

位置およびキーワードに基づく近似逆 k 最近傍検索

西尾 俊哉^{1,a)} 天方 大地^{1,b)} 原 隆浩^{1,c)}

概要：近年、位置情報およびキーワードを含むオブジェクトおよびサブスクリプションが大量に生成されている。逆 k 最近傍クエリは、オブジェクト集合およびサブスクリプション集合が与えられたとき、あるオブジェクトを k 最近傍に含むサブスクリプション（潜在顧客）を検索するクエリであり、市場分析やマーケティングへの応用が期待されている。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。そこで、本稿では、近似逆 k 最近傍クエリを新たに提案する。オブジェクト集合およびサブスクリプション集合が与えられたとき、近似逆 k 最近傍クエリは、クエリオブジェクトを k 最近傍に含むサブスクリプションに加えて、近似的に k 最近傍に含むサブスクリプションも解に含めてもよいとする。近似逆 k 最近傍クエリの解となり得るすべてのサブスクリプションに対して k 最近傍を計算する単純な手法は非効率的であり、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。この問題を解決するため、クエリの解になり得るサブスクリプションを限定し、それらのみチェックを行うアルゴリズムを提案する。実データを用いた実験により、提案アルゴリズムの有効性を示す。

1. はじめに

近年、SNS や GPS を搭載した端末の普及に伴い、位置情報やキーワードを含むオブジェクトが大量に生成されている。これらのオブジェクトは、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。これらのオブジェクトの中から有用な情報を検索するため、多くのアプリケーションでは、ユーザが自身の興味のある位置やキーワードをサブスクリプションとしてシステムに登録することで、有用な情報を受信する [9], [10]。このような環境において、あるオブジェクトに対して自身を有用と考えるサブスクリプション（潜在顧客）を把握することは市場分析やマーケティングの分野において非常に重要である。これまでに、位置情報およびキーワードに基づいて潜在顧客を検索する逆 k 最近傍クエリに関する研究が行われている [7], [11]。逆 k 最近傍クエリは、オブジェクト集合 O およびサブスクリプション集合 S が与えられたとき、クエリオブジェクト $o_q \in O$ および k を指定し、 o_q を k 最近傍オブジェクト（ kNN ）に含むサブスクリプションを検索する。ここで、あるサブスクリプション $s \in S$ の kNN は、位置情報およびキーワードに基づいて計算される s に対して有用な上位 k 個のオブジェクトの集合である。

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 [5], [6]。あるサブスクリプション s から k 番目のオブジェクト o_k と $k+1$ 番目のオブジェクト o_{k+1} までのそれぞれの距離が同程度であるとき、それらの有用度も同程度であると考えられる。このとき、 s は o_{k+1} に対して近似的に潜在顧客である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高速に検索できれば、より高度な市場分析への応用が期待できる。そこで、本稿では、近似逆 k 最近傍クエリを新たに提案する。近似逆 k 最近傍クエリは、 O および S が与えられたとき、クエリオブジェクト o_q 、 k 、および近似率 c を指定する。近似逆 k 最近傍クエリは、 o_q を kNN に含むサブスクリプションに加えて、 kNN に含まないが $d(s, o_q) < c \cdot d(s, o_k)$ を満たすようなサブスクリプション s も解に含めてもよいとする。ここで、 o_k は、 s に対する k 番目のオブジェクトであり、 $d(\cdot, \cdot)$ は、2点間のユークリッド距離である。

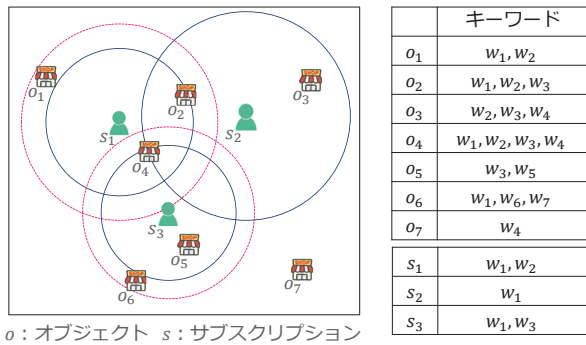
例 1. 図 1 では、3 人のユーザがサブスクリプションを登録しており、7 つのオブジェクトが存在している。 $k=2$ であると仮定すると、 s_1 の $2NN$ は、キーワード w_1 または w_2 を含むデータの中で、 s_1 に近い o_4 および o_2 である。同様に、 s_2 の $2NN$ は o_2 および o_4 、 s_3 の $2NN$ は o_5 および o_4 である。よって、 o_4 の潜在顧客は s_1 、 s_2 、および s_3 である。一方、 o_1 や o_6 の潜在顧客は存在しない。ここで、 o_1 は、 s_1 の $2NN$ に含まれないが、 $c \cdot d(s_1, o_2)$ の範囲内（点線の範囲内）に含まれている。そのため、近似逆 2 最近傍クエリのクエリオブジェクトとして o_1 を指定したと

