s は ART に含まれてもよい．
3. $s \notin RT$ かつ $\delta \cdot Edist(s.l, o_k.l) < Edist(s.l, o_q.l)$ ならば，$s \notin ART$ である．



△: サブスクリプション　○ : データ

s	キーワード
s_1	w_1, w_2
s_2	w_1
s_3	w_2, w_3
s_4	w_1, w_4, w_5, w_6
s_5	w_1, w_5
s_6	w_1, w_6
s_7	w_1, w_4, w_5
s_8	w_2
s_9	w_1, w_3
s_{10}	w_1

o	キーワード
o_1	w_1, w_4
o_2	w_1, w_2, w_3
o_3	w_1, w_2
o_4	w_1, w_4, w_5
o_5	w_2, w_3
o_6	w_3, w_6, w_7
o_7	w_1, w_2
o_8	w_1, w_3
o_9	w_1, w_2, w_3, w_5
o_{10}	w_1, w_7, w_8

図 4.1: サブスクリプションおよびデータの位置およびキーワードの例

ここで，o_k は，s に対する上位 k 番目のデータである．

今後，特に明記する必要がない場合，$q.k$ および $q.\delta$ をそれぞれ k および δ と表記する．

例 4.1. 図 4.1 は，本章を通して用いる例を示している．この例では，10 個のサブスクリプション $\{s_1, \cdots, s_{10}\}$ および 10 個のデータ $\{o_1, \cdots, o_{10}\}$ が存在している．このとき，ART クエリ $q = \{o_5, 1, \delta\}$ が与えられたとする．まず，o_5 は s_8 の Top-1 データであるため，s_8 は $q.ART$ に含まれる．次に，s_3 および s_9 の Top-1 データはそれぞれ o_3 および o_8 である．しかし，o_5 は $\delta \cdot Edist(s_3.l, o_3.l)$ の範囲内（点線の範囲内）に含まれているため，s_3 は $q.ART$ に含まれてもよい．一方，$\delta \cdot Edist(s_9.l, o_8.l)$ の範囲内には含まれないため，s_9 は $q.ART$ に含まれない．

本章では，4.3 節において問題 1，4.4 節において問題 2 に取り組む．

問題定義 1. データ集合 O，サブスクリプション集合 S，および ART クエリ q が与えられたとき，q の解を高速に計算する．

問題定義 2. データ集合 O，サブスクリプション集合 S，および ART クエリの集合 Q が与えられたとき，各 ART クエリ $q \in Q$ の解を高速に計算する．

4.3 ART クエリ処理手法

本節では，ART クエリの処理手法について述べる．

4.3.1 ベースラインアルゴリズム

本章は，ART クエリの解を求める問題に初めて取り組む．そのため，本項では，ベースラインとなるアルゴリズム BL について述べる．以下では，RT クエリの解を単純に検索する手法について述べた後，その手法を ART クエリに適用する方法について説明する．

まず，RT クエリの解を単純に検索する手法を考える．ある RT クエリ $q = (o_q, k)$ が与えられたとき，解となり得るサブスクリプションは，$o_q.\mathcal{W}$ に含まれるキーワードを少なくとも 1 つ含むすべてのサブスクリプションの集合 $S_{o_q.\mathcal{W}}$ である．ここで，q の解を計算する際，$s \in S_{o_q.\mathcal{W}}$ の正確な Top-k データを求める必要はなく，o_q が s の Top-k データに含まれるかどうかのみ判断できれば良い．o_q が s の Top-k データに含まれないと判断された時点で，s は q の解に含まれないと判断できるため，s のチェックを終了できる．具体的には，s に対して o_q より近い位置に存在するデータが k 個以上発見された時点で，o_q は s の Top-k データに含まれないと判断できる

これを実現するため，s の Top-k データになり得るデータ集合の中から，o_q より s に近い位置に存在するデータの集合 O^+ を検索する．s の Top-k データになり得るデータは，$s.\mathcal{W}$ に含まれるキーワードを少なくとも 1 つ含むデータの集合 $O_{s.\mathcal{W}}$ に含まれるデータである．そのため，$O_{s.\mathcal{W}}$ に含まれるデータの中で，$s.l$ から距離 $Edist(s.l, o_q.l)$ 以内の範囲に含まれるデータが O^+ である．そして，$|O^+|$ が k を超えたとき，s は q の解に含まれないと判断できるため，s のチェックを終了できる．O^+ に含まれるデータを効率的に検索可能するため，ベースライン手法では，各キーワード w に対して，w を含むデータの位置情報を Kd-tree[4] を用いて管理する．

アルゴリズム 7 は，BL の詳細を示す．各サブスクリプション $s \in S_{o_q.\mathcal{W}}$ に対して，$s.l$ から距離 $Edist(s.l, o_q.l)$ 以内の範囲に含まれるデータの集合 O^+ を検索す

Algorithm 7: BL

Input: $q = (o_q, k)$

1 $RT \leftarrow \varnothing$
2 **for** $\forall s \in S_{o_q.\mathcal{W}}$ **do**
3 $\quad O^+ \leftarrow \varnothing$
4 $\quad$ **for** $\forall w \in s.\mathcal{W}$ **do**
5 $\quad\quad o \leftarrow$GetNearestData$(w, s.l)$
6 $\quad\quad$ **while** $|O^+| < k \wedge Edist(s.l, o.l) < Edist(s.l, o_q.l)$ **do**
7 $\quad\quad\quad O^+ \leftarrow O^+ \cup \{o\}$
8 $\quad\quad\quad o \leftarrow$GetNextData$(w, s.l, o)$
9 $\quad\quad$ **if** $|O^+| \geq k$ **then**
10 $\quad\quad\quad$ **break**
11 $\quad$ **if** $|O^+| < k$ **then**
12 $\quad\quad RT \leftarrow RT \cup \{s\}$
13 return RT

る．具体的には，各キーワード $w \in s.\mathcal{W}$ に対して，w を含むデータを s から近い順に評価する．まず，GetNearestData により，w を含むデータの中で s に最も近いデータ o を取得する（5 行）．$|O^+|$ が k 未満かつ o が o_q よりも s に近い場合，o を O^+ に加え，GetNextData により，w を含むデータの中で o の次に s に近いデータを取得する（6–8 行）．この操作を，$|O^+|$ が k 以上または条件を満たすデータが無くなるまで繰り返す．最後に，$|O^+|$ が k 未満であれば s を解 RT に加える（11–12 行）．

次に，BL を ART クエリ $q = (o_q, k, \delta)$ に適用する方法を考える．まず，定義 4.5 より，以下の定理が成り立つ．

定理 4.1. ART クエリ $q = (o_q, k, \delta)$ およびサブスクリプション $s \in S_{o_q.\mathcal{W}}$ が与えられたとき，$Edist(s.l, o_q.l) \leq \delta \cdot Edist(s.l, o.l)$ を満たすデータ $o \in O_{s.\mathcal{W}}$ は，s に対して q の解 ART の正確性に影響を与えない．

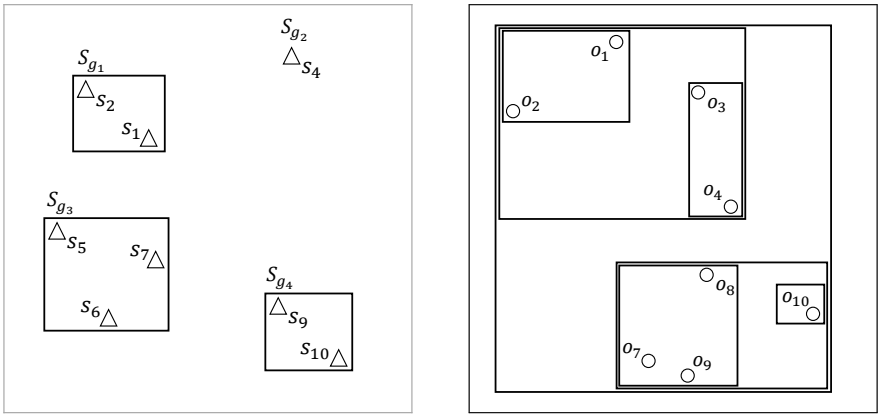
証明. $Edist(s.l, o_q.l) \leq \delta \cdot Edist(s.l, o.l)$ を満たすデータ $o \in O_{s.\mathcal{W}}$ が，$s.T$ に含まれる場合でも，定義 4.5 の条件（2）を満たすため，s は q の解 ART に含まれてもよい．よって，定理が成り立つ． □

定理 4.1 より，$\delta \cdot Edist(s.l, o.l) < Edist(s.l, o_q.l)$ を満たすデータ o の数が k 以上であるとき，s は ART に含まれず，k 未満であるとき，ART に含まれてもよい．つまり，$\delta \cdot Edist(s.l, o.l) < Edist(s.l, o_q.l)$ を満たすデータの数を把握することで，ART クエリの解 ART を求めることが可能である．よって，BL を ART クエリに適用するためには，アルゴリズム 7 の 6 行目において，$Edist(s.l, o.l) < Edist(s.l, o_q.l)$ ではなく，$\delta \cdot Edist(s.l, o.l) < Edist(s.l, o_q.l)$ を満たす場合のみそれ以降の処理を行う．

4.3.2 PART

本項では，ART クエリの解を高速に求める新たなアルゴリズム PART（**P**rocessing of **ART** クエリ）について述べる．RT クエリと同様に，ある ART クエリ $q = (o_q, k, \delta)$ が与えられたとき，解となり得るサブスクリプションは，$S_{o_q.\mathcal{W}}$ に含まれるすべてのサブスクリプションである．ここで，あるキーワード $w \in o_q.\mathcal{W}$ を含むサブスクリプション s_1 および s_2 の位置が近い場合，s_1 および s_2 は，同じデータが Top-k データになりやすい．

PART は，この特徴およびクエリオブジェクトが Top-k データに含まれるかどうかのみ判断できれば良いという 2 つの特徴に基づいて，ART クエリの解を高速に計算する．1 つ目の特徴を考慮し，PART では，同じデータが Top-k データになりやすいサブスクリプションのチェックを同時に行う．これを実現するため，サブスクリプション集合をキーワードごとに管理し，あるキーワード w を含むサブスクリプションの集合を互いに位置が近いもの同士で複数のグループに分割して管理する．このとき，w を含むサブスクリプションのグループの集合を $SG[w]$ とし，各サブスクリプションのグループ S_g は，そのグループに含まれるサブスクリプションの位置を包含する最小外接矩形 $S_g.mbr$ を保持する．PART は，分割されたサブスクリプションのグループごとにチェックを行う．2 つ目の特徴を考慮し，各サブスクリプションのグループ S_g に対して，BL と同様に，クエリオブジェクト o_q より近い位置にあるデータの集合を検索する．これを実現するため，文献 [86] で提案された Inverted linear Quad-tree のアイデアに基づき，データ集合をキー



(a) サブスクリプションのインデックス　　(b) データのインデックス

図 4.2: w_1 を含むサブスクリプションおよびデータ集合に対するインデックスの例

ワードごとに R 木を用いて管理する．

例 4.2. 図 4.1 に示すようにサブスクリプションおよびデータが存在するとき，キーワード w_1 を含むサブスクリプションの集合およびデータの集合は，図 4.2 に示すように管理される．サブスクリプション集合は 4 つのグループに分割されており，$SG[w_1] = \{S_{g_1}, S_{g_2}, S_{g_3}, S_{g_4}\}$ である．

