情報処理学会論文誌　データベース　Vol.6　No.3　29–39 (June 2013)

ソーシャルサーチのための効率的な検索アルゴリズムの提案

三浦 大樹[1,†1,a)]　諏訪 博彦[1]　鳥海 不二夫[2]　鬼塚 真[3]

受付日 2012年12月20日, 採録日 2013年4月13日

概要：ソーシャルサーチには，クエリに対する文書のヒット数やユーザ数の増加にともない，検索結果が確定するまでの応答時間が遅くなるという問題がある．本論文では，その問題に対処するための効率的な top-k 検索アルゴリズムとして，3 つのアルゴリズムを提案する．1 つ目は，全文書から構築した 1 つの転置ファイルを利用する Single Index アルゴリズムである．2 つ目は，ユーザごとに分割した文書から構築した転置ファイルをソーシャルグラフに基づき接続して利用する Social Index Graph アルゴリズムである．3 つ目は，Single Index アルゴリズムと Social Index Graph アルゴリズムをヒット数を基準に切り替えるハイブリッドアルゴリズムである．Twitter のデータを用いて性能評価を行った結果，文書のヒット数が小さい場合には Single Index アルゴリズムが高速であり，ヒット数が大きい場合には Social Index Graph アルゴリズムが高速であることを示した．さらに，ハイブリッドアルゴリズムにおける 2 つのアリゴリズムの妥当な切替え基準を確認した．

キーワード：情報検索，ソーシャルサーチ，ソーシャルネットワーク

Efficient Algorithms for Personalized Social Search

Hiroki Miura[1,†1,a)]　Hirohiko Suwa[1]　Fujio Toriumi[2]　Makoto Onizuka[3]

Received: December 20, 2012, Accepted: April 13, 2013

Abstract: In this paper we consider efficient algorithms for top-k personalized social search, in which a document score is synthesized from the relevancy to the query and social closeness between the searcher and the author of the document. Since the score is synthesized from the two factors, there are three approaches for the social search; the first algorithm builds a unified single index that efficiently computes the document relevancy to the query, and the second algorithm builds a social index graph that efficiently determines the top-k documents based on the social closeness. The third algorithm switch between first and second algorithms on the basis of the number of documents hits. We use Twitter data to compare the efficiency of the two algorithms and show that there is a trade-off between them; the social index graph is superior to the single index approach when the hit ratio of the query is high, because the search space is efficiently reduced only to the documents whose authors are close to the searcher. From the result we show validity of the change rule on the third algorithm.

Keywords: information retrieval, social search, social network

1. はじめに

近年爆発的な利用の拡大が見られるソーシャルメディアでは，ユーザはフレンド関係やフォロー・フォロワ関係によって形成されるソーシャルネットワーク上の人のつながりに基づいて情報を取得している．ソーシャルメディア上の文書を対象とする検索では，クエリと文書の適合度と，文書作成者とのユーザ関係に基づくスコアを合成させたパーソナライズドソーシャルサーチによって検索精度の向上が

[1] 電気通信大学
University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[2] 東京大学
The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[3] NTT ソフトウェアイノベーションセンタ
NTT Software Innovation Center, Musashino, Tokyo 180–8585, Japan
[†1] 現在，NTT ソフトウェアイノベーションセンタ
Presently with NTT Software Innovation Center
[a)] miura.hiroki@lab.ntt.co.jp

考えられる．以前からソーシャルサーチの有用性を示す研究はなされており，検索者と作成者とのソーシャルグラフ上の距離に基づいて計算される *Friendship* [1] やユーザの類似度に基づく *Similarity-based network* とユーザの親密度に基づく *Familiarity-based network* [2] 等のソーシャルネットワーク上のユーザ関係に基づくスコアが提案されている．しかし，これらの研究の主眼はスコア精度の評価であり，現実的な応答時間の課題については言及していない．

ソーシャルサーチの検索結果を決定するまでの応答時間を短縮するためには，ユーザ関係に基づくスコアの計算と合成の効率化を実現する必要がある．Turtle ら [3] は，クエリを構成する複数の単語によって適合する文書集合を絞り込む際に，単語ごとの部分集合の中でもスコアの高い文書に対して，次の単語を適用させる *max-score optimaization* によって検索時の I/O コストを削減できることを示している．Brown [4] は，転置ファイルのヘッダ部に文書の適合度への寄与率が高いと予測される文書の位置情報を保持することで，検索結果の候補となりうる文書集合の絞り込みを行っている．

このように文書集合がより小さくなるようなスコアの計算順序を考えることはソーシャルサーチにおいても重要である．しかし，ユーザ関係スコアは検索単語や文書に含まれる単語に依存しないため，適合文書取得のための転置ファイルでは効率的な処理が難しい．また，スコアが検索者ごとに異なるため，すべての文書に対して任意の 2 ユーザ間のスコアを事前に関連付けることは，計算量やインデックスの容量の観点から現実的ではない．したがって，ソーシャルサーチの応答時間を短縮するためには，検索時にスコアの計算と合成を高速に処理できるアルゴリズムが必要であるが，これらに取り組んだ高速検索技術に関する研究は行われていない．

よって本論文では，ソーシャルネットワーク上のユーザ関係に基づくスコアを文書のランキングに利用するための効率的な top-k 検索アルゴリズムを提案する．提案するアルゴリズムは，Single Index アルゴリズム，Social Index Graph アルゴリズム，ハイブリッドアルゴリズムの 3 つのアルゴリズムである．Single Index アルゴリズムは，クエリと文書の適合度を効率的に計算できるアルゴリズムである．全文書から構築した 1 つの転置ファイルを用いて，検索単語によって文書集合を絞り込んだ後に，各文書に対してユーザ関係スコアを合成することで，検索結果のランキングを確定する．転置ファイルに作成者情報を付加することによって作成者の高速な特定を実現している．Social Index Graph アルゴリズムは，ユーザ関係スコアを効率的に計算できるアルゴリズムである．全文書を作成者ごとに分割して各作成者の転置ファイルを構築し，ユーザ関係に基づいてそれらの転置ファイルを接続したデータ構造を用いる．検索者を開始点としてそのグラフを探索することで，検索実行時にユーザ間の距離を計算できる．ハイブリッドアルゴリズムは Single Index アルゴリズムと Social Index Graph アルゴリズムの 2 つのアルゴリズムを，クエリにヒットする文書数の大小によって切り替えて利用するアルゴリズムである．Twitter のデータを用いて性能評価を行い，Single Index アルゴリズムと Social Index Graph アルゴリズムの性能特性を明らかにし，その結果から計算されるハイブリッドアルゴリズムの切替え基準が妥当であることを確認する．