¹ 大阪大学大学院情報科学研究科マルチメディア工学専攻
Yamadaoka 1-5, Suita, Osaka 565-0871, Japan

a) nishio.syunya@ist.osaka-u.ac.jp

b) amagata.daichi@ist.osaka-u.ac.jp

c) hara@ist.osaka-u.ac.jp

図 1: 近似逆 k 最近傍クエリの例

き, s_1 は解に含まれてもよい. 同様に, o_6 を指定したとき, s_3 は解に含まれてもよい.

課題. 近似逆 k 最近傍クエリの解を求める最も単純なアルゴリズムは, クエリオブジェクトのキーワードを 1 つでも含むすべてのサブスクリプションに対して k 最近傍探索を行うものである. これは, サブスクリプションが大量に存在する環境では多大な計算コストが必要である. 実環境では, 同時に複数の近似逆 k 最近傍クエリが与えられることも想定されるため, 各クエリの解を高速に検索するアルゴリズムが望ましい.

貢献. 以下に, 本稿の貢献を示す.

- 本稿では, 位置およびキーワードに基づく近似逆 k 最近傍クエリを提案する (3 章). 筆者らの知る限り, 本問題はこれまでに扱われていない.
- 近似逆 k 最近傍クエリの解を高速に検索するアルゴリズムを提案する (4 章).
- 実データを用いた実験により, 提案アルゴリズムの有効性を示す (5 章).

上記の内容に加えて, 2 章で関連研究について述べ, 6 章で本稿をまとめる.

2. 関連研究

本章では, 位置およびキーワードに基づく逆 k 最近傍クエリに関する従来研究, および位置情報に基づく近似逆最近傍クエリに関する従来研究について紹介する.

位置およびキーワードに基づく逆 k 最近傍クエリ. 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逆 k 最近傍クエリに関する問題に取り組んでいる [3], [4], [7], [11]. 例えば, 文献 [11] 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 k 最近傍クエリの解を検索する問題に取り組んでいる. 具体的には, ユーザ集合およびオブジェクト集合を 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, 本研究とは問題が異なる.

位置情報に基づく近似逆最近傍クエリ. 文献 [5], [6] において, 近似逆最近傍クエリの解を検索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オブジェクトまでの距離を $d_{nn}(s)$ をしたとき, あるクエリオブジェクト o_q に対して $d(s, o_q) \leq x \cdot d_{nn}(s)$ ($x > 1$) 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, k 最近傍クエリ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, 本研究とは問題が異なる.

3. 予備知識

本章では, 本稿で考えるデータモデルおよび問題を詳細に定義し, その後, ベースラインとなるアルゴリズムを紹介する.

3.1 定義

オブジェクト集合を O とする. あるオブジェクト $o \in O$ は, 位置 ($o.p$) およびキーワード集合 ($o.t$) で構成される ($o = \langle p, t \rangle$). $o.p$ は, 緯度と経度によって表される 2 次元平面上の点とする. 同様に, サブスクリプション集合を S とする. あるサブスクリプション $s \in S$ は, 位置 ($s.p$) およびキーワード集合 ($s.t$) で構成される ($s = \langle p, t \rangle$). ここで, あるオブジェクト $o \in O$ があるサブスクリプション $s \in S$ に対して有用であるかどうかを判断するため, s に対して有用な k 個のオブジェクトとして kNN オブジェクト (kNN) を以下のように定義する.

定義 1 (kNN オブジェクト). オブジェクト集合 O が与えられたとき, $s \in S$ の kNN オブジェクト kNN は次の条件を満たす. (i) $|kNN| = k$, (ii) $\forall o \in kNN, \forall o' \in O_{s.t} \setminus kNN, d(s, o) < d(s, o')$. ここで, $O_{s.t}$ は, O に含まれるオブジェクトの中で $s.t$ に含まれるキーワードを 1 つ以上含むオブジェクトの集合であり, $d(\cdot, \cdot)$ は, サブスクリプションとオブジェクト間のユークリッド距離である.

次に, 位置およびキーワードに基づく逆 k 最近傍クエリを定義する.