以下では，提案したインデックスを用いたフィルタリング手法を説明し，その後，PART の詳細について述べる．

フィルタリング手法

PART では，ART クエリ $q = (o_q, k, \delta)$ が与えられたとき，キーワード $w \in o_q.\mathcal{W}$ を含む各サブスクリプションのグループ $S_g \in SG[w]$ に対して順に処理を行う．S_g に対する処理において，S_g に含まれるすべてのサブスクリプションに対して，o_q より近い位置に存在するデータの集合 $S_g.O^+$ を検索する．つまり，$\forall s \in S_g, \forall o \in S_g.O^+, Edist(s.l, o.l) < Edist(s.l, o_q.l)$ が成り立つ．そのため，$|S_g.O^+|$ が k 以上となった時点で，S_g に含まれるすべてのサブスクリプションは，$q.ART$ に含まれなくてよい．

ここで，あるサブスクリプションまたはサブスクリプションのグループとあるデータまたは R 木のノード間の距離の上界値および下界値を $Edist_{ub}(\cdot,\cdot)$ および $Edist_{lb}(\cdot,\cdot)$ とし，n を根とする部分木で管理されるデータの集合を O_n とする．このとき，あるサブスクリプションのグループ $S_g \in SG[w]$ および w に対する R 木のあるノード n に対して，2 つの定理が成り立つ．

まず，以下の定理が成り立つ．

定理 4.2. $Edist_{ub}(S_g.mbr, o_q.l) < \delta \cdot Edist_{lb}(S_g.mbr, n)$ ならば，$\forall o \in O_n$ は $\forall s \in S_g$ に対して $q.ART$ の正確性に影響を及ぼさない．

証明. $\forall s \in S_g, \forall o \in O_n, Edist_{lb}(S_g.mbr, n) \leq Edist(s.l, o.l)$ かつ $\forall s \in S_g$, $Edist(s.l, o_q.l) \leq Edist_{ub}(S_g.mbr, o_q.l)$ より，$Edist_{ub}(S_g.mbr, o_q.l) < \delta \cdot Edist_{lb}(S_g.mbr, n)$ ならば，$\forall s \in S_g, \forall o \in O_n, Edist(s.l, o_q.l) < \delta \cdot Edist(s.l, o.l)$ である．定理 4.1 より，定理が成り立つ． □

定理 4.2 より，$Edist_{ub}(S_g.mbr, o_q.l) < \delta \cdot Edist_{lb}(S_g.mbr, n)$ ならば，O_n に含まれるすべてのデータは $S_g.O^+$ に含まれなくてもよい．そのため，ノード n を根とする部分木をフィルタリングできる．

次に，以下の定理が成り立つ．

定理 4.3. $Edist_{ub}(S_g.mbr, n) < Edist_{lb}(S_g.mbr, o_q.l)$ ならば，$\forall o \in O_n, o \in S_g.O^+$ である．

証明. $\forall s \in S_g, \forall o \in O_n, Edist(s.l, o.l) \leq Edist_{ub}(S_g.mbr, n)$ かつ $\forall s \in S_g$, $Edist_{lb}(S_g.mbr, o_q.l) \leq Edist(s.l, o_q.l)$ より，$\forall s \in S_g, \forall o \in O_n, Edist(s.l, o.l) < Edist(s.l, o_q.l)$ である．よって，定理が成り立つ． □

定理 4.2 より，$Edist_{ub}(S_g.mbr, o_q.l) < \delta \cdot Edist_{lb}(S_g.mbr, n)$ ならば，O_n に含まれるすべてのデータを $S_g.O^+$ に追加できる．そのため，ノード n を根とする部分木をチェックする必要はない．これらの定理に基づいて R 木をチェックすることで，$S_g.O^+$ に含まれるデータを高速に検索できる．

Algorithm 8: PART

Input: q

1 $ART \leftarrow \varnothing$
2 **for** $\forall w \in o_q.\mathcal{W}$ **do**
3 　**for** $\forall S_g \in SG[w]$ **do**
4 　　$N_c, O^+ \leftarrow$ GetCandidates(q, w, S_g)
5 　　**if** $|O^+| < q.k$ **then**
6 　　$ART \leftarrow ART \cup$ Verification(q, w, S_g, N_c, O^+)
7 return ART

アルゴリズム

アルゴリズム 8 は PART の概要を示している．ある ART クエリ q が与えられたとき，キーワード $w \in o_q.\mathcal{W}$ を含むすべてのサブスクリプションのグループ $S_g \in SG[w]$ に対して以下の処理を行う．まず，GetCandidates により，w を含むデータの中で，o_q より S_g に近い位置に存在するデータの集合 O^+ および o_q より S_g に近い位置に存在するデータを含む可能性のあるノードの集合 N_c を検索する．その後，$|O^+|$ が k 未満である場合，Verification により，S_g に含まれるサブスクリプション s の中で，$q.ART$ に含まれるサブスクリプションを得る．

以下では，GetCandidates および Verification の詳細について述べる．

GetCandidates. アルゴリズム 9 は，キーワード w を含むあるサブスクリプションまたはサブスクリプションのグループ $\mathcal{S}$ に対して，w を含むデータの中で，o_q より近い位置に存在するデータの集合 O^+ および o_q より近い位置に存在するデータを含む可能性のあるノードの集合 N_c を検索するアルゴリズムを示している．w に対する R 木の根ノードから順に以下の操作を実行する．チェックを行うノード n が葉ノードである場合，n を N_c に加える（5–6 行）．一方，中継ノードである場合，n の子ノード n' に対して以下の処理を行う．$Edist_{ub}(\mathcal{S}.mbr, n) < Edist_{lb}(\mathcal{S}.mbr, o_q.l)$ である場合，定理 4.3 より，$\forall o \in O_{n'}$ は $\forall s \in \mathcal{S}$ に対して o_q より近い位置に存在するため，O^+ に追加する．さらに，$|O^+|$ が k 以上となった場合，$\mathcal{S}$ は，q の解に含まれなくてもよいためチェックを終了する（9–13 行）．ここで，$\mathcal{S}.mbr$ は，$\mathcal{S}$ がサ

Algorithm 9: GetCandidates$(q, w, \mathcal{S})$

Input: $q = (o_q, k, c), w, \mathcal{S}$ // a subscription or a group of subscriptions

1 $O^+ \leftarrow \varnothing, N_c \leftarrow \varnothing, H \leftarrow \varnothing$
2 Push root node of R-tree for w into H
3 **while** $H \neq \varnothing$ **do**
4 $\quad n \leftarrow$ the node popped from H
5 $\quad$ **if** *n is a leaf node* **then**
6 $\quad\quad N_c \leftarrow N_c \cup \{n\}$
7 $\quad$ **else**
8 $\quad\quad$ **for** $\forall n' \in N$ *//N is n's children* **do**
9 $\quad\quad\quad$ **if** $Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{lb}(\mathcal{S}.mbr, o_q.l)$ **then**
10 $\quad\quad\quad\quad$ **for** $\forall o \in O_{n'}$ **do**
11 $\quad\quad\quad\quad\quad O^+ \leftarrow O^+ \cup \{o\}$
12 $\quad\quad\quad\quad$ **if** $|O^+| \geq k$ **then**
13 $\quad\quad\quad\quad\quad$ return O^+, N_c
14 $\quad\quad\quad$ **else**
15 $\quad\quad\quad\quad$ **if** $\delta \cdot Edist_{lb}(\mathcal{S}.mbr, n') \leq Edist_{ub}(\mathcal{S}.mbr, o_q.l)$ **then**
16 $\quad\quad\quad\quad\quad$ Push n' into H
17 return O^+, N_c

ブスクリプションのグループ S_g ならば $S_g.mbr$，サブスクリプション s ならば $s.l$ を表す．一方，$\delta \cdot Edist_{lb}(\mathcal{S}.mbr, n') > d_{ub}(\mathcal{S}.mbr, o_q.l)$ である場合，定理 4.2 より，n' を根とする部分木に含まれるすべてのデータは q の解に影響を及ぼさないためチェックを行う必要はない．そこで，$\delta \cdot Edist_{lb}(\mathcal{S}.mbr, n') \leq Edist_{ub}(\mathcal{S}.mbr, o_q.l)$ を満たす場合のみ，再帰的に処理を行うため，n' を H に挿入する（14–16 行）．

Verification．アルゴリズム 10 は，S_g に含まれるサブスクリプション s の中で，ART に含まれるサブスクリプションを検索するアルゴリズムを示している．S_g に含まれる各サブスクリプション s に対して以下の処理を実行する．まず，S_g に対して検索された O^+ および N_c から，GetCandidates と同様の処理を行い，$s.O^+$ および $s.N_c$ を得る（3–10 行）．その後，w を除く s の各キーワード $w' \in s.\mathcal{W} \backslash \{w\}$ に対し

Algorithm 10: Verification(q, w, S_g, N_c, O^+)

```
Input: q, w, S_g, N_c, O^+
1  ART ← ∅
2  for ∀s ∈ S_g do
3      s.O^+ ← O^+, s.N_c ← ∅
4      for ∀n ∈ N_c do
5          if Edist_ub(s.l, n') < Edist_lb(s.l, o_q.l) then
6              for ∀o ∈ O_n' do
7                  s.O^+ ← s.O^+ ∪ {o}
8          else
9              if δ · Edist_lb(s.l, n') ≤ Edist_ub(s.l, o_q.l) then
10                 s.N_c ← s.N_c ∪ {n}
11     for ∀w' ∈ s.W\w do
12         if |s.O^+| ≥ k then
13             break
14         s.O^+, s.N_c ← s.O^+, s.N_c ∪ GetCandidates(q, w', s)
15     if |s.O^+| < k then
16         ART ← ART ∪ CheckSubscription(q, s, s.O^+, s.N_c)
17 return ART
```

て，GetCandidates を実行し，$s.O^+$ および $s.N_c$ を得る．このとき，$|s.O^+|$ が k 以上となった場合，s は $q.ART$ に含まれなくてもよいため処理を終了する（11–14 行）．最後に，各キーワードに対する処理が終了した時点で，$|s.O^+|$ が k 未満である場合，s は ART に含まれる可能性があるため，CheckSubscription により，s が ART に含まれるかどうかチェックする（15–16 行）．ここで，CheckSubscription は，$s.O^+$ および $s.N_c$ に含まれるすべてのデータの中で s に対して o_q より近い位置に存在するデータの数が，k 未満であれば s を返し，k 以上であれば $NULL$ を返す関数である．

ここで，PART は $|s.O^+|$ が k 未満となるサブスクリプション s をクエリの解とするため，$o_q \in s.T$ である s は必ず解に含まれる．また，δ が大きくなると，アル

ゴリズム 9 の 15 行目およびアルゴリズム 10 の 9 行目の条件を満たすノードの数が減少し，それ以降の処理を行う回数が減少する．そのため，PART は，δ が大きいほど，より高速に処理を行うことが可能である．

4.4 バッチ処理手法

本節では，ART クエリの集合に対するバッチ処理手法について説明する．

4.4.1 B-PART

本項では，与えられた ART クエリの集合 Q に対して，各クエリの解を高速に求める新たなアルゴリズム B-PART（**B**atch **P**rocessing of **ART** クエリ）について述べる．ここで，Q に含まれるクエリ q_1 および q_2 に対して，$o_{q_1}.\mathcal{W}$ および $o_{q_2}.\mathcal{W}$ が共通するキーワード w を含んでいると仮定する．このとき，PART を単に適用すると，Q に含まれる各 ART クエリの解を順に計算するため，w を含むサブスクリプションのグループ $S_g \in SG[w]$ に対して，複数回チェックを行ってしまう．B-PART では，複数のクエリに共通するサブスクリプションのグループのチェックを同時に行うことにより，実行時間の削減を図る．これを実現するため，ART クエリの集合 Q が与えられたとき，Q に含まれるすべてのクエリオブジェクト o_q が持つキーワードの集合 $\mathcal{W}_Q$ を計算する．そして，各キーワード $w \in \mathcal{W}_Q$ に対する処理を順に行い，w を含む各サブスクリプションのグループを 1 度だけチェックする．