本論文の構成は以下のとおりである．2 章では本論文が提案する top-k パーソナライズドソーシャルサーチのための 3 つの効率的なアルゴリズムの説明を行う．3 章では Twitter のデータを用いて，Single Index アルゴリズムと Social Index Graph アルゴリズムの性能比較と，ハイブリッドアルゴリズムの有効性の検証を行う．4 章では，我々が開発したソーシャルサーチシステムである，*Ego-Centric Social Network search engine* の説明とそのシステムを用いた検索例を提示する．5 章で関連研究を述べ，6 章では本論文の結論を述べる．

2. Top-k パーソナライズドソーシャルサーチ

2.1 文書のスコアリング

本研究では，文書のスコアを 3 つの指標を用いて計算する．その内訳は，クエリと文書の適合度に基づいて計算される Relevancy，ユーザ間の属性の類似度に基づいて計算される Similarity，ソーシャルネットワーク上のユーザ間の距離に基づいて計算される Familiarity，である．Similarity と Familiarity を合わせてユーザ関係スコアと呼ぶことにする．検索者ごとに異なる値であるユーザ関係スコアを利用することで，検索結果のパーソナライズを実現する．

検索者を u，クエリを q，文書を d，文書 d の作成者を $v(d)$ とし，式 (1) によって文書のスコアを計算する．

$$\begin{aligned} Score(u,q,d) = \alpha R(q,d) + (1-\alpha)\{\beta S(u,v(d)) \\ + (1-\beta)F(u,v(d))\} \end{aligned} \tag{1}$$

α は，$R(q,d)$ とユーザ関係スコア，β は $S(u,v(d))$ と $F(u,v(d))$ の線形和をとるための重みである．$R(q,d)$，$S(u,v(d))$，$F(u,v(d))$ は以下のように定義される．

- Relevancy 導入の目的は検索対象がテキスト情報であるためである．クエリ q と文書 d の適合度である TF-IDF [5] に基づいて，文書中のクエリ単語 t の出現頻度である $tf(t,d)$ と，クエリ単語 t を含む文書の出現頻度 $df(t)$ を用いて計算を行う．検索対象の全文書数を N とする．

$$R(q,d) = \sum_{t \in q} \left\{ \sqrt{tf(t,d)} \times \left(1 + \log \frac{N}{df(t)+1} \right) \right\} \tag{2}$$

- Similarity の導入の目的は，ユーザ間の属性は似ているが直接つながりのないユーザ同士の情報を取得できるようにするためであり，検索者 u と文書 d の作成者 $v(d)$ とのユーザプロファイルの類似度に基づいて計算される．利用するユーザプロファイルは性別や年代，興味・関心・話題等，様々なものが考えられる．
  検索者 u のユーザプロファイル集合を U，文書の作成者 $v(d)$ のユーザプロファイル集合を V として，U, V の Jaccard 係数を用いる．Similarity $S(u, v(d))$ は以下のように定義される．

$$S(u, v(d)) = \frac{|U \cap V|}{|U \cup V|} \tag{3}$$

- Familiarity 導入の目的は，友人の情報や友人の友人の情報を上位にランキングさせるためであり，検索者 u と文書 d の作成者 $v(d)$ とのソーシャルネットワーク上の距離に基づいて計算される．ソーシャルネットワークについても，フレンド関係，フォロー/フォロワ関係や，リプライ/コメントのようなコミュニケーションの有無等，様々な観点から構築することが可能である．
  議論の簡略化のため，本研究ではエッジの重みをすべて 1 とした場合の検索者 u と文書 d の作成者 $v(d)$ との距離 $hop(u, v(d))$ を用いるが，重みありエッジの場合へのアルゴリズムの拡張は容易である．Familiarity $F(u, v(d))$ は以下のように定義される．

$$F(u, v(d)) = \frac{1}{\log\left(hop(u, v(d)) + 1\right)} \tag{4}$$

2.2　検索アルゴリズム

本論文のパーソナライズドソーシャルサーチは top-k 検索であり，式 (1) のスコアに基づいてランキングされた上位 k 件の文書を検索結果と定める[*1]．検索処理はクエリにヒットする文書を取得し，$R(q, d)$ を計算する *Index lookup* と，$S(u, v(d))$ と $F(u, v(d))$ を計算し，式 (1) に従ってスコアの合成計算を行う *Personalization* の 2 つのステップに分割して考えることができる．それぞれのステップの処理には多様な方法が考えられるが，本論文では高速に処理が可能と考えられる，Single Index アルゴリズムと Social Index Graph アルゴリズム，その 2 つを組み合わせたハイブリッドアルゴリズムを提案する．

2.2.1　Single Index アルゴリズム

Single Index アルゴリズムはクエリと文書の適合度である Relevancy を効率的に計算できるアルゴリズムである．全作成者の全文書から構築した 1 つの転置ファイルを用いて検索単語による文書集合の絞り込みを行った後に，各文書に対してユーザ関係スコアである Similarity と Familiarity を合成することで検索結果のランキングを確定する．

[*1] 一般にユーザは検索結果のうち上位のものしか見ないことが示されている [6].

Algorithm 1 Single Index アルゴリズム

Require: u : searcher, q : query, k : # of highest documents
Ensure: $docs$: top-k documents
1: $docs \leftarrow$ empty priority queue
2: $D(q) \leftarrow getDocs(q)$
3: **for all** $d \in D(q)$ **do** {in descending order of $R(q, d)$}
4: $v \leftarrow author(d)$
5: $Score(u, q, d) \leftarrow \alpha R(q, d) + (1 - \alpha)\{\beta S(u, v) + (1 - \beta)F(u, v)\}$
6: $\overline{Score}(u, q, d) \leftarrow \alpha R(q, d) + (1 - \alpha)\{\beta \overline{S}(u, v) + (1 - \beta)\overline{F}(u, v)\}$
7: **if** $docs.size < k$ **or** $docs.kthscore < Score(u, q, d)$ **then**
8: $docs \leftarrow (d, Score(u, q, d))$
9: **else if** $docs.kthscore > \overline{Score}(u, q, d)$ **then**
10: $break$
11: **end if**
12: **end for**

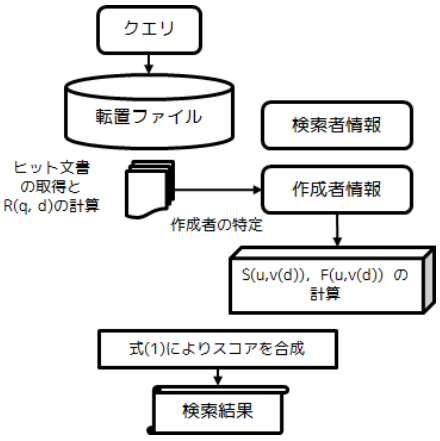

図 1　Single Index アルゴリズム検索処理イメージ
Fig. 1　Schematic of the Single Index Algorithm.