定義 2 (逆 k 最近傍クエリ). サブスクリプション集合 S およびオブジェクト集合 O が与えられたとき, 逆 k 最近傍クエリ $q = \{o_q, k\}$ ($o_q \in O$) は, o_q を kNN に含むサブスクリプションを検索する. つまり, q の解 $RkNN$ は次の条件を満たす. $RkNN = \{s \in S | o_q \in s.kNN\}$.

最後に, 位置およびキーワードに基づく近似逆 k 最近傍クエリを定義する.

定義 3 (近似逆 k 最近傍クエリ). サブスクリプション集合 S およびオブジェクト集合 O が与えられたとき, 近似逆 k 最近傍クエリ $q = \{o_q, k, c\}$ ($o_q \in O, c \geq 1$) の解を $ARkNN$ とする. 逆 k 最近傍クエリ $q' = \{o_q, k\}$ の解を

$RkNN$ 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) $s \in RkNN$ ならば, $s \in ARkNN$ である.
- (2) $s \notin RkNN$ かつ $d(s, o_q) \leq c \cdot d(s, o_k)$ ならば, s は $ARkNN$ に含まれてもよい.
- (3) $s \notin RkNN$ かつ $c \cdot d(s, o_k) < d(s, o_q)$ ならば, $s \notin ARkNN$ である.

ここで, o_k は, s に対する k 番目のオブジェクトである.

本稿では, 近似逆 k 最近傍クエリの解を高速に計算することを目的とする.

3.2 ベースラインアルゴリズム

本稿は, 本問題に初めて取り組むため, ベースラインとなるアルゴリズムについて考える. まず, 逆 k 最近傍クエリの解を単純に検索するアルゴリズムを考える. ある逆 k 最近傍クエリ $q = (o_q, k)$ が与えられたとき, 解となり得るサブスクリプションは, $o_{q,t}$ 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 $S_{o_q,t}$ である. そのため, 各サブスクリプション $s \in S_{o_q,t}$ に対して, $s.t$ に含まれるキーワードを少なくとも1つ含むオブジェクトの集合 $O_{s,t}$ の中で, s から距離 $d(s, o_q)$ 以内の範囲に含まれるオブジェクトの集合 O_{near} を検索する. $|O_{near}|$ が k を超えたとき, s は q の解になり得ない. ここで, ベースラインアルゴリズムでは, オブジェクトの各キーワードに対して Kd 木 [1] を用いてオブジェクトの位置情報を管理している. そのため, あるキーワード w を含むすべてのオブジェクトを s から近い順に効率的に探索可能である.

アルゴリズム 1 は, ベースラインアルゴリズムを示している. 各サブスクリプション $s \in S_{o_q,t}$ に対して, o_q より s から距離 $d(s, o_q)$ 以内の範囲に含まれるオブジェクトの集合 O_{near} を検索する. 具体的には, 各キーワード $w \in s.t$ に対して, w を含むオブジェクトを s から近い順に評価する. まず, $\text{RetrieveNearestObject}(w, s)$ により, w を含むオブジェクトの中で s に最も近いオブジェクト o を取得する (5行). $|O_{near}|$ が k 未満かつ o が o_q よりも s に近い場合, o を O_{near} に加え, $\text{RetrieveNextObject}(w, s, o)$ により, w を含むオブジェクトの中で o の次に s に近いオブジェクトを取得する (6–8行). この操作を, $|O_{near}|$ が k 以上または条件を満たすオブジェクトが無くなるまで繰り返す. 最後に, $|O_{near}|$ が k 未満であれば s を解に加える (11–12行).

次に, 近似逆 k 最近傍クエリ $q = (o_q, k, c)$ の解 $ARkNN$ を単純に検索するアルゴリズムを考える. まず, 定義 3 より, 以下の定理が成り立つ.

定理 1. $d(s, o_q) \leq c \cdot d(s, o)$ を満たすオブジェクト o は, s に対して q の解の正確性に影響を与えない.