アルゴリズム 11 は B-PART の概要を示している．まず，与えられた ART クエリの集合 Q に対して，Q に含まれるすべてのクエリオブジェクト o_q が持つキーワードの集合 $\mathcal{W}_Q$ および各キーワード $w \in \mathcal{W}_q$ を含むクエリの集合 Q_w を計算する．そして，各キーワード $w \in \mathcal{W}_Q$ に対して順に処理を行う．具体的には，w を含むすべてのサブスクリプションのグループ $S_g \in SG[w]$ に対して以下の処理を行う．まず，BatchGetCandidates により，w を含むデータの中で，Q_w に含まれるすべてのクエリオブジェクトより S_g に近い位置に存在するデータの集合 O^+ および

Algorithm 11: B-PART

Input: Q

1 $W_Q \leftarrow \varnothing, Q. \leftarrow \varnothing$
2 **for** $\forall q \in Q$ **do**
3 　$q.ART \leftarrow \varnothing$
4 　**for** $\forall w \in o_q.\mathcal{W}$ **do**
5 　　$\mathcal{W}_Q \leftarrow w, Q_w \leftarrow q$
6 **for** $\forall w \in \mathcal{W}_Q$ **do**
7 　**for** $\forall S_g \in SG[w]$ **do**
8 　　$N_c, O^+ \leftarrow$ BatchGetCandidates(Q_w, w, S_g)
9 　　**if** $|Q_w| > 0 \wedge |O^+| < k_{max}$ $//k_{max} = \max\{q.k | q \in Q_w\}$ **then**
10 　　$Q_w.ART \leftarrow Q_w.ART \cup$ BatchVerification(Q_w, w, S_g, N_c, O^+)
11 return each $q.ART$

S_g に近い位置に存在するデータを含む可能性のあるノードの集合 N_c を検索する．その後，$|O^+|$ が k_{max} 未満である場合，BatchVerification により，S_g に含まれるサブスクリプション s の中で，Q_w に含まれる各クエリ q の解 $q.ART$ に含まれるサブスクリプションを得る．ここで，k_{max} は，Q_w に含まれる各クエリ q の $q.k$ の中で最大の値であり，$Q_w.ART$ は，$q.ART$（$q \in Q_w$）の集合を表す．

以下では，BatchGetCandidates および BatchVerification の詳細について述べる．

BatchGetCandidates．アルゴリズム 12 は，w を含むデータの中で，Q_w に含まれるすべてのクエリオブジェクトに対して，$\mathcal{S}$ に近い位置に存在するデータの集合 O^+ および $\mathcal{S}$ に近い位置に存在するデータを含む可能性のあるノードの集合 N_c を検索するアルゴリズムを示している．GetCandidates と同様に，w に対する R 木の根ノードから順に操作を実行する．ある子ノード n' に対する処理では，まず，$Edist_{ub}(\mathcal{S}.mbr, n')$ と $Edist_{min}$ を比較する（12 行）．ここで，$Edist_{min}$ は，Q_w に含まれる各クエリオブジェクト o_q に対する $Edist_{lb}(\mathcal{S}.mbr, o_q.l)$ の最小値であるため，$Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{min}$ ならば，$\forall q \in Q_w, Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{lb}(\mathcal{S}.mbr, o_q.l)$ が成り立つ．よって，$O_{n'}$ に含まれるすべてのデータを O^+ に追加する．さらに，$k_{max} = \max\{q.k | q \in Q_w\}$ であるため，$|O^+|$ が k_{max} 以上なら

Algorithm 12: BatchGetCandidates($Q_w, w, \mathcal{S}$)

Input: $Q_w, w, \mathcal{S}$

1 Sort Q_w in descending order of $Edist_{ub}(\mathcal{S}.mbr, o_q.l)$
2 $O^+ \leftarrow \varnothing, N_c \leftarrow \varnothing, H \leftarrow \varnothing, k_{max} \leftarrow \max\{q.k | q \in Q_w\}, \delta_{min} \leftarrow \min\{q.\delta | q \in Q_w\}$
3 $Edist_{min} \leftarrow \min\{Edist_{lb}(\mathcal{S}.mbr, o_q.l) | q \in Q_w\}$
4 Push root node of R-tree for w into H
5 $q' \leftarrow$ the first query of Q_w
6 **while** $H \neq \varnothing$ **do**
7 $\quad n \leftarrow$ the node popped from H
8 $\quad$ **if** *n is a leaf node* **then**
9 $\quad\quad N_c \leftarrow N_c \cup \{n\}$
10 $\quad$ **else**
11 $\quad\quad$ **for** $\forall n' \in N$ *//N is n's children* **do**
12 $\quad\quad\quad$ **if** $Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{min}$ **then**
13 $\quad\quad\quad\quad$ **for** $\forall o \in O_{n'}$ **do**
14 $\quad\quad\quad\quad\quad O^+ \leftarrow O^+ \cup \{o\}$
15 $\quad\quad\quad\quad$ **if** $|O^+| \geq k_{max}$ **then**
16 $\quad\quad\quad\quad\quad$ return O^+, N_c
17 $\quad\quad\quad$ **else**
18 $\quad\quad\quad\quad$ **while** $|O^+ \cup O_{n'}| \geq q'.k \wedge Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{lb}(\mathcal{S}.mbr, o_{q'}.l)$ **do**
19 $\quad\quad\quad\quad\quad Q_w \leftarrow Q_w \backslash \{q'\}$
20 $\quad\quad\quad\quad\quad$ **if** $Q_w = \varnothing$ **then**
21 $\quad\quad\quad\quad\quad\quad$ return O^+, N_c
22 $\quad\quad\quad\quad\quad q' \leftarrow$ the first query of Q_w
23 $\quad\quad\quad\quad$ **if** $\delta_{min} \cdot Edist_{lb}(\mathcal{S}.mbr, n') \leq Edist_{ub}(\mathcal{S}.mbr, o_{q'}.l)$ **then**
24 $\quad\quad\quad\quad\quad$ Push n' into H
25 return O^+, N_c

ば，$\mathcal{S}$ は，$\forall q \in Q_w$ の解に含まれなくてもよいためチェックを終了する（15–16 行）．

一方，$Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{min}$ でない場合，以下の処理を行う．Q_w は $Edist_{ub}(\mathcal{S}.mbr, o_q.l)$ の降順にソートされているため，先頭のクエリ q' ほど，$\mathcal{S}$ に対し

て，$o_{q'}$ より近い位置に存在するデータが多く存在する．そこで，$|O^+ \cup O_{n'}| \geq q'.k$ である場合，$Edist_{ub}(\mathcal{S}.mbr, n')$ と $Edist_{lb}(\mathcal{S}.mbr, o_{q'}.l)$ を比較する．$Edist_{ub}(\mathcal{S}.mbr, n') < Edist_{lb}(\mathcal{S}.mbr, o_{q'}.l)$ ならば，$o_{q'}$ より近い位置に存在するデータが $q'.k$ 個以上存在するため，$\mathcal{S}$ は q' の解に含まれなくてもよい．そのため，q' を Q_w から取り除き，次のクエリに対して同様の処理を行う（18–22 行）．一方，$|O^+ \cup O_{n'}| \geq q'.k$ でない場合，$\delta_{min} \cdot Edist_{lb}(\mathcal{S}.mbr, n') \leq Edist_{ub}(\mathcal{S}.mbr, o_{q'}.l)$ ならば，n' を H に挿入する（23–24 行）．ここで，$\delta_{min} = \min\{q.\delta | q \in Q_w\}$ である．

BatchVerification．アルゴリズム 13 は，S_g に含まれるサブスクリプション s の中で，$Q_w.ART$ に含まれるサブスクリプションを検索するアルゴリズムを示している．Verification と同様に，S_g に含まれる各サブスクリプション s に対して処理を実行する．$s.O^+$ および $s.N_c$ を得る際，$Edist_{min}$，$Edist_{max}$，および δ_{min} に基づいて処理を行う（7–13 行）．w を除く s の各キーワード $w' \in s.\mathcal{W} \backslash \{w\}$ に対する処理が終了した後，$|s.O^+|$ が k_{max} 未満である場合，Q_w に含まれる各クエリに対して，CheckSubscription を実行する（18–20 行）．

4.4.2 PB-PART

本項では，与えられた ART クエリの集合に対する処理をより高速に行うため，B-PART をマルチコア環境で行うアルゴリズム PB-PART（**P**arallel **B**atch **P**rocessing of **ART** クエリ）について述べ，効率的に並列処理を行うために各コアに対して負荷分散を行う方法について議論する．アルゴリズム 11 の 6 行目以降，B-PART は，各キーワード $w \in \mathcal{W}_Q$ に対して順に処理を行う．これらの処理は互いに独立しているため，並列して行うことが可能である．各キーワード w に対する処理では，w を含むサブスクリプションのグループに対して，クエリオブジェクトより近い位置に存在するデータを検索する．そのため，各キーワード w に対する処理時間はそのキーワードを含むデータの分布に依存する．w が多くのデータに含まれる場合，または w を含むデータがある領域に密集している場合，処理時間が大きくなる．w が多くのデータに含まれる場合，検索するデータの数が単純に増加するため，処理時間は大きくなる．w を含むデータがある領域に密集している場

Algorithm 13: BatchVerification(Q_w, w, S_g, N_c, O^+)

Input: Q_w, w, S_g, N_c, O^+

1 $Q_w.ART \leftarrow \varnothing$
2 $k_{max} \leftarrow \max\{q.k | q \in Q_w\}, \delta_{min} \leftarrow \min\{q.\delta | q \in Q_w\}$
3 **for** $\forall s \in S_g$ **do**
4 $\quad s.O^+ \leftarrow O^+, s.N_c \leftarrow \varnothing$
5 $\quad Edist_{min} \leftarrow \min\{Edist(s.l, o_q.l) | q \in Q_w\}$
6 $\quad Edist_{max} \leftarrow \max\{Edist(s.l, o_q.l) | q \in Q_w\}$
7 $\quad$ **for** $\forall n \in N_c$ **do**
8 $\quad\quad$ **if** $Edist_{ub}(s.l, n') < Edist_{min}$ **then**
9 $\quad\quad\quad$ **for** $\forall o \in O_{n'}$ **do**
10 $\quad\quad\quad\quad s.O^+ \leftarrow s.O^+ \cup \{o\}$
11 $\quad\quad$ **else**
12 $\quad\quad\quad$ **if** $\delta_{min} \cdot Edist_{lb}(s.l, n') \leq Edist_{max}$ **then**
13 $\quad\quad\quad\quad s.N_c \leftarrow s.N_c \cup \{n\}$
14 $\quad$ **for** $\forall w' \in s.\mathcal{W} \backslash w$ **do**
15 $\quad\quad$ **if** $|s.O^+| \geq k_{max}$ **then**
16 $\quad\quad\quad$ **break**
17 $\quad\quad s.O^+, s.N_c \leftarrow s.O^+, s.N_c \cup$ BatchGetCandidates(Q_w, w', s)
18 $\quad$ **if** $|s.O^+| < k_{max}$ **then**
19 $\quad\quad$ **for** $\forall q \in Q_w$ **do**
20 $\quad\quad\quad q.ART \leftarrow q.ART \cup$ CheckSubscription$(q, s, s.O^+, s.N_c)$
21 return $Q_w.ART$