図 1 は，Single Index アルゴリズムにおける top-k パーソナライズソーシャルサーチの手順を模式的に表している．検索は以下のように実行される．1) 検索対象になる文書集合 D から取得した文書 d について $R(q, d)$ を計算する．2) 文書 d の作成者 $v(d)$ を特定する．3) 検索者 u と特定した作成者 $v(d)$ を基に，$S(u, v(d))$，$F(u, v(d))$ を計算する．4) 式 (1) によってスコアの合成を行い，上位 k 件の文書を確定する．

このアルゴリズムでは，文書の作成者の特定に性能ボトルネックがある．文書の作成者の特定には，転置ファイルの各文書 ID に文書の作成者情報を付加する拡張によって高速化を実現している．一般的な転置ファイルでは，文書 ID とスコアの組である，<docid,tf, 単語の位置情報 > というエンティティを基にランキングされるが，転置ファイルに文書の作成者の情報である authorid を埋め込み <docid,tf,authorid, 単語の位置情報 > と拡張を行うことで，ユーザ関係スコアの取得時にインデックスアクセスが不要になり，高速な文書作成者の特定が可能になる．

Algorithm 1 は検索アルゴリズムの詳細な手順を表し

Algorithm 2 Social Index Graph アルゴリズム

Require: u : searcher, q : query, k : # of highest documents
Ensure: $docs$: top-k documents

```
 1: docs ← empty priority queue
 2: Nodes ← empty queue
 3: Visited ← empty list {visited node list}
 4: u → Nodes, Visited
 5: while Nodes not empty do
 6:    w ← Node
 7:    for neighbor v of w do
 8:       if v ∉ Visited then
 9:          v → Visited, Node
10:          D(v,q) ← getDocDivIdx(v,q)
11:          for all d ∈ D do {in descending order of R(q,d)}
12:             Score(u,q,d) ← αR(q,d)+(1−α){βS(u,v)+(1−
                β)F(u,v)}
13:             Score‾(u,q,d) ← αR‾(q,d)+(1−α){βS‾(u,v)+(1−
                β)F(u,v)}
14:             if docs.size < k or docs.kthscore < Score(u,q,d)
                then
15:                docs ← (d, Score(u,q,d))
16:             else if docs.kthscore > Score‾(u,q,d) then
17:                return
18:             end if
19:          end for
20:       end if
21:    end for
22: end while
```

ている．上位 k 件を格納するキュー $docs$ を用意する（行 1）．文書集合 D に対して構築された転置ファイルを用いて $R(q,d)$ を計算する（行 2）．文書 d を $R(q,d)$ の値の降順で取得し各 d ごとに作成者を特定する（行 4）．ユーザ関係スコア $S(u,v(d))$，$F(u,v(d))$ の計算と，スコア合成を行い $Score(u,q,d)$ を計算する（行 5）．同時に，$S(u,v(d))$，$F(u,v(d))$ の上限値に基づいて $Score(u,q,d)$ の上限値 $\overline{Score}(u,q,d)$ を計算する（行 6）．ここで，$\overline{Score}(u,q,d)$ を用いて Threshold Algorithm [7] を適用することで，計算の打ち切り判定を行い上位 k 件の検索結果を確定する（行 7–11）．

このアルゴリズムの利点は，全文書に対するクエリと文書の適合度 $R(q,d)$ を 1 度に計算できることである．作成者情報を付与した転置ファイルからの文書作成者の高速な特定を行うことが可能であり，$R(q,d)$ について降順に並べられたヒット文書に対して，順番に式 (1) に基づいてスコアの合成を行うことができる．しかし，top-k 文書の作成者の $S(u,v(d))$，$F(u,v(d))$ の値が大きい場合等にはこの利点がうまく機能しない．$S(u,v(d))$，$F(u,v(d))$ の値についてはソートして格納されていないので，検索者から距離が遠い作成者のスコア計算を省略することができない．

2.2.2 Social Index Graph アルゴリズム

Social Index Graph アルゴリズムは，検索者と作成者との距離に基づいて計算される Familiarity を効率的に計算できるアルゴリズムである．全文書を作成者ごとに分割してそれぞれに対して転置ファイルを構築し，それらの転置ファイルをソーシャルネットワークのユーザ関係に従ってグラフ構造に接続したデータ構造を用いる．検索者を中心として幅優先でソーシャルグラフを探索するのと同時にユーザ関係スコアの計算をし，探索した作成者の転置ファイルを用いて適合文書の Relevancy の計算とスコアの合成を行う．

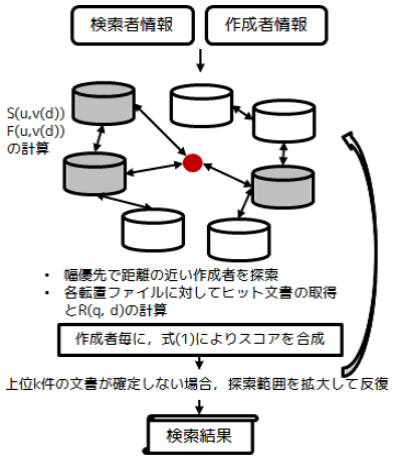

図 2　Social Index Graph アルゴリズム検索処理イメージ
Fig. 2　Schematic of the Social Index Graph Algorithm.

図 2 は，Social Index Graph アルゴリズムを用いた top-k パーソナライズソーシャルサーチの手順を模式的に表している．検索は以下のように実行される．1) 検索者を中心として，幅優先で距離 1 の作成者の転置ファイルを探索する．2) 探索した作成者 v の $S(u,v(d))$，$F(u,v(d))$ を計算する．3) 作成者 v の文書集合 D から取得した文書 d について $R(q,d)$ の計算とスコアの合成を行う．4) 上位 k 件の文書が確定しない場合は，さらに探索範囲を広げて反復する．

Algorithm 2 は Social Index Graph アルゴリズムによる検索処理の詳細を表している．訪問予定の作成者と訪問済みの作成者を格納する $Nodes$，$Visited$ を用意し，ソーシャルグラフを幅優先で探索する（行 1–7）．作成者 v が未訪問ならば $Visited$ に格納し，作成者 v の転置ファイルに対して検索処理を行い，$R(q,d)$ を計算する（行 8–10）．取得した文書 d に対して，スコア $Score(u,q,d)$ を計算すると同時に，$R(q,d)$ の上限に基づいて，上限スコア $\overline{Score}(u,q,d)$ を計算する（行 11–13）．Threshold Algorithm を適用し，上位 k 件の検索結果を確定する（行 14–18）．この行程は打ち切りが発生しない限りすべてのノードを訪問するまで行われる．