証明. $d(s, o_q) \leq c \cdot d(s, o)$ を満たすオブジェクト o が, $s.kNN$ に含まれる場合でも, 定義 3 の条件 (2) を満たす

Algorithm 1: Baseline

Input: $q = (o_q, k)$

```

1  $RkNN \leftarrow \emptyset$ 
2 for  $\forall s \in S_{o_q,t}$  do
3    $O_{near} \leftarrow \emptyset$ 
4   for  $\forall w \in s.t$  do
5      $o \leftarrow \text{RetrieveNearestObject}(w, s)$ 
6     while  $|O_{near}| < k \wedge d(s, o) < d(s, o_q)$  do
7        $O_{near} \leftarrow O_{near} \cup \{o\}$ 
8        $o \leftarrow \text{RetrieveNextObject}(w, s, o)$ 
9     if  $|O_{near}| \geq k$  then
10      break
11   if  $|O_{near}| < k$  then
12      $RkNN \leftarrow RkNN \cup \{s\}$ 
13 return  $RkNN$ 

```

ため, s は q の $ARkNN$ に含まれてもよい. よって, 定理が成り立つ. $\square$

定理 1 より, $c \cdot d(s, o) < d(s, o_q)$ を満たすオブジェクト o の数が k 以上であるとき, s は解に含まれず, k 未満であるとき, 解に含まれてもよい. つまり, $c \cdot d(s, o) < d(s, o_q)$ を満たすオブジェクトの数を把握することで, 近似逆 k 最近傍クエリの解を求めることが可能である. よって, アルゴリズム 1 の 6 行目において, $d(s, o) < d(s, o_q)$ ではなく, $c \cdot d(s, o) < d(s, o_q)$ を満たす場合のみ以降の処理を行う.

4. 提案アルゴリズム

提案アルゴリズムでは, ある近似逆 k 最近傍クエリ $q = (o_q, k, c)$ が与えられたとき, クエリの解になり得るサブスクリプションを限定し, それらのみチェックを行うことで効率的に解を検索する. ここで, q の解になりやすいサブスクリプションは $o_q.p$ に近いサブスクリプションである. そこで, 提案アルゴリズムでは, オブジェクト集合をキーワード w ごとに R 木 $R[w]$ を用いて管理し, サブスクリプション集合をキーワードごとに複数のグループに分割して管理する. w を含むサブスクリプションのグループの集合を $G[w]$ とする. また, 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に基づく最小外接矩形 $S_g.MBR$ を保持する.

4.1 フィルタリング技術

提案アルゴリズムは, ある近似逆 k 最近傍クエリ $q = (o_q, k, c)$ が与えられたとき, キーワード $w \in o_{q,t}$ を含むサブスクリプションのグループ $S_g \in G[w]$ において, そのグループ内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オブジェクトの集合 $S_g.O_{near}$ を検

索する。ここで、 $S_g.O_{near}$ に含まれるすべてのオブジェクト o は、 $\forall s \in S_g, d(s, o) < d(s, o_q)$ を満たす。このとき、あるサブスクリプションのグループ $S_g \in G[w]$ および w に対する R 木 $R[w]$ のあるノード n を考える。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オブジェクトまたはノード間の距離の上界値および下界値を $d_{ub}(\cdot, \cdot)$ および $d_{lb}(\cdot, \cdot)$ とすると、次の定理が成り立つ。

定理 2. $d_{ub}(S_g, n) < d_{lb}(S_g, o_q)$ ならば、 $\forall s \in S_g, \forall o \in n, o \in S_g.O_{near}$ である。

証明. $\forall s \in S_g, \forall o \in n, d(s, o) \leq d_{ub}(S_g, n)$ かつ $\forall s \in S_g, d_{lb}(S_g, o_q) \leq d(s, o_q)$ より、 $\forall s \in S_g, \forall o \in n, d(s, o) < d(s, o_q)$ である。よって、定理が成り立つ。□

定理 2 より、 $d_{ub}(S_g, n) < d_{lb}(S_g, o_q)$ を満たすノード n がいくつかに存在し、 $|S_g.O_{near}|$ が k 以上となったとき、 $\forall s \in S_g$ は、 $s \notin RkNN$ であるため、クエリの解に含まれなくてもよい。そのため、 $|S_g.O_{near}|$ 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をフィルタリングできる。

また、次の定理も成り立つ。

定理 3. $d_{ub}(S_g, o_q) < c \cdot d_{lb}(S_g, n)$ ならば、 $\forall o \in n$ は $\forall s \in S_g$ に対して q の解の正確性に影響を及ぼさない。

証明. $\forall s \in S_g, \forall o \in n, d_{lb}(S_g, n) \leq d(s, o)$ かつ $\forall s \in S_g, d(s, o_q) \leq d_{ub}(S_g, o_q)$ より、 $d_{ub}(S_g, o_q) < c \cdot d_{lb}(S_g, n)$ ならば、 $\forall s \in S_g, \forall o \in n, d(s, o_q) < c \cdot d(s, o)$ である。定理 1 より、定理が成り立つ。□