合，提案手法における R 木では，データが多く存在する領域を再帰的に分割するため，データが密集している領域ほど木の深さが深くなる．その結果，その領域を検索するとき，処理時間が大きくなる．

これらの特徴の特徴を考慮し，各コアにおける処理時間が均一になるように，キーワードを分散する手法を提案する．以下では，各キーワード w に対する処理時間を予測するため，w に対する処理のコスト $cost(w)$ を定義する．そして，コストに基づいて，各キーワードを処理するコアを決定する方法について説明する．

まず，w に対する処理のコスト $cost(w)$ を以下のように定義する．

$$cost(w) = \sum_{\forall n \in LN_w} \phi^{n.depth} \cdot |O_n| \tag{4.1}$$

ここで，LN_w は w に対する R 木に含まれる葉ノードの集合，$n.depth$ はノード n の深さ，および ϕ はノードの深さに対する重みである．

式（4.1）により計算されるコストに基づき，各キーワードを処理するコアを決定する方法を説明する．まず，以下の定理が成り立つ．

定理 4.4. $\mathcal{W}_Q$ が与えられたとき，各コア間の処理時間の差を最小化するキーワードの分散を決定することは，NP 困難である．

証明. $\mathcal{W}_Q$ に含まれる各キーワード w のコアへの最適な分散を決定することは，以下の問題を解くことと同値である．m 個の整数 $a_1, a_2, \cdots, a_m$ を考える．これらを，互いに素な $|\mathcal{T}|$ 個の集合に分割したとき，各集合に含まれる整数の和を $\sum(t_i)$ とする．ここで，t_i は i 番目のコアを表す．このとき，$\max_{t_i \in \mathcal{T}} \sum(t_i) - \min_{t_i \in \mathcal{T}} \sum(t_i)$ を最小化する．この問題は，multi-way number partitioning と呼ばれ，NP 困難な問題である [37]． □

最適なキーワードの分散を決定することは NP 困難であるため，PB-PART では，$\mathcal{W}_Q$ が与えられたとき，貪欲法を用いて分散を決定する．

4.5　評価実験

本節では，提案手法および拡張手法の性能評価のために行った実験の結果について述べる．提案手法の有効性を検証するため，シングルコア環境において，クエリの指定する k の最大値，サブスクリプションの数，データの数，クエリの指定する近似率，および一度に与えられるクエリの数に対するスケーラビリティを評価した．拡張手法の有効性を検証するため，マルチコア環境において，スレッドの数に対するスケーラビリティを評価した．

本実験は，Windows 10 Pro，36 core Intel Xeon Gold 6154 (3.0GHz)，および 512GB 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．

コンパイルには最適化オプションとして-O2 を使用し，マルチコア処理を行うため，OpenMP を用いた．

4.5.1 評価環境

本実験では，シングルコア環境での性能を評価するため，以下の手法の性能を評価した．

- **B-PART.** 4.4.1 項で提案した手法.
- **PART.** 4.3.2 項で提案した手法.
- **BL.** 4.3.1 項で述べた手法.
- **RG.** 文献 [89] で提案された手法を本問題に適用したもの．この手法を本問題に適用した場合，データ集合を 1 つの IR 木 [43] で管理する．そして，ART クエリ $q = \{o_q, k, \delta\}$ が与えられたとき，o_q が Top-k データになり得るすべてのサブスクリプションに対して，Top-k 検索を実行する．Top-k 検索の途中で，o_q を Top-k データに含む可能性が無いと判断された時点で検索を終了する．
- **SF-SEP.** 文献 [15] で提案された手法を本問題に適用したもの．この手法を本問題に適用した場合，データ集合を 1 つの IR 木で管理する．そして，ART クエリ $q = \{o_q, k, \delta\}$ が与えられたとき，o_q が Top-k データになり得るすべてのサブスクリプションに対して，Top-k 検索を実行する．

また，マルチコア環境での性能を評価するため，以下の手法の性能を評価した．

- **PB-PART.** 4.4.2 項で提案した手法.
- **PB-PART-naive.** 4.4.2 項で提案した手法において，計算された W_Q に含まれる各キーワードをハッシュパーティションして処理を行う手法.

表 4.1: パラメータの設定

パラメータ	値
$q.k$ の最大値，k_{max}	5, **10**, 15, 20, 25, 30
サブスクリプションの数，$\|S\|$ [$\times 10^6$]	**0.25**, 0.5, 0.75, 1
データの数，$\|O\|$ [$\times 10^6$]	**1**, 2, 3, 4, 5
近似率，$\overline{\delta}$	1, **1.5**, 2, 2.5, 3, 3.5, 4
一度に与えられるクエリの数，$\|Q\|$ [$\times 10^3$]	**1**, 2.5, 5, 7.5, 10
スレッドの数，$\|\mathcal{T}\|$	**1**, 3, 6, 9, 12, 15, 18

- **P-PART-naive.** 4.3.2 項で提案した手法において，与えられた Q に含まれるクエリをハッシュパーティションして処理を行う手法.

データセット. 本実験では，3 章の評価実験において用いたデータセットと同じものを用いて，同様の方法でサブスクリプションを生成した．このとき，サブスクリプションの位置は選ばれたデータの位置とした.

クエリ. ある ART クエリ q は，データ集合に含まれるデータをランダムに選びクエリオブジェクト o_q とする．また，$q.k$ は 1 から k_{max} の間の一様乱数とし，$s.\delta$ は $\overline{\delta}$ を平均とする正規分布に従うとする.

パラメータ. 表 4.1 により，本実験で用いたパラメータを示す．太字で表されている値はデフォルトの値である.

4.5.2 評価結果

本実験では，ART クエリの集合が 100 回与えられたときの各手法における平均計算時間（与えられた ART クエリの集合に対して，各クエリの解の計算が完了するまでの時間の平均値）の結果について述べる．B-PART を除いた各手法における各 ART クエリの平均処理時間は，平均計算時間を $|Q|$ で割ったものである.

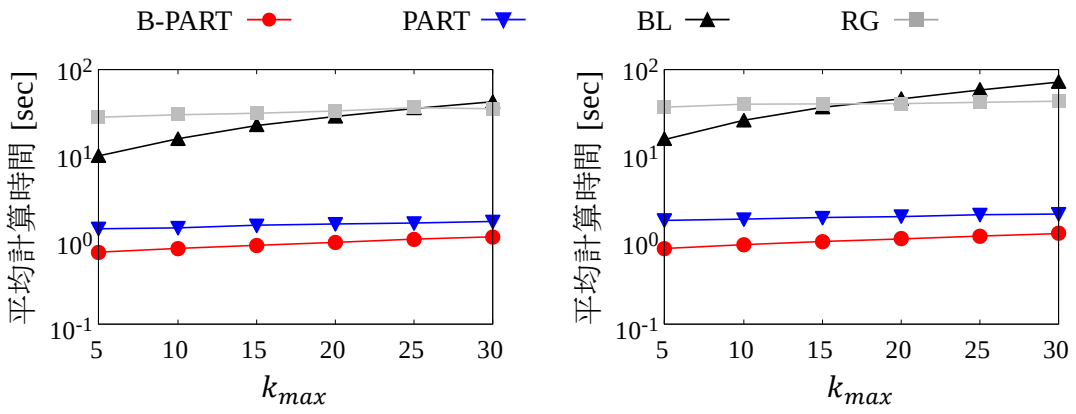
表 4.2: 各手法における平均計算時間 [sec]

手法	*TWEETS*	*PLACES*
SF-SEP	8094.601	9750.082
RG	30.168	38.185
BL	15.516	25.762
PART	1.453	1.748
B-PART	0.777	0.856

シングルコア処理

各手法の性能の比較． 表 4.2 に，各手法における平均計算時間の結果を示す．使用したデータセットによらず，すべての手法の中で，提案手法（PART および B-PART）は高速に解を計算できることがわかる．また，各サブスクリプションのグループに対する重複するチェックを削減するため，B-PART は PART と比べて高速に解を計算できる．ここで，*TWEETS* を使用した場合よりも *PLACES* を使用した場合の方が平均計算時間が大きくなる．これは，*TWEETS* に比べて *PLACES* の方がキーワードの種類が少なく，各データおよびサブスクリプションが同じキーワードを含みやすくなる．その結果，与えられたクエリに対して，チェックを行うサブスクリプションの数および各サブスクリプションのチェックにおいて評価するデータの数が多くなるため，平均計算時間が大きくなる．さらに，SF-SEP は，チェックを行うすべてのサブスクリプションに対して正確な Top-k データを計算するため，他の手法と比較して，平均計算時間が非常に大きくなる．各パラメータの影響を評価した実験では，SF-SEP との比較は行わず，その他の 4 つの手法の比較を行った．

k_{max} の影響． 図 4.3 にクエリの指定する k の最大値 k_{max} を変えたときの結果を示す．提案手法は，RG および BL と比較して，常に高速に解を計算できることがわかる．また，k_{max} の増加に伴い，各手法において平均計算時間が増加するが，提案手法の平均計算時間の増加率の方が小さく，k_{max} が大きい場合，提案手法はより有用であることがわかる．さらに，BL は k_{max} の増加による影響が最も大きい



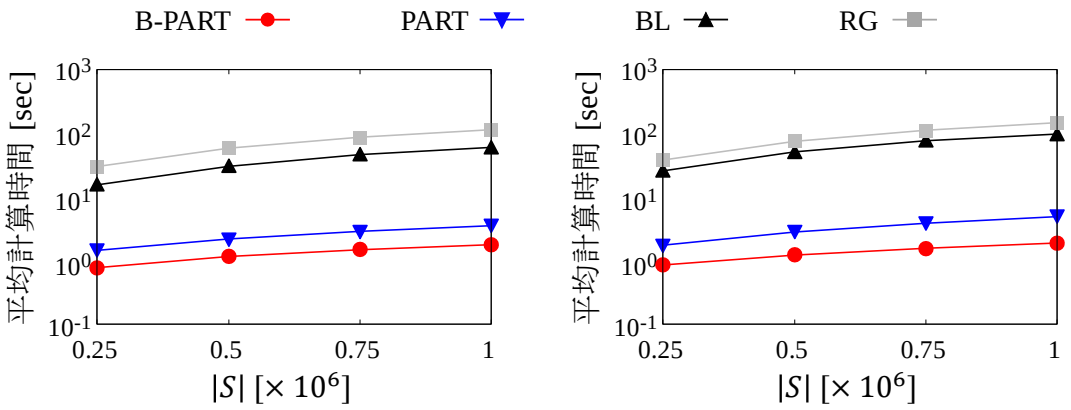
(a) *TWEETS*　　(b) *PLACES*

図 4.3: k_{max} の影響

ことがわかる．BL は，与えられたクエリオブジェクトよりもチェックするサブスクリプションに近いデータを順に探索していく．k_{max} が大きくなると，探索を行うデータの数が多くなるため，平均計算時間が増加する．