このアルゴリズムの利点は，検索者を中心に幅優先でグラフを探索することで距離の近い作成者を探索し，$F(u,v(d))$，$F(u,v(d))$ の計算を行うことが可能な点である．あらかじめ作成者ごとに文書が分割されているので，文書ごとの作成者の特定を必要としない．また，$R(q,d)$ を計算する

際に用いる $df(t)$ は全文書に対する値を用いる必要があるが，Social Index Graph アルゴリズムでは，転置ファイルを作成者ごとに構築するため，構築時に全文書数と $df(t)$ の値を保持しておく．Threshold Algorithm は $S(u, v(d))$, $F(u, v(d))$ を基準に行われ，検索者から距離の近い作成者から順に効率的な top-k 文書の検索が可能である．

2.2.3　ハイブリッドアルゴリズム

このアルゴリズムは，2.2.1 項で説明した Single Index アルゴリズムと 2.2.2 項で説明した Social Index Graph アルゴリズムの 2 つを，切り替えて使用することによって各アルゴリズムの利点を利用するアルゴリズムである．Single Index はヒット数が大きくなるにつれ，応答時間が大きくなると考えられる．これはヒットした文書の数だけスコアの合成計算を行う必要があることに起因する．Social Index Graph は，ヒットする文書が多いほど，作成者の近隣で top-k 文書が確定する可能性が高いので，ヒット数が大きくなるにつれ応答時間が小さくなることが考えられる．

このアルゴリズムで重要な点は 2 つのアルゴリズムを切り替える基準の決定である．まず学習ユーザを選出し，複数のヒット数が異なるクエリに対して応答時間を計測する．その結果を用いて，それぞれのアルゴリズムについて，応答時間を従属変数，クエリにヒットする文書数を独立変数ととるような回帰分析を行い，2 つのアルゴリズムの性能が逆転すると推定される，双方の回帰直線の交点を切替えの基準ヒット数とする．

このアルゴリズムでは Single Index アルゴリズムで用いる 1 つの大きな転置ファイルと，Social Index Graph アルゴリズムで用いる，ユーザ関係が関連付けられている分割された転置ファイルの 2 種類の転置ファイルを作成する必要があり，どちらか一方と比較して転置ファイルの容量が大きくなる．しかし，応答時間については，Single Index アルゴリズムが不利なヒット数が多い場合と，Social Index Graph が不利なヒット数が少ない場合のそれぞれの欠点を補えるため，クエリに対するヒット数に依存せず，高速な top-k パーソナライズソーシャルサーチが可能である．また実際の検索エンジンでは，スループットを向上させる目的あるいはサービスを停止させないように転置ファイルを含めて多重化することが一般的である．転置ファイルを多重化する際にハイブリッドアルゴリズムを適用することで，多重化すると同時に応答時間を向上させることが可能になる．

表 1　各アルゴリズムの計算量
Table 1　Complexity of Algorithms.

アルゴリズム	計算量
Single Index	$O(w \log T + h + l \log k)$
Social Index Graph	$O(m(w \log T + \frac{h}{n}) + m\frac{h}{n} \log k)$

2.2.4　アルゴリズムの計算量

全文書の単語数を T，クエリに含まれる単語数を w，ヒット数を h，打ち切りまでに計算した文書数を l，全ユーザ数を n，打ち切りまでに探索した作成者の数を m とすると，検索処理全体の計算量は**表 1** のように表される．

Single Index アルゴリズムでは，クエリに含まれる各単語ごとにサイズが T であるインデックスを探索して該当の単語にヒットする文書集合を特定し（コストが $w \log T$），クエリにヒットする全文書を取得する（コストが h）．ヒットした文書ごとにスコア合成を行い Threshold Algorithm を用いて上位 k 件の文書を特定する（コストが $l \log k$）．一方，Social Index Graph アルゴリズムでは，全文書は n 分割されるため，1 人の作成者の転置ファイルでクエリにヒットする文書の数は平均 $\frac{h}{n}$ となる．最終的に計算する文書数は Threshold Algorithm により作成者 m 人分となる．h が大きいとき，$m << n$ となり，全体の計算量が小さくなる．

3.　性能評価

性能評価では Single Index アルゴリズムと Social Index Graph アルゴリズムの 2 つのアルゴリズムの応答時間を計測することで各アルゴリズムの性能特徴を示す．またその結果を用いてハイブリッドアルゴリズムの切替え基準ヒット数の妥当性を確認する．

3.1　データセット

評価に用いるのは 2011 年 3 月 5 日から 3 月 24 日に投稿された Twitter のツイートデータである．本実験では，メンション（投稿に含まれる *@ScreenName*）を基にソーシャルネットワークを構築する．たとえば，ユーザ A が @ユーザ B の含まれる記事を投稿した場合，ユーザ A からユーザ B に対して有向エッジが張られることになる．ただし，公式リツイートと非公式リツイートは含まない．データの詳細を**表 2** に示す．1 ツイートに含まれる文字数は平均 51.4 文字である．

各アルゴリズムで用いるインデックスのサイズを**表 3** で示す．通常の転置ファイルは，一般的に全文検索に用いるために構築するために作成されるものである．Single Index アルゴリズムで利用するインデックスの内訳は，転置

表 2　Twitter データの詳細
Table 2　Overview of data sets of Twitter.

文書数	ユーザ数	エッジ数	平均エッジ数
69 M	108 K	1.3 M	12.8

表 3　各アルゴリズムのインデックスのサイズ
Table 3　Size of indexes.