定理 3 より、 $d_{ub}(S_g, o_q) < c \cdot d_{lb}(S_g, n)$ を満たすノード n を根とする部分木に含まれるすべてのオブジェクトをチェックを行うことなくフィルタリングできる。

アルゴリズム 2 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、 $R[w]$ の中で、 o_q より近い位置に存在するオブジェクトの集合 O_{near} および o_q より近い位置に存在するオブジェクトを含む可能性のあるノードの集合 N_c を返すアルゴリズムを示している。 $R[w]$ の根ノードから順に以下の操作を実行する。評価を行うノード 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 の各子ノード n' に対して以下の処理を行う。 $c \cdot d_{lb}(\mathcal{S}, n') > d_{ub}(\mathcal{S}, o_q)$ ならば、定理 3 より、 n' を根とする部分木に含まれるすべてのオブジェクト o 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。そのため、 $c \cdot d_{lb}(\mathcal{S}, n') \leq d_{ub}(\mathcal{S}, o_q)$ を満たす場合のみ処理を行う (9–13 行)。 $d_{ub}(\mathcal{S}, n) < d_{lb}(\mathcal{S}, o_q)$ である場合、 $\forall o \in n'$ 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_{near} に追加する。そうでない場合、再帰的に処理を行う。

Algorithm 2: GetCandidates($q, w, \mathcal{S}$)

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

```

1  $N_c \leftarrow \emptyset, O_{near} \leftarrow \emptyset, H \leftarrow \emptyset$ 
2 Push root node of  $R[w]$  into  $H$ 
3 while  $H \neq \emptyset$  do
4    $n \leftarrow$  the node popped from  $H$ 
5   if  $n$  is a leaf node then
6      $N_c \leftarrow N_c \cup \{n\}$ 
7   else
8     for  $\forall n' \in N$  //  $N$  is  $n$ 's children do
9       if  $c \cdot d_{lb}(\mathcal{S}, n') \leq d_{ub}(\mathcal{S}, o_q)$  then
10        if  $d_{ub}(\mathcal{S}, n') < d_{lb}(\mathcal{S}, o_q)$  then
11          for  $\forall o \in n'$  do
12             $O_{near} \leftarrow O_{near} \cup \{o\}$ 
13        else
14          Push  $n'$  into  $H$ 
15 return  $N_c, O_{near}$ 

```

4.2 アルゴリズム

本節では、定理 2 および 3 に基づいて、クエリの解を効率的に検索するアルゴリズムの詳細を紹介する。アルゴリズム 3 は、提案アルゴリズムを示している。近似逆 k 最近傍クエリ $q = (o_q, k, c)$ が与えられたとき、各キーワード $w \in o_q.t$ を含むすべてのサブスクリプションのグループ $S_g \in G[w]$ に対して以下の処理を行う。まず、 S_g に対して、GetCandidates を実行し、 $S_g.N_c$ および $S_g.O_{near}$ を取得する (4 行)。そして、 $|S_g.O_{near}|$ が k 以上である場合、 S_g に含まれるすべてのサブスクリプションは解に含まなくてもよいため処理を終了する。 k 未満である場合、各サブスクリプション $s \in S_g$ に対して以下の処理を実行する。まず、各ノード $n \in S_g.N_c$ に対して、GetCandidates と同様の処理を行い、 $s.N_c$ および $s.O_{near}$ を取得する (8–16 行)。そして、 $|s.O_{near}|$ が k 以上である場合、 s は解に含まなくてもよいため処理を終了する。 k 未満である場合、 w を除く s の各キーワード $w' \in s.t \setminus \{w\}$ に対して、 $R[w']$ から、 $s.N_c$ および $s.O_{near}$ を取得する (17–21 行)。その結果、 $|s.O_{near}|$ が k 未満である場合、 s は解に含まれる可能性があるため、 s が解に含まれるかどうかチェックを行う関数 Verification を実行する (22–23 行)。Verification は、 $s.O_{near}$ および $s.N_c$ に含まれるすべてのオブジェクトの中で s に対して o_q より近い位置に存在するオブジェクトの数が、 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。これらの処理を、解となり得るサブスクリプションに対して行うことで効率的に解を検索することが可能である。