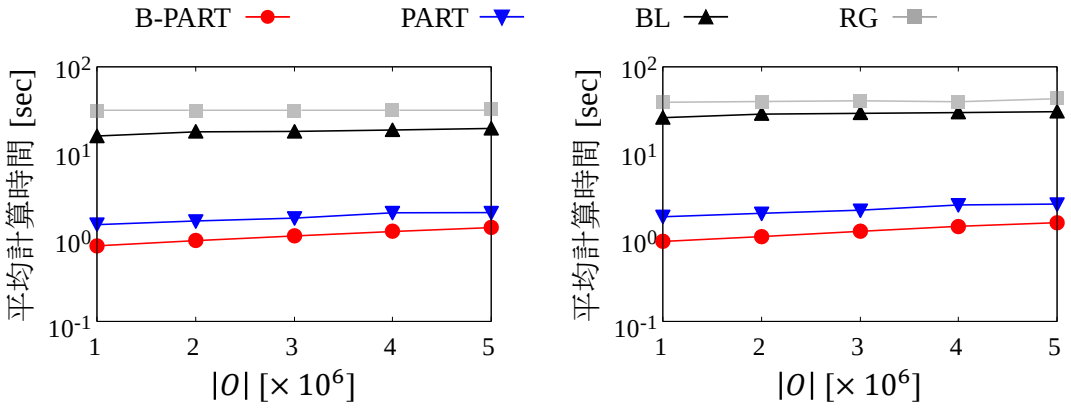
$|S|$ の影響． 図 4.4 にサブスクリプションの数 $|S|$ を変えたときの結果を示す．提案手法は，RG および BL と比較して，常に高速に解を計算できることがわかる．また，$|S|$ の増加に伴い，各手法において平均計算時間が増加し，さらに，提案手法の平均計算時間の増加率は RG および BL と比較して小さいことがわかる．RG および BL は評価を行う必要のあるサブスクリプションを順にチェックするのに対し，提案手法は，サブスクリプションをグループに分割し，グループ単位でチェックを行う．その結果，チェックを行う回数を削減できるため，$|S|$ の増加による影響が小さくなる．

$|O|$ の影響． 図 4.5 にデータの数 $|O|$ を変えたときの結果を示す．提案手法は，RG および BL と比較して，常に高速に解を計算できることがわかる．また，$|O|$ の増加に伴い，各手法において平均計算時間がわずかに増加することがわかる．$|O|$ が増加すると，RG では，IR 木のノードの数が増加するため，各サブスクリプションに対する Top-k 検索の実行時間がわずかに増加する．BL では，GetNearestData の実行時間が増加する．また，提案手法では，解となり得るサブスクリプションの数が減少するため，Verification および CheckSubscription を実行する回数は減少



(a) *TWEETS* (b) *PLACES*

図 4.4: $|S|$ の影響

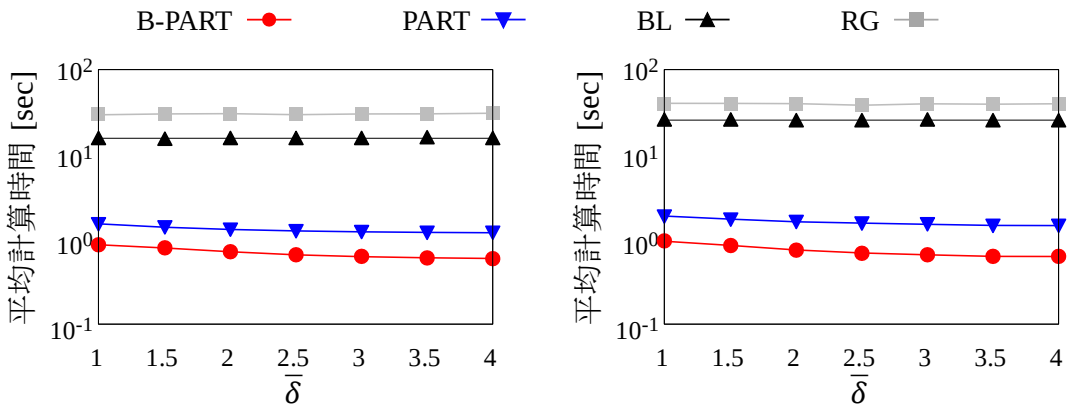


(a) *TWEETS* (b) *PLACES*

図 4.5: $|O|$ の影響

するが，各サブスクリプションのグループに対して評価する R 木のノードの数が増加するため，平均計算時間がわずかに増加する．

$\bar{\delta}$ の影響． 図 4.6 に近似率 $\bar{\delta}$ を変えたときの結果を示す．提案手法は，RG および BL と比較して，常に高速に解を計算できることがわかる．また，$\bar{\delta}$ が増加すると，RG および BL の平均計算時間はほぼ一定であるのに対し，提案手法の平均計算時間は減少することがわかる．これは，$\bar{\delta}$ が増加すると，定理 4.2 の条件を満たすノードの数が増加し，より多くのノード（そのノードを根とする部分木に含まれるデー



(a) *TWEETS*　　(b) *PLACES*

図 4.6: $\overline{\delta}$ の影響

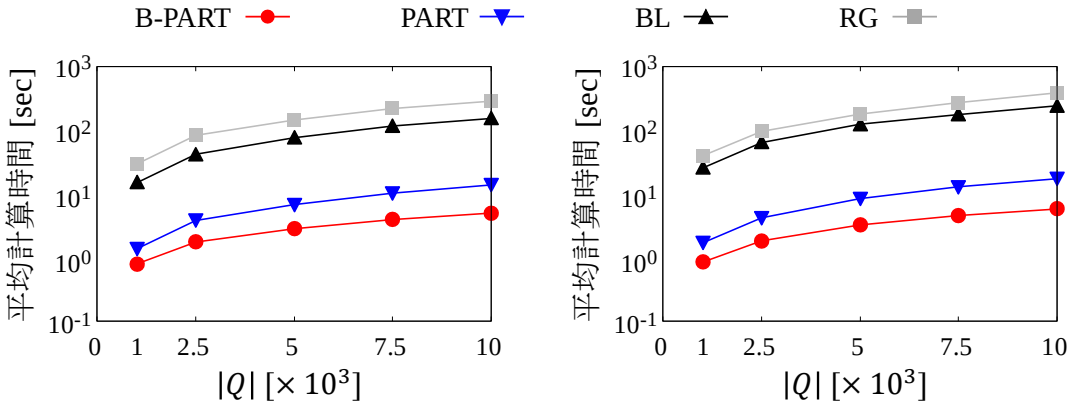
タ）をチェックする必要がなくなるためである．

$|Q|$ の影響． 図 4.7 に一度に与えられるクエリの数 $|Q|$ を変えたときの結果を示す．提案手法は，RG および BL と比較して，常に高速に解を計算できることがわかる．また，$|Q|$ の増加に伴い，各手法において平均計算時間が増加することがわかる．さらに，$|Q|$ が増加すると，B-PART と PART の平均計算時間の差が大きくなることがわかる．$|Q|$ が増加すると，与えられたクエリ集合の中で，共通するキーワードを持つクエリオブジェクトの数が増加する．そのため，B-PART は PART と比較して，各サブスクリプションのグループに対するチェックの回数をより多く削減できるため，平均計算時間の差が大きくなる．

マルチコア処理

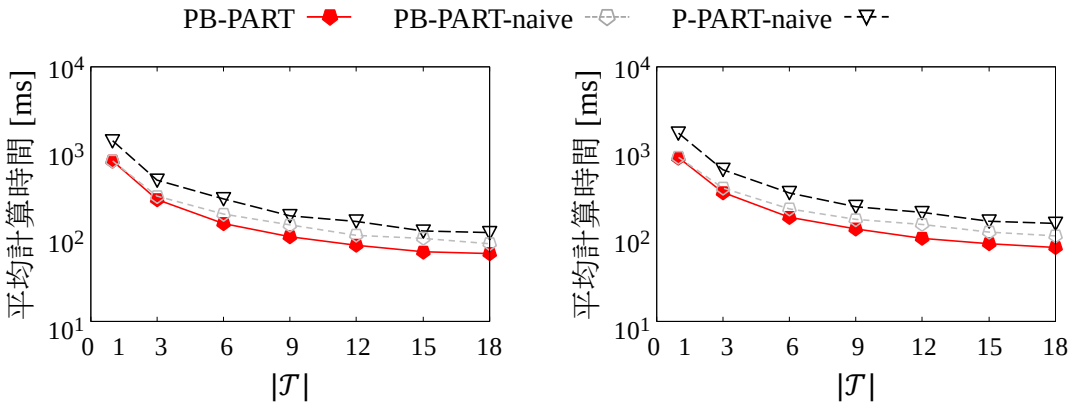
図 4.8 に，使用する CPU コアの数 $|\mathcal{T}|$ を変えたときの結果を示す．PB-PART および PB-PART-naive を比較すると，PB-PART の方が $|\mathcal{T}|$ の増加に対して，平均計算時間が減少することがわかり，提案したコストモデルに基づくキーワードの分散が有効であることを示している．また，PB-PART および P-PART-naive を比較すると，PB-PART は常に高速に解を計算できることがわかる．

さらに，図 4.9 に，各手法のシングルコア処理に対する計算速度の増加率を示す．



(a) *TWEETS* (b) *PLACES*

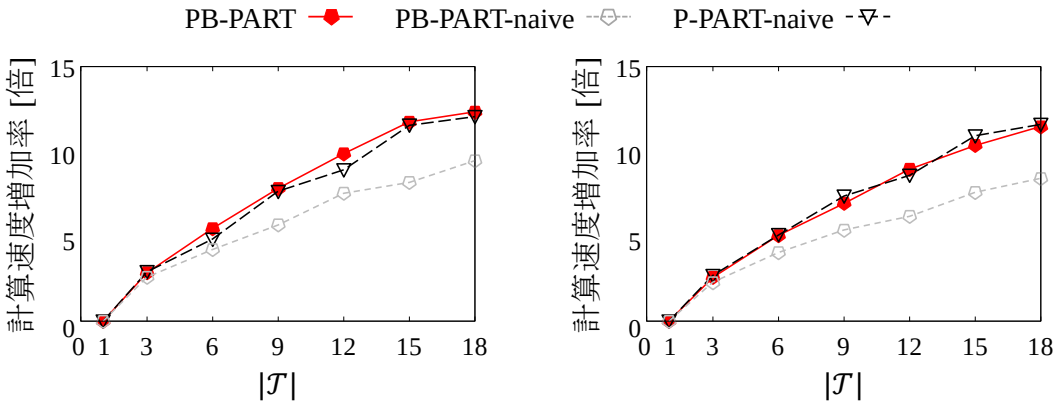
図 4.7: $|Q|$ の影響



(a) *TWEETS* (b) *PLACES*

図 4.8: $|\mathcal{T}|$ の影響（計算時間）

PB-PART-naive および P-PART-naive を比較すると，PB-PART-naive の方が増加率が小さいことがわかる．P-PART-naive は，与えられたクエリ集合に含まれるクエリをハッシュパーティションして処理を行う．このとき，各クエリに対する処理時間が大きく異なることは少ないため，各コアにおける処理時間が大きく異なることは少ない．一方，PB-PART-naive は，計算されたキーワード集合に含まれるキーワードをハッシュパーティションして処理を行う．このとき，各キーワードに対する処理時間はそのキーワードを含むデータおよびサブスクリプションの



(a) *TWEETS*　　(b) *PLACES*

図 4.9: $|\mathcal{T}|$ の影響（計算速度）

数に依存して大きく異なる．そのため，処理時間の大きいキーワードを多く処理するコアにおける処理時間が大きくなる．その結果，PB-PART-naive の計算時間の増加率は小さくなる．PB-PART は，PB-PART-naive における問題を解決するため，コストモデルに基づいてキーワードを分散する．これにより，各コアにおける処理時間の差を小さくできるため，P-PART-naive と同程度の計算時間の増加率を達成できる．

4.6　関連研究

本節では，逆 Top-k 検索に関する従来研究，および近似逆最近傍検索に関する従来研究について述べる．

4.6.1　逆 Top-k 検索

これまでに，多くの研究が逆 Top-k 検索に関する問題に取り組んでいる．文献 [5, 6, 23, 38, 69, 78, 80, 81, 84] では，位置情報に基づき逆 k 最近傍検索を行う問題に取り組んでいる．しかし，これらの研究はキーワードを考慮していないため，本章における問題とは異なる．一方，様々な研究が位置およびキーワードに基づく