通常の転置ファイル	Single Index	Social Index Graph
14.8 GB	25.7 GB	20.3 GB

ファイル 17.4 GB，事前計算した $S(u, v(d))$ と $F(u, v(d))$ のデータ 8.3 GB であり，合計 25.7 GB である．Social Index Graph アルゴリズムで利用するインデックスの内訳は，転置ファイル 20.1 GB，ソーシャルグラフ 15 MB，ユーザのプロファイル 150 MB の合計 20.3 GB である．Single Index アルゴリズムでは転置ファイルに文書の作成者情報を付与しているため，通常の転置ファイルに比べて転置ファイルのサイズが大きくなる．Social Index アルゴリズムではユーザごとに文書を分割して転置ファイルを構築するため，Single Index アルゴリズムと比較して転置ファイルのサイズが約 15% 増加しているが，$S(u, v(d))$ と $F(u, v(d))$ の事前計算が不要なため，全体としては約 20%サイズが小さくなっている．

応答時間の測定には，検索者として全ユーザからランダムに 1,000 ユーザを選択し，検索クエリは全文書中から 5,000〜100 万ヒットするような単語 100 語を選択した．例として，**表 4** にヒット数上位 20 件の単語を示す．top-k 検索の上位件数は $k = 100$ とした．k の値は Threshold Algorithm によるスコアの合成計算で省略できる文書数と関係するが，全文書数に対して k の値が十分に小さい場合は Single Index アルゴリズムと Social Index Graph アルゴリズムの両アルゴリズムとも，ヒット文書数に対する応答時間の関係に変化は少ない．k の値が小さいことは，検索結果の文書数が少ないということを意味しており，Single Index アルゴリズムと Social Index Graph アルゴリズムの両アルゴリズムとも検索結果確定までの応答時間が短縮される．k の値が大きい場合は逆に，両アルゴリズムとも処理する文書数が増加するため応答時間は遅くなる．

本実験では，ユーザの直近 200 件の投稿中に含まれる単語のおいて，助詞を除くため 2 文字以上の単語を抽出し，活用等の処理を行う．そのような単語の集合の中で出現頻度が多い上位 100 単語を，ユーザプロファイルと定める．このユーザプロファイル集合を基に Similarity を計算する．

本実験で用いた，表 4 のクエリにおいて，たとえば「水」というクエリを用いた場合の検索結果は，検索者からみて，Similarity や Familiarity の高い作成者の水に関する情報が現れる．本実験では，2011 年に発生した東日本大震災時の Twitter のデータを用いているため，断水，給水，等に関する情報が検索結果に多く現れると考えられる．

表 4　応答時間の測定に用いた検索単語（ヒット数上位 20 単語）
Table 4　Top 20 queries using for the experiment.

順位	単語						
1	原発	6	nhk	11	帰る	16	心配
2	日本	7	食べる	12	仕事	17	緊急
3	被災	8	無事	13	節電	18	携帯
4	東京	9	福島	14	電話	19	募金
5	水	10	避難	15	ニュース	20	報道

3.2　実験環境

実験に用いたハードウェアおよびソフトウェアの環境は**表 5** のとおりである．

3.3　結果

図 3 は，Single Index アルゴリズムと Social Index アルゴリズムを用いた場合のクエリにヒットした文書数と，応答時間を表している．クエリにヒットした文書数とは，全文書中で与えられたクエリにヒットする文書の件数を表しており，検索結果として定める上位 k 件の文書とは異なる数値である．Single Index アルゴリズムでは，クエリにヒットした文書数に対して応答時間は線形に増加する傾向が見られる．これは，文書のスコア計算の方法に起因するもので，Single Index アルゴリズムは，クエリにヒットする文書を最初にすべて取得し，それぞれの文書に対してスコアの合成計算を行うためである．Social Index Graph アルゴリズムでは，クエリにヒットした文書数の増加にともなって，応答時間の大幅な短縮が見られる．このアルゴリズムは，検索者を中心にソーシャルネットワークを幅優先で探索し，近隣の作成者の文書から検索を行う手法であり，上位 k 件に該当する文書の作成者が，近隣のユーザに限定される場合に，高速な検索が可能になる．全文書に占めるクエリにヒットする文書の量が多い場合，近隣の作成者がヒット文書の作成者である可能性が高くなるため，効果的な打ち切りを行うことが可能である．

図 4 は全体の応答時間を，クエリにヒットする文書の取得と $R(q, d)$ の計算を行う *Index lookup* と，$S(u, v(d))$ と

表 5　性能測定環境
Table 5　Environment measurement.

CPU	Intel Xeon 3.06 GHz
Memory	12.0 GB
OS	Windows 7
言語	Java SE 6
全文検索ライブラリ	Lucene 2.1
日本語形態素解析	Sen 1.2.2.1

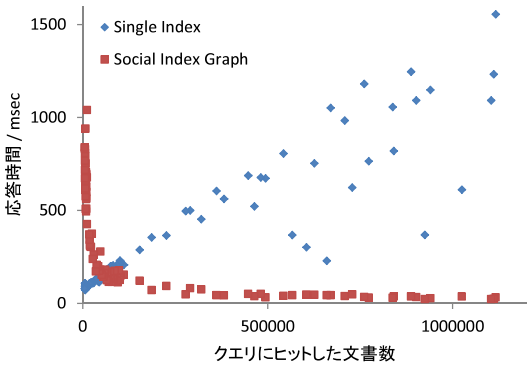

図 3　2 つのアルゴリズムの応答時間とクエリにヒットする文書数
Fig. 3　Response time and the number of documents.

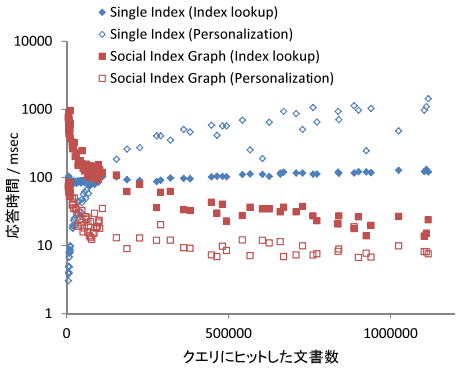

図 4　各アルゴリズムが Index lookup と Personalization に要する応答時間

Fig. 4　The response time that Index lookup and Personalization takes.

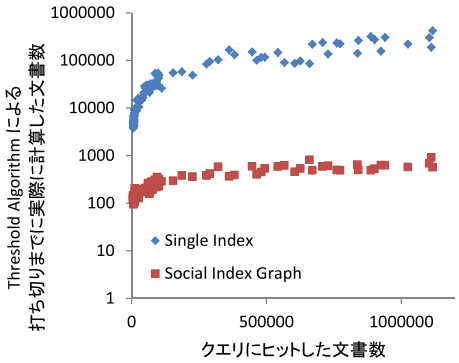

図 5　Threshold Algorithm による打ち切りまでに計算する文書数とクエリにヒットする文書数

Fig. 5　The number of the documents making a hit for the number of the documents and a query to calculate by a break by Threshold Algorithm.