Algorithm 3: Proposed

Input: $q = (o_q, k, c)$

```

1  $ARkNN \leftarrow \emptyset$ 
2 for  $\forall w \in o_q.t$  do
3   for  $\forall S_g \in G[w]$  do
4      $S_g.N_c, S_g.O_{near} \leftarrow \text{GetCandidates}(q, w, S_g)$ 
5     if  $|S_g.O_{near}| < k$  then
6       for  $\forall s \in S_g$  do
7          $s.N_c \leftarrow \emptyset, s.O_{near} \leftarrow S_g.O_{near}$ 
8         for  $\forall n \in S_g.N_c$  do
9           if  $c \cdot d_{lb}(s, n) \leq d_{ub}(s, o_q)$  then
10            if  $d_{ub}(s, n') < d_{lb}(s, o_q)$  then
11              for  $\forall o \in n$  do
12                 $s.O_{near} \leftarrow s.O_{near} \cup \{o\}$ 
13              if  $|s.O_{near}| \geq k$  then
14                break
15            else
16               $s.N_c \leftarrow s.N_c \cup \{n\}$ 
17          if  $|s.O_{near}| < k$  then
18            for  $\forall w' \in s.t \setminus \{w\}$  do
19               $s.N_c, s.O_{near} \leftarrow s.N_c, s.O_{near} \cup$ 
20                 $\text{GetCandidates}(q, w', s)$ 
21              if  $|s.O_{near}| \geq k$  then
22                break
23          if  $|s.O_{near}| < k$  then
24             $ARkNN \leftarrow ARkNN \cup$ 
25               $\text{Verification}(q, s, s.N_c, s.O_{near})$ 
26 return  $ARkNN$ 

```

ここで、提案アルゴリズムは、 $|s.O_{near}|$ が k 未満となる s をクエリの解とするため、 o_q を $s.kNN$ に含むサブスクリプションは必ず解に含まれる。また、以下の定理が成り立つ。

定理 4. 提案アルゴリズムは、 c が増加すると高速化する。

証明. c が増加すると、アルゴリズム 2 の 9 行目、およびアルゴリズム 3 の 9 行目の条件を満たすノードの数が減少する。その結果、その後の処理を行う回数が減少するため、定理が成り立つ。 $\square$

さらに、 c が増加することでフィルタリングされるノードの中には、 $s.O_{near}$ に含まれるオブジェクトを含むノードも存在する可能性がある。そのため、本来、それらのオブジェクトを含むことで $|s.O_{near}|$ が k 以上となるサブスクリプションもクエリの解に含まれる場合が存在する。よっ

表 1: データセットの詳細

データセット	TWEETS	PLACES
オブジェクトの数	20,000,000	9,400,000
キーワードの数	2,145,092	54,044
あるオブジェクトに含まれる 平均のキーワードの数	5.2	2.9

表 2: パラメータの設定

パラメータ	値
オブジェクトの数, $ O [\times 10^6]$	1.0 , 2.0, 3.0, 4.0, 5.0
サブスクリプションの数, $ S [\times 10^6]$	0.25 , 0.5, 0.75, 1.0
k	5, 10 , 15, 20, 25, 30
近似率, c	1.0, 1.5 , 2.0, 2.5, 3.0, 3.5, 4.0

て、提案アルゴリズムにおいて、 c が増加すると、解となるサブスクリプションの数は増加する。

5. 評価実験

本章では、提案アルゴリズムの性能評価のために行った実験の結果を紹介する。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。