逆 Top-k 検索に関する問題に取り組んでいる [16, 22, 46, 47, 89, 90]．例えば，文献 [89] では，位置情報およびキーワードに加えてソーシャル関係に基づく逆 Top-k 検索の解を検索する問題に取り組んでいる．具体的には，ユーザ集合およびデータ集合を GIM-tree と呼ばれる木構造で管理し，クエリの解となり得るユーザの集合を限定することで，効率的に解を計算する．しかし，この研究では，距離関数として道路ネットワーク上での距離を想定しているため，本章における問題とは異なる．

4.6.2 近似逆最近傍検索

文献 [30, 31] において，近似逆最近傍検索に関する問題に取り組んでいる．これらの文献では，各サブスクリプション s に対して最近傍であるデータまでの距離を $Edist_{nn}(s)$ をしたとき，あるクエリオブジェクト o_q に対して $Edist(s.l, o_q.l) \leq \delta \cdot Edist_{nn}(s)$（$\delta > 1$）を満たすすべてのサブスクリプションを検索する．しかし，これらの文献は，Top-k 検索を対象としていない．さらに，キーワードも考慮していないため，本章における問題とは異なる．

4.7 むすび

情報配信や情報推薦の分野において，情報提供側が自身のデータに興味を持つユーザである潜在顧客を把握することは，市場分析やマーケティング調査を行う上で重要な課題である．Pub/Sub システムにおいて，あるデータを Top-k データに含むユーザを検索する逆 Top-k 検索の解は，そのデータに対する潜在顧客とみなすことができる．

本章では，Pub/Sub システムにおいて，位置およびキーワードに基づいて近似的に逆 Top-k 検索を行う近似逆 Top-k クエリ（ART クエリ）を新たに提案した．そして，ART クエリの解を高速に検索する手法を提案した．提案手法では，サブスクリプション集合をキーワードごとに位置が近いもの同士で複数のグループに分割して管理し，データ集合をキーワードごとに木構造を用いて管理する．これ

により，あるクエリが与えられたとき，クエリの解となり得るサブスクリプションのグループに対して処理を行い，クエリの解となり得ないと判断された時点でそのグループに対する処理を中断することで，計算時間を削減する．また，提案手法を拡張し，複数のクエリが同時に与えられたとき，それらに対してバッチ処理を行う手法を提案した．さらに，マルチコア環境で処理を行う手法も提案した．実データを用いた実験の結果から，提案手法は，高速に解を計算できることを確認した．

本章における今後の課題として，継続的な逆 Top-k 検索への対応が考えられる．本章では，ある時刻において逆 Top-k 検索を行う問題に取り組んだ．しかし，アプリケーションによっては，継続的に逆 Top-k 検索を実行し，データの生成および削除による潜在顧客の変化を市場分析やマーケティング調査に利用することも考えられる．このような環境に適応するために，手法の拡張を行う必要がある．

第5章　結論

5.1　本論文のまとめ

本論文では，位置依存型 Pub/Sub システムにおける Top-k 検索手法について議論した．位置依存型 Pub/Sub システムにおける Top-k 検索手法は，位置情報が付加されたデータをそれらに興味を持つユーザに効率的に配信するために有用な手法である．ここで，近年では，飲食店や観光スポットなどの PoI だけでなく，一般のユーザも多くのデータを発信しているため，多種多様なデータが大量に生成されている．それに伴い，データを受信するユーザ側の要求も多様化しているため，事前に指定した位置情報およびキーワードのみに基づいてデータを配信するだけでは，各ユーザの興味に合わせたデータを配信することは困難である．そこで，本論文では，位置依存型 Pub/Sub システムにおいて，情報受信側と情報配信側の双方に対して，新たな付加価値を付けたサービスを提供するため，ユーザ個々の興味により合うデータを効率的に配信できる手法や，情報配信側が潜在顧客を高速に把握できる手法を提案した．

まず，第 1 章では，位置依存型 Pub/Sub システムにおいて，各ユーザのニーズにより合わせたデータの配信を実現するため，情報受信側と情報提供側のそれぞれの要求について議論し，それらを達成することの重要性について述べた．

第 2 章では，システムにおいて，位置，キーワード，およびソーシャル関係に基づく，各サブスクリプションの Top-k データをカウントベースのスライディングウィンドウ上でモニタリングする手法を提案した．カウントベースのスライディングウィンドウでは，新たにデータが生成されると，ウィンドウがスライドし，最も古いデータがウィンドウから削除され，生成されたデータがウィンドウに挿入される．データが生成および削除されると，各サブスクリプションの Top-k デー

タが変化する可能性があるため，効率的に Top-k データをモニタリングするためには，データの生成および削除に対して高速に処理を行う必要がある．データの削除に対して，削除されたデータを Top-k データに含むサブスクリプションについて新たな Top-k データになるデータを高速に発見するため，提案手法では，各サブスクリプションに対して，後に Top-k データとなり得るデータの集合である k-スカイバンドを管理する．一方，データの生成に対して，生成されたデータが Top-k データになる可能性のあるサブスクリプションの候補を限定し，それらの Top-k データのみを更新するため，提案手法では，サブスクリプションを木構造で管理し，木構造に対応したリストでソーシャルスコアを管理する．提案手法の有効性を示すために，シミュレーション実験による性能評価を行った．その結果より，提案手法は，ウィンドウのスライドに対して，高速に各サブスクリプションの Top-k データを更新できることを確認した．

第3章では，ユーザの移動を考慮した上で，位置およびキーワードに基づく，各サブスクリプションの Top-k データをモニタリングする新たなシステム Lamps を提案した．ユーザの移動，データの生成および削除により，各サブスクリプションの Top-k データが変化する可能性がある．そのため，効率的に Top-k データをモニタリングするためには，ユーザの移動，データの生成および削除それぞれに対して高速に処理を行う必要がある．ユーザの移動に対して，Top-k データが変化する場合を把握するため，Lamps では，各サブスクリプションに対して，ユーザが移動しても Top-k データが変化しない領域である Safe Region を計算する．データの生成に対して，ユーザの移動を考慮した上でサブスクリプションを管理し，生成されたデータが Top-k データになり得るサブスクリプションの候補を高速に限定するため，Lamps では，計算された Safe Region に基づく新たな木構造（SQ-tree）を用いてサブスクリプションを管理する．データの削除に対して，新たに Top-k データとなるデータを高速に発見するため，Lamps では，Kd 木およびキーワードごとの転置ファイルおよび四分木を組み合わせた新たなインデックス（KIQF）を用いてデータを管理する．Lamps の有効性を示すために，シミュレーション実験による性能評価を行った．その結果より，Lamps は，ユーザの移動，データの生成および削除に対して，高速に各サブスクリプションの Top-k データを更新でき

ることを確認した．

第 4 章では，位置およびキーワードに基づく近似逆 Top-k クエリ（ART クエリ）を新たに提案し，ART クエリの解を高速に計算する手法を提案した．高速にクエリの解を計算するため，以下の 2 つの特徴に注目した．1 つ目は，指定する位置やキーワードが類似したサブスクリプション同士は，同じデータが Top-k データとなりやすい．2 つ目は，クエリの解を計算する際，各サブスクリプションの正確な Top-k データを求める必要はなく，クエリとして指定したデータ（クエリオブジェクト）が Top-k データに含まれるかどうかのみ判断できれば良い．クエリオブジェクトよりスコアの良いデータが k 個以上発見された時点で，Top-k データに含まれないと判断できる．1 つ目の特徴を考慮し，類似したサブスクリプションのチェックを同時に実行するため，提案手法では，サブスクリプション集合をキーワードごとに互いに近いもの同士で複数のグループに分割して管理する．2 つ目の特徴を考慮し，サブスクリプションのグループに対して，指定したデータよりスコアの良いデータを高速に発見するため，提案手法では，オブジェクト集合をキーワードごとに木構造を用いて管理する．また，複数のクエリが同時に与えられることを想定し，それらに対してバッチ処理を行う手法およびマルチコア環境でバッチ処理を行う手法も提案した．提案手法の有効性を示すために，シミュレーション実験による性能評価を行った．その結果より，提案手法は，与えられたクエリの解を高速に計算できることを確認した．

本論文で提案した手法は，ソーシャル関係という新たな要素を考慮した上での検索結果やユーザの現在地のように動的に条件を考慮した上での検索結果を，システムにおいて高速に更新することができる．例えば，あるサービスシステムが，スコアリング関数を限定して Pub/Sub サービスを展開する際，百万規模のサブスクリプションであれば，100 個のデータの更新（データの生成および削除）に対して，1 秒かかることなく各サブスクリプションの Top-k データを更新し配信することができる．さらに逆 Top-k 検索手法は，各ユーザが必要としているデータ（情報）に関する知識を，情報提供側が高速に得ることができる．従って本論文で提案した手法は，情報推薦や情報配信の分野において，各ユーザの求めるデータ（情報）のリアルタイムな配信を実現することに向けて，大きな進展をもたらすもの

である．

5.2　今後の展望

近年，ICT を活用した情報配信は盛んに行われており，企業から一般の人々までが多種多様な情報を含むデータを発信している．その結果，配信対象となるデータに含まれる要素やユーザの検索要求は，今後さらに多様化することが予想される．このような環境では，同一のスコアリング関数を用いて Top-k データを計算するだけでは，各ユーザ（サブスクリプション）にとって有用なデータを出力することは難しい．そのため，対象とするデータに含まれる要素やユーザの検索要求（指定する条件やスコアリング関数など）がそれぞれ異なる場合でも，各ユーザにとって有用なデータを効率的に検索・配信できる枠組みが必要である．

そこで，本論文の内容をより発展させ，多種多様なデータおよび検索要求を対象とした場合でも，各ユーザに対して有用なデータ（情報）を配信可能なインデックスやアルゴリズムを実現することが今後の重要な研究課題として挙げられる．例えば，ユーザ（サブスクリプション）ごとに異なるスコアリング関数を指定するという想定の下で，システムにおいて，各サブスクリプションの Top-k データをモニタリングすることを考える．このとき，同一のスコアリング関数を指定するサブスクリプションごとにインデックスを作成するのではなく，任意のスコアリング関数に対応できるインデックスによりサブスクリプションを管理し，データの生成や削除に対して，まとめて処理を行うことが可能なアルゴリズムを設計することが考えられる．これにより，ユーザのいかなる要求にも対応できる Pub/Sub システムを実現することが可能になるため，情報配信や情報推薦の分野のさらなる発展が期待される．

謝辞

本研究を推進するにあたり，懇切なる御指導と惜しみない御助言を頂きました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 原 隆浩教授に謹んで御礼申し上げます．

本研究を推進するにあたり，直接の御指導，御助言，御討論を頂きました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 天方 大地助教に衷心より感謝申し上げます．

本論文をまとめるにあたり，大変有益な御指導と御助言を多数賜りました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 鬼塚 真教授，大阪大学データビリティフロンティア機構 春本 要教授　に心より感謝申し上げます．

講義，学生生活を通じて，学問に取り組む姿勢をご教授頂きました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　藤原　融教授，下條 真司教授，松下 康之教授，萩田 紀博教授に厚く感謝申し上げます．

本研究において，多大なる御助言，御支援を頂きました大阪大学西尾 章治郎総長に深く御礼申し上げます．

本研究において，多大なる御助言，御協力，御支援を頂きました大阪大学サイバーメディアセンター 義久 智樹准教授，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 前川 卓也准教授，白川 真澄招聘准教授，大阪大学データビリティフロンティア機構 中野 賢特任准教授に深謝致します．

本研究において，ともに研究を進め，多大なるご協力を頂きました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 高 博奇氏，加藤　慎也氏，鶴岡　翔平氏，新井　悠介氏，中谷　諒平氏，仲摩　隼人氏に深く御礼申し上げます．

本研究を進めるにあたり，多くの御討論や御助言を頂きました大阪大学大学院

情報科学研究科マルチメディア工学専攻 原研究室の諸氏に心より感謝申し上げます.