$F(u, v(d))$ の取得とスコアの合成計算を行う *Personalization* の 2 つのステップに分割して，性能の内訳を表している．Single Index アルゴリズムでは，*Index lookup* の応答時間はヒット数が増加しても 100 msec 程度の安定した応答時間を保っているが，*Personalization* に関してはヒット数に従って大幅に増加している．Social Index Graph アルゴリズムについては，クエリに対するヒット数が少ない場合の *Index lookup* の応答時間は，広範囲のユーザを探索しなければならないため，遅くなっている．*Personalization* に関しては作成者の特定が不要であり，かつ $S(u, v(d))$, $F(u, v(d))$ の合成が高速であることと，Threshold Algorithm が効果的に機能しているため，応答時間が速い．

また，両アルゴリズムにとって重要な要素である Threshold Algorithm の効果についてさらに詳細な特徴を明らかにする．**図 5** は，2 つのアルゴリズムにおける Threshold Algorithm の効果を表している．Single Index アルゴリズムの場合，Threshold Algorithm の効果は，クエリにヒットする文書数に加えて，上位 k 件の文書のスコア分布にも依存するが，クエリにヒットする文書数に対してほぼ

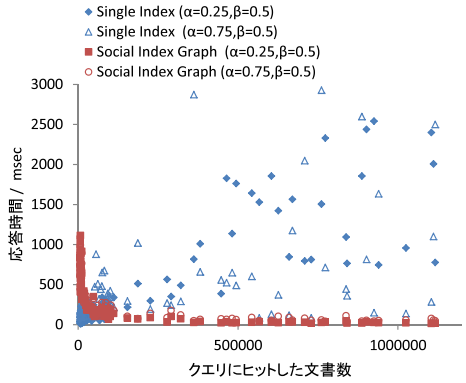

図 6　$\alpha = 0.25, 0.75, \beta = 0.5$ の場合の応答時間

Fig. 6　Response time ($\alpha = 0.25, 0.75, \beta = 0.5$).

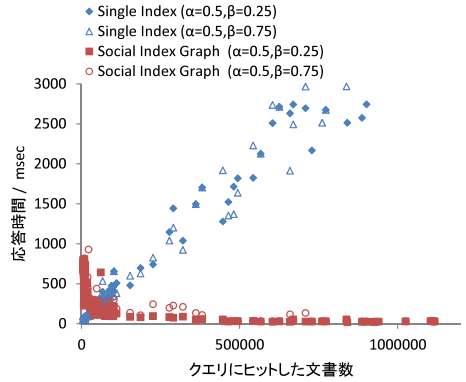

図 7　$\alpha = 0.5, \beta = 0.25, 0.75$ の場合の応答時間

Fig. 7　Response time ($\alpha = 0.5, \beta = 0.25, 0.75$).

線形に増加する．Social Index Graph アルゴリズムにおいては，Single Index アルゴリズムと比較して，Threshold Algorithm の効果が大きい．

式 (1) の重みである，α，β の値を変化させた場合の応答時間の測定を行う．**図 6** は，$\alpha = 0.25$，0.75，$\beta = 0.5$ の場合の応答時間の様子である．**図 7** は，$\alpha = 0.5$，$\beta = 0.25$，0.75 の場合の応答時間の様子である．線形和の重みである α，β に依存して各スコア値が変化するため，応答時間は各々異なるが，ヒット数が少ない場合に Sin Index アルゴリズムが高速であり，ヒット数が大きい場合に Social Index アルゴリズムが高速であるという性能特徴は変化しない．

3.4　ハイブリッドアルゴリズムにおける切替え基準ヒット数の検証

図 3 の結果より，クエリにヒットする文書数が少ない場合は，Single Index アルゴリズムが Social Index Graph アルゴリズムよりも応答時間が短く，クエリに対するヒット数が多い場合は Social Index Graph アルゴリズムが応答時間が長いことが確認できた．ハイブリッドアルゴリズムにおける Single Index アルゴリズムと Social Index Graph アルゴリズムの回帰直線の交点から推定される切替え基準ヒット数を h_s とおき，交差検証法を用いて h_s の妥当性を検証した．分割数は 10 とした．各グループで h_s を算出

表 6　各グループで推定された h_s と的中率 r の値
Table 6　Value of h_s and r estimated in each group.

	1	2	3	4	5	
h_s	59,739	59,779	58,288	60,147	57,953	
r	0.852	0.844	0.841	0.861	0.850	
	6	7	8	9	10	平均値
	56,857	62,193	56,912	59,481	58,784	58,940.9
	0.840	0.843	0.842	0.850	0.841	0.846

し，残りのグループのデータに対してテストを行った．クエリにヒットした文書数が h_s 未満の場合は Single Index アルゴリズムを用いた場合の応答時間が短く，h_s 以上の場合は Social Index Graph アルゴリズムの応答時間が短くなる場合の比率である，的中率 r を計算した．その結果を**表 6** に示す．

的中率 r の平均値は 0.846 であった．h_s，r ともに各グループ間の差は小さく，的中率も十分に大きいことから，回帰分析による切替え基準ヒット数の推定は妥当であると考えられる．よって，ハイブリッドアルゴリズムを用いることで，クエリにヒットする文書数に依存せずに top-k パーソナライズドソーシャルサーチが可能であるといえる．

4. 関連研究

ソーシャルサーチに関連する研究には，新たな指標を提案し検索精度を評価する研究と，応答時間の短縮や使用容量の削減等の検索性能を評価する研究の 2 つがある．それぞれ 4.1 節では指標に関する研究を，4.2 節では高速化技術に関する研究を説明する．

4.1 ソーシャルサーチに用いられるスコア

ソーシャルサーチに用いられる文書スコアの指標は，パーソナライズの有無で分類することができる．検索結果のパーソナライズがされない指標としては，ユーザのタグ付け行為に基づいたランキングアルゴリズムである，*FolkRank* [8]，*SBRank* [9]，*SocialPageRank*，*SocialSimRank* [10] 等が提案されている．ただしこれらは本研究の対象とするパーソナライズドソーシャルサーチに関する指標ではない．

検索結果のパーソナライズに関する指標は，主に検索者から見た他ユーザとのソーシャルグラフ上のユーザ関係を用いるものが多い．Bender ら [1] は，ユーザ間の距離に基づいて計算される *Friendship* という指標をソーシャルネットワーク対して *PageRank* の考え方を適用した *UserRank* と組み合わせて文書のスコアリングに用いることを提案している．また，Carmel ら [2] はユーザ同士の類似度と親密度の 2 つの異なるソーシャルネットワークに基づく指標を提案している．ユーザプロファイルの類似度を表す *Similarity-based network* とユーザ間の親密度を表す *Familiarity-based network* の 2 つのスコアによって検索結果のパーソナライズを行い，実データを用いて提案指標の有効性を検証している．本研究は今まで取り組まれていなかった，ユーザ間の関係を用いたパーソナライズドソーシャルサーチに対する検索処理の高速化に取り組むものである．