5.1 セッティング

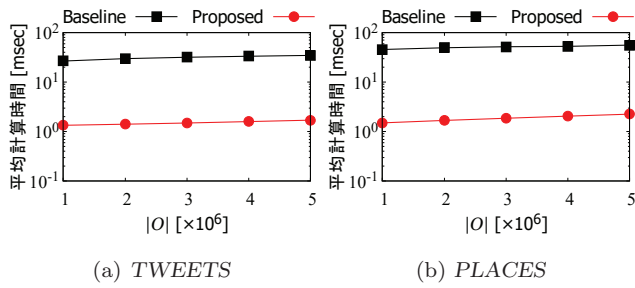
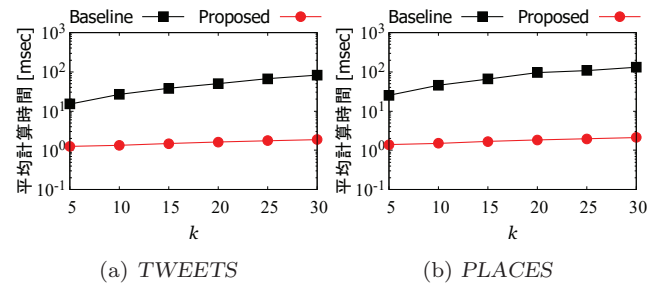
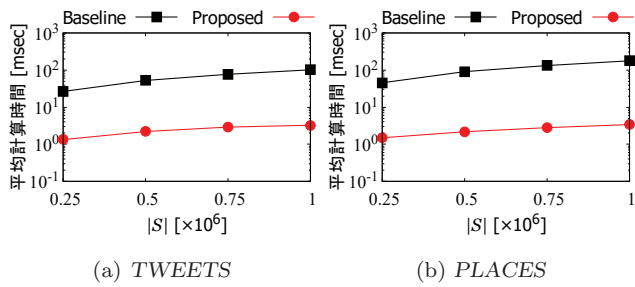
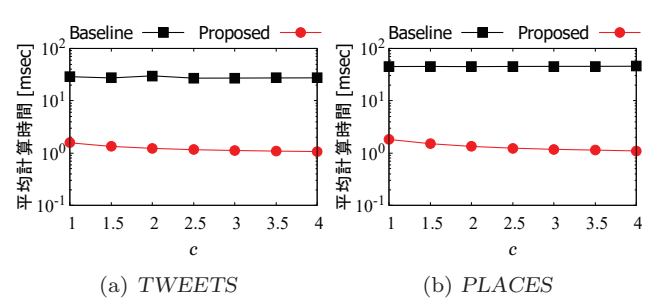
データセット. 本実験では、オブジェクト集合として 2 つの実データ (TWEETS [2] および PLACES [8]) を用いた。表 1 に、データセットの詳細を示す。各サブスクリプションは、1 つのオブジェクトをランダムに選択し、オブジェクトの位置をサブスクリプションの位置、オブジェクトのキーワード集合の中から 5 個以下の単語をランダムに選び、サブスクリプションのキーワード集合とする。

パラメータ. 表 2 は、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。

5.2 評価結果

各アルゴリズムにおける平均計算時間（ある近似逆 k 最近傍クエリの解を検索するまでの平均時間 [msec]）の結果を示す。各結果は、100,000 個のクエリの解を検索した際の平均の値である。

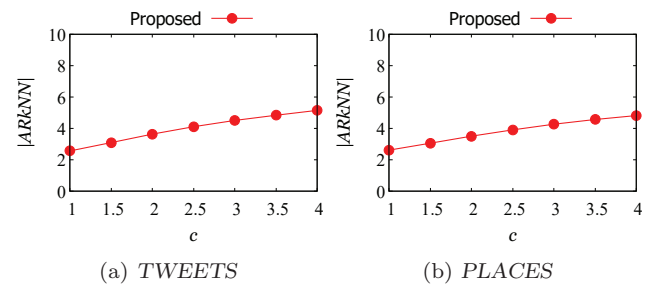
$|O|$ の影響. 図 2 に $|O|$ を変えたときの結果を示す。提案アルゴリズムはベースラインアルゴリズムと比較して、高速に解を検索できることがわかる。また、 $|O|$ の増加に伴い、各アルゴリズムにおいて計算時間が大きくなる。これは、 $|O|$ が増加すると、各キーワードを含むオブジェクトの数が増加するためである。 $|O|$ が増加すると、ベースラインアルゴリズムでは、RetrieveNearestObject の実行時間が増加する。一方、提案アルゴリズムでは、解となり得る

図 2: $|O|$ の影響図 4: k の影響図 3: $|S|$ の影響図 5: c の影響 (計算時間)

サブスクリプションの数が減少するため、Verification を実行する回数は減少するが、各サブスクリプションのグループに対して、評価する R-tree のノードの数が増加するため、計算時間がわずかに増加する。

$|S|$ の影響. 図 3 に $|S|$ を変えたときの結果を示す. 提案アルゴリズムはベースラインアルゴリズムと比較して、高速に解を検索できることがわかる. また、 $|S|$ の増加に伴い、各アルゴリズムにおいて計算時間が大きくなるが、提案アルゴリズムの方が、計算時間の増加率が小さい. ベースラインアルゴリズムは、解となり得るすべてのサブスクリプションに対して 1 つずつ処理を行うため、 $|S|$ が増加すると、計算時間は線形に増加する. 一方、提案アルゴリズムは、サブスクリプションをグループに分割し、複数のサブスクリプションをまとめて評価するため、計算時間の増加を抑えることができる.

k の影響. 図 4 に k を変えたときの結果を示す. 提案アルゴリズムはベースラインアルゴリズムと比較して、高速に解を検索できることがわかる. また、 k の増加に伴い、両アルゴリズムの更新時間の差は大きくなり、 k が大きい場合、提案アルゴリズムはより有用であることが分かる. さらに、2 つのデータセットにおける結果を比較すると、PLACES を用いたとき、わずかに計算時間が大きい. これは、PLACES の方がキーワードの種類が少なく、各オブジェクトおよびサブスクリプションは同じキーワードを含みやすく、あるクエリに対して解となり得るサブスクリプションの数やあるサブスクリプションに対して kNN となり得るオブジェクトの数が増加するためである.