最後に，私のこれまでの人生，そして研究生活を送る上で，暖かい支援と理解を頂きました家族や友人に心から感謝致します.

参考文献

[1] Ahuja, R., Armenatzoglou, N., Papadias, D., and Fakas, G. J.: Geo-Social Keyword Search, in *Proc. 14th Int'l Conf. on Symposium on Spatial and Temporal Databases (SSTD 2015)*, pp. 431–450, Springer (2015).

[2] Almaslukh, A., and Magdy, A.: Evaluating spatial-keyword queries on streaming data, in *Proc. 26th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems (ACM SIGSPATIAL 2018)*, pp. 209–218, ACM (2018).

[3] Armenatzoglou, N., Papadopoulos, S., and Papadias, D.: A general framework for geo-social query processing, *Proc. the VLDB Endowment (PVLDB)*, Vol. 6, No. 10, pp. 913–924 (2013).

[4] Bentley, J. L.: Multidimensional binary search trees used for associative searching, *Communications of the ACM*, Vol. 18, No. 9, pp. 509–517 (1975).

[5] Cheema, M. A., Lin, X., Zhang, W., and Zhang, Y.: Influence zone: Efficiently processing reverse k nearest neighbors queries, in *Proc. 27th Int'l Conf. on Data Engineering (ICDE 2011)*, pp. 577–588, IEEE (2011).

[6] Cheema, M. A., Zhang, W., Lin, X., and Zhang, Y.: Efficiently processing snapshot and continuous reverse k nearest neighbors queries, *The VLDB Journal (VLDBJ)*, Vol. 21, No. 5, pp. 703–728 (2012).

[7] Chen, L., Cong, G., and Cao, X.: An efficient query indexing mechanism for filtering geo-textual data, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2013)*, pp. 749–760, ACM (2013).

[8] Chen, L., Cong, G., Cao, X., and Tan, K.-L.: Temporal spatial-keyword top-k publish/subscribe, in *Proc. 31st Int'l Conf. on Data Engineering (ICDE 2015)*, pp. 255–266, IEEE (2015).

[9] Chen, L., Cong, G., Jensen, C. S., and Wu, D.: Spatial keyword query processing: an experimental evaluation, *Proc. the VLDB Endowment (PVLDB)*, Vol. 6, No. 3, pp. 217–228 (2013).

[10] Chen, L., Lin, X., Hu, H., Jensen, C. S., and Xu, J.: Answering why-not questions on spatial keyword top-k queries, in *Proc. 31st Int'l Conf. on Data Engineering (ICDE 2015)*, pp. 279–290, IEEE (2015).

[11] Chen, L., Shang, S., Zheng, K., and Kalnis, P.: Cluster-based Subscription Matching for Geo-Textual Data Streams, in *Proc. 35th Int'l Conf. on Data Engineering (ICDE 2019)*, pp. 890–901, IEEE (2019).

[12] Chen, L., Xu, J., Jensen, C. S., and Li, Y.: YASK: A why-not question answering engine for spatial keyword query services, *Proc. the VLDB Endowment (PVLDB)*, Vol. 9, No. 13, pp. 1501–1504 (2016).

[13] Chen, L., Xu, J., Lin, X., Jensen, C. S., and Hu, H.: Answering why-not spatial keyword top-k queries via keyword adaption, in *Proc. 32nd Int'l Conf. on Data Engineering (ICDE 2016)*, pp. 697–708, IEEE (2016).

[14] Chou, P. B., Grossman, E., Gunopulos, D., and Kamesam, P.: Identifying prospective customers, in *Proc. 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2000)*, pp. 447–456, ACM (2000).

[15] Choudhury, F. M., Culpepper, J. S., Bao, Z., and Sellis, T.: Batch processing of Top-k Spatial-textual Queries, *ACM Trans. on Spatial Algorithms and Systems (TSAS)*, Vol. 3, No. 4, p. 13 (2018).

[16] Choudhury, F. M., Culpepper, J. S., Sellis, T., and Cao, X.: Maximizing bichromatic reverse spatial and textual k nearest neighbor queries, *Proc. the VLDB Endowment (PVLDB)*, Vol. 9, No. 6, pp. 456–467 (2016).

[17] Chuitian, R., Chunbin, L., Yasin, N. S., Jianguo, W., Wei, L., and Xiaoyong, D.: Fast and Scalable Distributed Set Similarity Joins for Big Data Analytics, in *Proc. 33rd Int'l Conf. on Data Engineering (ICDE 2017)*, pp. 1059–1070, IEEE (2017).

[18] Dimitris, P., Yufei, T., Greg, F., and Bernhard, S.: Progressive Skyline Computaion in Database Systems, *ACM Trans. on Database Systems (TODS)*, Vol. 30, No. 1, pp. 41–82 (2005).

[19] Dong, Y., Chen, H., and Kitagawa, H.: Continuous Search on Dynamic Spatial Keyword Objects, in *Proc. 35th Int'l Conf. on Data Engineering (ICDE 2019)*, pp. 1578–1581, IEEE (2019).

[20] Fabret, F., Jacobsen, H. A., Llirbat, F., Pereira, J., Ross, K. A., and Shasha, D.: Filtering algorithms and implementation for very fast publish/subscribe systems, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2001)*, pp. 115–126, ACM (2001).

[21] Fujiwara, Y., Nakatsuji, M., Yamamuro, T., Shiokawa, H., and Onizuka, M.: Efficient personalized pagerank with accuracy assurance, in *Proc. 18th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2012)*, pp. 15–23, ACM (2012).

[22] Gao, Y., Qin, X., Zheng, B., and Chen, G.: Efficient reverse top-k boolean spatial keyword queries on road networks, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 27, No. 5, pp. 1205–1218 (2014).

[23] Gao, Y., Zheng, B., Chen, G., Lee, W.-C., Lee, K. C., and Li, Q.: Visible reverse k-nearest neighbor query processing in spatial databases, *IEEE Trans.*

on Knowledge and Data Engineering (TKDE), Vol. 21, No. 9, pp. 1314–1327 (2009).

[24] Groh, G., and Ehmig, C.: Recommendations in taste related domains: collaborative filtering vs. social filtering, in *Proc. Int'l Conf. on Supporting Group Work (GROUP 2007)*, pp. 127–136, ACM (2007).

[25] Guo, L., Shao, J., Aung, H. H., and Tan, K.-L.: Efficient continuous top-k spatial keyword queries on road networks, *GeoInformatica*, Vol. 19, No. 1, pp. 29–60 (2015).

[26] Guo, L., Zhang, D., Li, G., Tan, K.-L., and Bao, Z.: Location-aware pub/sub system: When continuous moving queries meet dynamic event streams, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2015)*, pp. 843–857, ACM (2015).

[27] Haghani, P., Michel, S., and Aberer, K.: Evaluating top-k queries over incomplete data streams, in *Proc. 19th Int'l Conf. on Information and Knowledge Management (CIKM 2009)*, pp. 877–886, ACM (2009).

[28] Haghani, P., Michel, S., and Aberer, K.: The gist of everything new: personalized top-k processing over web 2.0 streams, in *Proc. 20th Int'l Conf. on Information and Knowledge Management (CIKM 2010)*, pp. 489–498, ACM (2010).

[29] He, Z., and Lo, E.: Answering why-not questions on top-k queries, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 26, No. 6, pp. 1300–1315 (2014).

[30] Hidayat, A., Cheema, M. A., and Taniar, D.: Relaxed reverse nearest neighbors queries, in *Proc. 14th Int'l Conf. on Symposium on Spatial and Temporal Databases (SSTD 2015)*, pp. 61–79, Springer (2015).

[31] Hidayat, A., Yang, S., Cheema, M. A., and Taniar, D.: Reverse Approximate Nearest Neighbor Queries, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 30, No. 2, pp. 339–352 (2018).

[32] Hu, H., Liu, Y., Li, G., Feng, J., and Tan, K.-L.: A location-aware publish/subscribe framework for parameterized spatio-textual subscriptions, in *Proc. 31st Int'l Conf. on Data Engineering (ICDE 2015)*, pp. 711–722, IEEE (2015).

[33] Huang, W., Li, G., Tan, K.-L., and Feng, J.: Efficient Safe-region Construction for Moving top-K Spatial Keyword Queries, in *Proc. 21st Int'l Conf. on Information and Knowledge Management (CIKM 2012)*, pp. 932–941, ACM (2012).

[34] Jeh, G., and Widom, J.: SimRank: a measure of structural-context similarity, in *Proc. 8th ACM SIGKDD Int'l Conf. on Knowledge discovery and Data Mining (SIGKDD 2002)*, pp. 538–543, ACM (2002).

[35] Jeh, G., and Widom, J.: Scaling personalized web search, in *Proc. Int'l Conf. on World Wide Web (WWW)*, pp. 271–279, ACM (2003).

[36] König, A. C., Church, K., and Markov, M.: A data structure for sponsored search, in *Proc. 25th Int'l Conf. on Data Engineering (ICDE 2009)*, pp. 90–101, IEEE (2009).

[37] Korf, R. E.: Multi-way number partitioning, in *Proc. 21th Int'l Joint Conf. on Artifical intelligence (IJCAI 2009)*, pp. 538–543 (2009).

[38] Korn, F., and Muthukrishnan, S.: Influence Sets Based on Reverse Nearest Neighbor Queries, in *Proc. 6th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2000)*, pp. 201–212, ACM (2000).

[39] Li, C., Gu, Y., Qi, J., Yu, G., Zhang, R., and Yi, W.: Processing moving k NN queries using influential neighbor sets, *Proc. the VLDB Endowment (PVLDB)*, Vol. 8, No. 2, pp. 113–124 (2014).

[40] Li, G., Feng, J., and Xu, J.: Desks: Direction-aware spatial keyword search, in *Proc. 28th Int'l Conf. on Data Engineering (ICDE 2012)*, pp. 474–485, IEEE (2012).

[41] Li, G., Wang, Y., Wang, T., and Feng, J.: Location-aware publish/subscribe, in *Proc. 19th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2013)*, pp. 802–810, ACM (2013).

[42] Li, Y., Chen, R., Xu, J., Huang, Q., Hu, H., and Choi, B.: Geo-Social K-Cover Group Queries for Collaborative Spatial Computing, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 27, No. 10, pp. 2729–2742 (2015).

[43] 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. on Knowledge and Data Engineering (TKDE)*, Vol. 23, No. 4, pp. 585–599 (2010).

[44] Lin, X., Xu, J., and Hu, H.: Reverse keyword search for spatio-textual top-k queries in location-based services, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 27, No. 11, pp. 3056–3069 (2015).

[45] Liu, W., Sun, W., Chen, C., Huang, Y., Jing, Y., and Chen, K.: Circle of friend query in geo-social networks, in *Proc. 17th Int'l Conf. on Database Systems for Advanced Applications (DASFFA 2012)*, pp. 126–137Springer (2012).

[46] Lu, J., Lu, Y., and Cong, G.: Reverse spatial and textual k nearest neighbor search, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011)*, pp. 349–360, ACM (2011).