4.2 ソーシャルサーチの高速化

パーソナライズドソーシャルサーチの高速化を構成する技術として，スコア合成，転置ファイルを用いた適合文書の取得，ユーザ関係スコアの取得，の 3 点に分類することができる．それぞれの研究について以下で詳しく説明する．

複数の単語やスコアを利用する場合のスコア合成に関しては，単語の適用順序やスコアの計算順序に関する研究がなされている．Turtle ら [3] は，クエリを構成する複数の単語によって適合する文書集合を絞り込む際に，単語ごとの部分集合の中でもスコアの高い文書に対して，次の単語を適用させることで検索時の I/O コストを削減できることを示している．Brown [4] は，転置ファイルのヘッダ部に文書の適合度への寄与率が高いと予測される文書の位置情報を保持することで，検索結果の候補となりうる文書集合の絞り込みを行っている．ユーザ関係スコアは，クエリや文書の内容に依存しないことに加え，値が検索者ごとに異なるため，適合文書の取得のための転置ファイルでは効率的に処理できず，任意の 2 ユーザに関するスコアを関連付けて格納することは，計算量や容量の観点から現実的ではない．

転置ファイルを用いた適合文書取得の高速化についても様々な研究がなされている．Zobel ら [11] は転置ファイルに関する，検索モデル，圧縮手法，スキップリストや文書の格納順の工夫等の高速化技術について網羅的にサーベイしており，検索コストの削減には単語の出現頻度やスコアへの寄与率に従った転置ファイルの並べ替えが効果的であると述べている．その中でも，Anh ら [12], [13] の研究では，複数単語が選言的に指定される検索を効率的に実行できる転置ファイルの構築手法を提案している．適合度の高い文書順に転置ファイルを構築することによって，転置ファイルの探索の処理を足切りをすることで検索結果の高速な決定を実現している．Chen ら [14] は，Twitter のデータの新規性を重視した場合の効率的な転置ファイルの構築手法を提案している．転置ファイルに時間の情報を埋め込み，ブロック単位で検索を行うことで，新規性の高い情報を重視した文書の検索の高速化を実現している．これらの研究は，転置ファイルを構築する際に決められた優先度によって文書を格納する手法を提案している．

従来研究で提案されている手法と本研究で提案する手法との違いは検索者に依存する文書のスコアを用いている点である．従来研究の手法では，検索者に依存しない文書のスコアを利用しているため，すべての検索者に共通のデー

タを 1 つ持つことで実現できる．パーソナライズドソーシャルサーチでは，検索者ごとにスコアの値が変わるため，同じ手法を適用する場合，すべての文書に対して N^2（N はユーザ数）のスコアの情報を保持しなければならない．また，ソーシャルグラフに変更があった場合，そのすべての情報を基に転置ファイルの再構築が必要になる．そのため，パーソナライズドソーシャルサーチでは，検索時にユーザ関係スコアの計算と，スコア合成を行うため，この作業を高速に処理できるアルゴリズムが必要である．

本研究で提案する Single Index アルゴリズムでは，作成者の情報を転置ファイルに埋め込むことによって，作成者の高速な特定を可能にしている．また，Social Index Graph アルゴリズムでは，作成者ごとに分割され，ソーシャルグラフのユーザ関係によって接続された転置ファイルを用いることによって，検索範囲を距離が近い作成者に限定することで，高速な検索が可能である．またヒット数の大小によって効率的なスコアの計算順序が異なるため，ハイブリッドアルゴリズムによって，2 つのアルゴリズムを切り替えることでヒット数によらずに高速なソーシャルサーチを実現している．

ユーザ関係スコアの取得コストも性能低下の要因である．人を検索対象とするソーシャルサーチエンジンである Aardvark [15] では，構築したソーシャルグラフを転置ファイルに格納することで，検索者に対するユーザ関係スコアの高速な取得を実現している．本研究では事前計算が必要な Single Index アルゴリズムでは文献 [16] の手法を用いて 2 ユーザ間の距離を求めている．Social Index Graph アルゴリズムでは検索時にユーザ間の距離を計算しているため，ユーザ関係スコアを格納するための特別な工夫は必要ない．

5. Ego-Centric Social Network search engine

我々はソーシャルサーチエンジン『Ego-Centric Social Network search engine』を開発した．このサーチエンジンは**図 8** のように検索結果を 2 種類の方法で表示する．文書の適合度のみによって計算されたスコアによるランキング（Nomal Search：左）と式 (1) によって計算されたスコアによるランキング（Social Search：右）の 2 つの検索結果が表示される『文書ランキング表示タブ』と，検索者と検索結果に現れる文書の作成者との関係を表すネットワークが表示される，『作成者ネットワーク表示タブ』の 2 つのタブから構成される．ノードの色はそれぞれユーザの種別を表しており，赤色は検索者，青色は Normal Search の結果に現れる文書の作成者，黄色は Social Search の結果に現れる文書の作成者，緑色は 2 つの検索結果の両方に現れる文書の作成者である．白色のノードは検索結果には現れないが，検索者と検索結果に現れるユーザを仲介するユーザである．ノードの大きさは検索者からのネットワーク上の距離を表しており，サイズが大きいほど距離が近い．

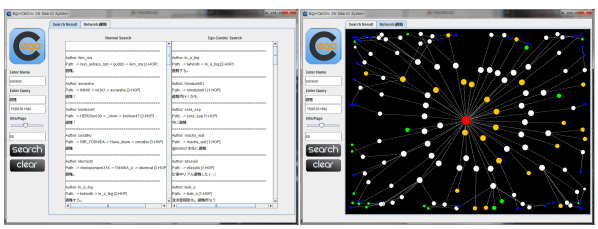

図 8　Ego-Centric Social Network Search System の画面
Fig. 8　UI of Ego-Centric Social Network Search System.