図 6: c の影響 (解の数)

c の影響. 図 4 に c を変えたときの結果を示す. 提案アルゴリズムはベースラインアルゴリズムと比較して、高速に解を検索できることがわかる. また、 c が増加すると、ベースラインアルゴリズムの計算時間はほぼ一定であるのに対し、提案アルゴリズムの計算時間は小さくなる. これは、提案アルゴリズムにおいて、 c が増加すると、定理 3 の条件を満たすノードの数が増加し、より多くのオブジェクトのチェックを行う必要がなくなるためである.

さらに図 6 は、 c を変えたときの提案アルゴリズムにおける解となるサブスクリプションの数 $|ARkNN|$ の変化を示している. c の増加に伴い、解となるサブスクリプションの数が増加することが分かる、つまり、提案アルゴリズムは、 c が増加すると、解となるサブスクリプションの数が増加するがそれらをより高速に検索することが可能である.

6. まとめ

近年、位置情報およびキーワードを含むオブジェクトおよびサブスクリプションが大量に生成されている。本稿では、位置およびキーワードに基づく近似逆 k 最近傍クエリを新たに提案した。近似逆 k 最近傍クエリは、クエリオブジェクト o_q を kNN に含むサブスクリプションに加えて、 kNN に含まないが $d(s, o_q) < c \cdot d(s, o_k)$ を満たすようなサブスクリプション s も解に含めてもよい。近似逆 k 最近傍クエリの解を効率的に検索するため、オブジェクト集合およびサブスクリプション集合をキーワードごとに位置の近いもの同士でグループに分割し、クエリが与えられたとき、解となり得るサブスクリプションを限定し、それらのみチェックを行うアルゴリズムを提案した。実データを用いた評価実験の結果から、提案アルゴリズムの有効性を確認した。

謝辞. 本研究の一部は、文部科学省科学研究費補助金・基盤研究 (A)(18H04095), および基盤研究 (B)(T17KT0082a) の研究助成によるものである。ここに記して謝意を表す。

参考文献

- [1] Bentley, J.L.: Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9), 509–517 (1975)
- [2] Chen, L., Cong, G., Cao, X., Tan, K.L.: Temporal spatial-keyword top-k publish/subscribe. In: *ICDE*. pp. 255–266 (2015)
- [3] Choudhury, F.M., Culpepper, J.S., Sellis, T., Cao, X.: Maximizing bichromatic reverse spatial and textual k nearest neighbor queries. *PVLDB* 9(6), 456–467 (2016)
- [4] Gao, Y., Qin, X., Zheng, B., Chen, G.: Efficient reverse top-k boolean spatial keyword queries on road networks. *IEEE Transactions on Knowledge and Data Engineering* 27(5), 1205–1218 (2014)
- [5] Hidayat, A., Cheema, M.A., Taniar, D.: Relaxed reverse nearest neighbors queries. In: *SSTD*. pp. 61–79 (2015)
- [6] Hidayat, A., Yang, S., Cheema, M.A., Taniar, D.: Reverse approximate nearest neighbor queries. *TKDE* 30(2), 339–352 (2018)
- [7] Lu, J., Lu, Y., Cong, G.: Reverse spatial and textual k nearest neighbor search. In: *SIGMOD*. pp. 349–360 (2011)
- [8] Mahmood, A.R., Aly, A.M., Aref, W.G.: Fast: Frequency-aware indexing for spatio-textual data streams. In: *ICDE*. pp. 305–316 (2018)
- [9] Wang, X., Zhang, W., Zhang, Y., Lin, X., Huang, Z.: Top-k spatial-keyword publish/subscribe over sliding window. *The VLDB Journal* 26(3), 301–326 (2017)
- [10] Wang, X., Zhang, Y., Zhang, W., Lin, X., Wang, W.: Ap-tree: efficiently support location-aware publish/subscribe. *The VLDB Journal* 24(6), 823–848 (2015)
- [11] Zhao, J., Gao, Y., Chen, G., Jensen, C.S., Chen, R., Cai, D.: Reverse top-k geo-social keyword queries in road networks. In: *ICDE*. pp. 387–398 (2017)