[47] Lu, Y., Lu, J., Cong, G., Wu, W., and Shahabi, C.: Efficient algorithms and cost models for reverse spatial-keyword k-nearest neighbor search, *ACM Trans. on Database Systems (TODS)*, Vol. 39, No. 2, p. 13 (2014).

[48] Mahmood, A. R., Aly, A. M., and Aref, W. G.: FAST: Frequency-Aware Indexing for Spatio-Textual Data Streams, in *Proc. 34th Int'l Conf. on Data Engineering (ICDE 2018)*, pp. 305–316, IEEE (2018).

[49] Mouratidis, K., Bakiras, S., and Papadias, D.: Continuous monitoring of top-k queries over sliding windows, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2006)*, pp. 635–646, ACM (2006).

[50] Mouratidis, K., Li, J., Tang, Y., and Mamoulis, N.: Joint search by social and spatial proximity, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 27, No. 3, pp. 781–793 (2015).

[51] Mouratidis, K., and Pang, H.: Efficient evaluation of continuous text search queries, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 23, No. 10, pp. 1469–1482 (2011).

[52] Nanongkai, D., Sarma, A. D., Lall, A., Lipton, R. J., and Xu, J.: Regret-minimizing representative databases, *Proc. the VLDB Endowment (PVLDB)*, Vol. 3, No. 1-2, pp. 1114–1124 (2010).

[53] 西尾俊哉, 天方大地, 原隆浩, 西尾章治郎：位置，キーワードおよびソーシャル関係に基づく Top-k パブリッシュ/サブスクライブ, 第 8 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2016) 論文集, D3-3 (2016).

[54] Nishio, S., Amagata, D., and Hara, T.: Geo-Social Keyword Top-k Data Monitoring over Sliding Window, in *Proc. 28th Int'l Conf. on Database and Expert Systems Applications (DEXA 2017), Part I*, LNCS 10438, pp. 409–424, Springer (2017).

[55] 西尾俊哉, 天方大地, 原隆浩 : スライディングウィンドウ上での位置・キーワード・ソーシャル関係に基づく Top-k パブリッシュ/サブスクライブ, 第 9 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2017) 論文集, E4-5 (2017).

[56] 西尾俊哉, 天方大地, 原隆浩 : 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, 情報処理学会論文誌データベース (TOD), Vol. 10, No. 2, pp. 8–18 (2017).

[57] 西尾俊哉, 天方大地, 原隆浩 : 位置・キーワードに基づくムービング Top-k パブリッシュ/サブスクライブ, 第 10 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2018) 論文集, G7-2 (2018).

[58] 西尾俊哉, 天方大地, 原隆浩 : ユーザの移動を考慮したキーワード付き空間データに対する Top-k 問合せのためのストリーミングアルゴリズム, 電子情報通信学会和文論文誌 D, Vol. J102-D, No. 1, pp. 1–12 (2019).

[59] 西尾俊哉, 天方大地, 原隆浩 : ユーザの移動を考慮した位置・キーワードに基づく Top-k データモニタリング, 第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM Forum 2019) 論文集, D5-5 (2019).

[60] 西尾俊哉, 天方大地, 原隆浩 : 位置およびキーワードに基づく近似逆 k 最近傍検索, 情報処理学会 第 27 回マルチメディア通信と分散処理ワークショップ (DPSWS 2019) 論文集, pp. 118–124 (2019).

[61] Park, M.-H., Hong, J.-H., and Cho, S.-B.: Location-based recommendation system using bayesian user's preference model in mobile devices, in *Proc. 4th Int'l Conf. on Ubiquitous Intelligence and Computing (UIC 2007)*, pp. 1130–1139 (2007).

[62] Pripužić, K., Žarko, I. P., and Aberer, K.: Top-k/w publish/subscribe: finding k most relevant publications in sliding time window w, in *Proc. 2nd Int'l Conf. on Distributed Event-Based Systems (DEBS 2008)*, pp. 127–138, ACM (2008).

[63] Pripužić, K., Žarko, I. P., and Aberer, K.: Time-and space-efficient sliding window top-k query processing, *ACM Trans. on Database Systems (TODS)*, Vol. 40, No. 1, p. 1 (2015).

[64] Qi, J., Zhang, R., Jensen, C. S., Ramamohanarao, K., and HE, J.: Continuous Spatial Query Processing: A Survey of Safe Region Based Techniques, *ACM Computing Surveys (CSUR)*, Vol. 51, No. 3, p. 64 (2018).

[65] Sadoghi, M., and Jacobsen, H.-A.: Be-tree: an index structure to efficiently match boolean expressions over high-dimensional discrete space, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2001)*, pp. 637–648, ACM (2011).

[66] Schenkel, R., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J. X., and Weikum, G.: Efficient top-k querying over social-tagging networks, in *Proc. 31st ACM SIGIR Int'l Conf. on Research and Development in Information Retrieval (SIGIR 2008)*, pp. 523–530, ACM (2008).

[67] Shraer, A., Gurevich, M., Fontoura, M., and Josifovski, V.: Top-k publish-subscribe for social annotation of news, *Proc. the VLDB Endowment (PVLDB)*, Vol. 6, No. 6, pp. 385–396 (2013).

[68] Subbian, K., Aggarwal, C., and Hegde, K.: Recommendations for streaming data, in *Proc.25th Int'l Conf. on Information and Knowledge Management (CIKM 2016)*, pp. 2185–2190, ACM (2016).

[69] Tao, Y., Papadias, D., and Lian, X.: Reverse kNN search in arbitrary dimensionality, in *Proc. 30th Int'l Conf. on Very Large Data Bases (VLDB 2004)*, pp. 744–755 (2004).

[70] Vlachou, A., Doulkeridis, C., Nørvåg, K., and Kotidis, Y.: Branch-and-bound algorithm for reverse top-k queries, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2013)*, pp. 481–492, ACM (2013).

[71] Wang, X., Zhang, W., Zhang, Y., Lin, X., and Huang, Z.: Top-k spatial-keyword publish/subscribe over sliding window, *The VLDB Journal (VLDBJ)*, Vol. 26, No. 3, pp. 301–326 (2017).

[72] Wang, X., Zhang, Y., Zhang, W., Lin, X., and Huang, Z.: Skype: top-k spatial-keyword publish/subscribe over sliding window, *Proc. the VLDB Endowment (PVLDB)*, Vol. 9, No. 7, pp. 588–599 (2016).

[73] Wang, X., Zhang, Y., Zhang, W., Lin, X., and Wang, W.: Ap-tree: Efficiently support continuous spatial-keyword queries over stream, in *Proc. 31st Int'l Conf. on Data Engineering (ICDE 2015)*, pp. 1107–1118, IEEE (2015).

[74] Wang, X., Zhang, Y., Zhang, W., Lin, X., and Wang, W.: AP-Tree: efficiently support location-aware Publish/Subscribe, *The VLDB Journal (VLDBJ)*, Vol. 24, No. 6, pp. 823–848 (2015).

[75] Whang, S. E., Garcia-Molina, H., Brower, C., Shanmugasundaram, J., Vassilvitskii, S., Vee, E., and Yerneni, R.: Indexing boolean expressions, *Proc. the VLDB Endowment (PVLDB)*, Vol. 2, No. 1, pp. 37–48 (2009).

[76] Wu, D., Li, Y., Choi, B., and Xu, J.: Social-aware top-k spatial keyword search, in *Proc. 15th Int'l Conf. on Mobile Data Management (MDM 2014)*, pp. 235–244IEEE (2014).

[77] Wu, D., Yiu, M. L., Jensen, C. S., and Cong, G.: Efficient continuously moving top-k spatial keyword query processing, in *Proc. 27th Int'l Conf. on Data Engineering (ICDE 2011)*, pp. 541–552, IEEE (2011).

[78] Wu, W., Yang, F., Chan, C.-Y., and Tan, K.-L.: Finch: Evaluating reverse k-nearest-neighbor queries on location data, *Proc. the VLDB Endowment (PVLDB)*, Vol. 1, No. 1, pp. 1056–1067 (2008).

[79] Yang, D.-N., Shen, C.-Y., Lee, W.-C., and Chen, M.-S.: On socio-spatial group query for location-based social networks, in *Proc. 18th ACM SIGKDD*

Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2012), pp. 949–957, ACM (2012).

[80] Yang, S., Cheema, M. A., Lin, X., and Wang, W.: Reverse k nearest neighbors query processing: experiments and analysis, *Proc. the VLDB Endowment (PVLDB)*, Vol. 8, No. 5, pp. 605–616 (2015).

[81] Yang, S., Cheema, M. A., Lin, X., and Zhang, Y.: SLICE: reviving regions-based pruning for reverse k nearest neighbors queries, in *Proc. 30th Int'l Conf. on Data Engineering (ICDE 2014)*, pp. 760–771, IEEE (2014).

[82] Yi, K., Yu, H., Yang, J., Xia, G., and Chen, Y.: Efficient maintenance of materialized top-k views, in *Proc. 19th Int'l Conf. on Data Engineering (ICDE 2003)*, pp. 189–200, IEEE (2003).

[83] Yiu, M. L., and Mamoulis, N.: Efficient processing of top-k dominating queries on multi-dimensional data, in *Proc. 33rd Int'l Conf. on Very Large Data Bases (VLDB 2007)*, pp. 483–494 (2007).

[84] Yiu, M. L., Papadias, D., Mamoulis, N., and Tao, Y.: Reverse nearest neighbors in large graphs, *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, Vol. 18, No. 4, pp. 540–553 (2006).

[85] Žalik, B.: Two efficient algorithms for determining intersection points between simple polygons, *Computers & Geosciences*, Vol. 26, No. 2, pp. 137–151 (2000).

[86] Zhang, C., Zhang, Y., Zhang, W., and Lin, X.: Inverted linear quadtree: Efficient top k spatial keyword search, in *Proc. 29th Int'l Conf. on Data Engineering (ICDE 2013)*, pp. 901–912, IEEE (2013).

[87] Zhang, D., Chan, C.-Y., and Tan, K.-L.: An efficient publish/subscribe index for e-commerce databases, *Proc. the VLDB Endowment (PVLDB)*, Vol. 7, No. 8, pp. 613–624 (2014).

[88] Zhang, D., Tan, K.-L., and Tung, A. K.: Scalable top-k spatial keyword search, in *Proc. 16th Int'l Conf. on Extending Database Technology (EDBT 2013)*, pp. 359–370ACM (2013).

[89] Zhao, J., Gao, Y., Chen, G., Jensen, C. S., Chen, R., and Cai, D.: Reverse top-k geo-social keyword queries in road networks, in *Proc. 33rd Int'l Conf. on Data Engineering (ICDE 2017)*, pp. 387–398, IEEE (2017).

[90] Zhao, P., Fang, H., Sheng, V. S., Li, Z., Xu, J., Wu, J., and Cui, Z.: Monochromatic and bichromatic ranked reverse boolean spatial keyword nearest neighbors search, *World Wide Web (WWW)*, Vol. 20, No. 1, pp. 39–59 (2017).

[91] Zhou, Y., Xie, X., Wang, C., Gong, Y., and Ma, W.-Y.: Hybrid index structures for location-based web search, in *Proc. 15th Int'l Conf. on Information and Knowledge Management (CIKM 2005)*, pp. 155–162, ACM (2005).

[92] Zhu, Q., Hu, H., Xu, C., Xu, J., and Lee, W.-C.: Geo-social group queries with minimum acquaintance constraints, *The VLDB Journal (VLDBJ)*, Vol. 26, No. 5, pp. 709–727 (2017).