このシステムを用いて，本研究で用いたパーソナライズドソーシャルサーチのランキング手法の妥当性に関して，パーソナルな情報とローカルな情報という 2 つの観点で検索例を示す．検索結果に現れる x-HOP（$x = 1, 2, \cdots$）という表記は，検索者から見て作成者が距離 x 離れていることを表している．

図 9 はクエリは “避難” で検索した場合の Normal Search（左）と Ego Centric Search（右）の検索結果の違いを表している．Normal Search による検索結果はユーザ関係スコアを考慮していないため，上位に現れてくる文書の作成者と検索者との距離も様々である．以下は，Ego Centric Search による検索結果の上位に出現する文書の例である．

“医学部体育館は避難場所らしいです．”

Ego Centric Search による検索結果では，距離の近い作成者の避難している場所の情報等のパーソナルな情報を取得できている．

また，異なる検索者による検索結果の比較を行うことで，検索者に即した検索結果が取得できていることを確認する．居住地がそれぞれ宮城，福島，茨城，千葉と異なる検索者 A，B，C，D を選出する．**図 10** は，Normal Search による検索結果の文書ランキングを示している．クエリは “給水” である．Normal Search に現れる文書は，いずれも距離の遠い作成者の文書であり，情報量も少なく場所についても多様である．**図 11** は検索者 A，B，C，D それぞれを検索者とした場合の，文書ランキングである．各検索者

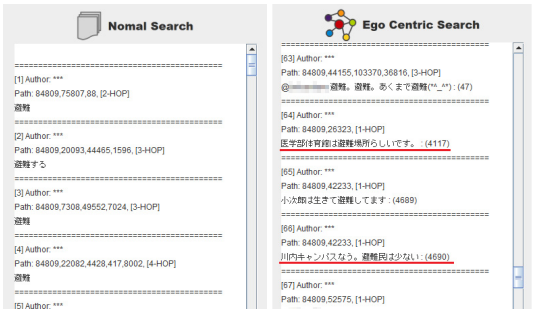

図 9　クエリ “避難” の検索結果
Fig. 9　Retrieval result (“Hinan”).

図 10　Normal Search による検索結果の文書ランキング（クエリ“給水”）

Fig. 10　Retrieval result of normal search (“Kyu-sui”).

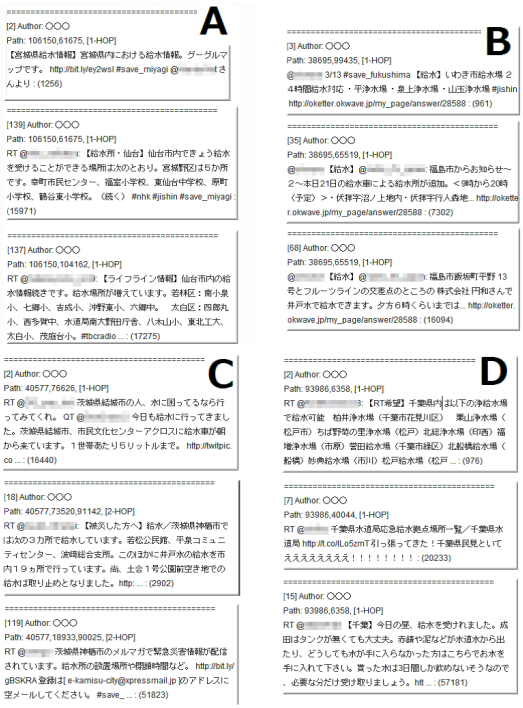

図 11　検索者 A，B，C，D の検索結果：文書ランキング（クエリ“給水”）

Fig. 11　Ranking of documents of searcher A, B, C and D (“Kyu-sui”).

の居住地に関係する情報が文書ランキングに出現していることが分かる．給水に関する情報は検索者の居住地に強く関連があると考えることができる．以下は検索者 A の検索結果に現れる文書の例である．検索者 A の居住地である宮城の給水所に関する情報が検索結果の上位にランキングされている．

> “【宮城県給水情報】宮城県内における給水情報。グーグルマップです。http://bit.ly/ey2wsl\
> #save_miyagi @○○○ さんより”

図 12 は，検索者 A，B，C，D それぞれを検索者とした場合の，作成者ネットワークである．Normal Search と Ego-Centric Search の両方で上位に現れる作成者を示すノード（緑）が少なく，スコア全体に占める Similarity，Familiarity の影響がより強く現れていることが分かる．居住地に強い関連があるローカルな情報を取得できている．

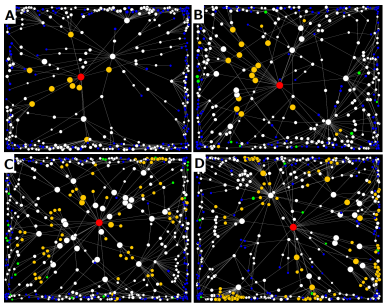

図 12　検索者 A，B，C，D の検索結果：作成者ネットワーク（クエリ“給水”）

Fig. 12　Author’s network of searcher A, B, C and D (“Kyu-sui”).

位置に関する検索結果を得るためには，位置情報付きのツイート等を用いることも考えられるが，検索対象のツイートが極端に少なくなるという問題がある．一方，提案手法では，検索者と作成者のツイートに含まれる単語を用いることは比較的容易であり，Similarity としてスコアに加えることで，位置情報に限らず，自身が言及していて関心があると考えられる情報を取得できる．

6.　おわりに

本論文では，top-k パーソナライズソーサシャルサーチのための効率的なアルゴリズムである，Single Index アルゴリズム，Social Index Graph アルゴリズム，ハイブリッドアルゴリズム，の 3 つの検索アルゴリズムを提案した．性能評価によって，Single Index アルゴリズムはクエリにヒットする文書数が少ない場合，Social Index Graph アルゴリズムはヒットする文書数が多い場合に検索結果確定までの応答時間が短いという性能特徴を示した．ハイブリッドアルゴリズムに関しては，回帰分析による切替え基準ヒット数の推定を行い，その妥当性を交差検証法により確認した．性能評価により，ハイブリッドアルゴリズムを用いることでヒット数の多寡によらず，高速なパーソナライズドソーシャルサーチが可能であることを示した．

本論文では提案するアルゴリズムの性能評価のために，3 つの基本的な指標として，クエリと文書の適合度である Relevancy，ユーザプロファイルの類似度である Similarity，ユーザの親密度である Famiriarity を用いた．ソーシャルサーチに関しては，これらのほかに，4.1 節で述べた以外にも各ユーザと検索クエリとの関係をスコアに用いるような指標等，多くの指標が提案されている．各指標の特徴に合わせた様々な高速化手法が考えられるが，ユーザ間の距

離をスコアに用いる場合には本研究の提案するアルゴリズムの適用が可能であると考える．

謝辞　実験データの収集にご協力いただいた兼山元太氏（クックパッド株式会社）に感謝する．

参考文献

[1] Bender, M., Crecelius, T., Kacimi, M., Michel, S., Neumann, T., Parreira, J.X., Schenkel, R. and Weikum, G.: Exploiting social relations for query expansion and result ranking, *Proc. ICDE Workshops*, pp.501–506 (2008).

[2] Carmel, D., Zwerdling, N., Guy, I., Ofek-Koifman, S., Har'el, N., Ronen, I., Uziel, E., Yogev, S. and Chernov, S.: Personalized social search based on the user's social network, *Proc. CIKM*, pp.1227–1236 (2009).

[3] Turtle, H. and Flood, J.: Query evaluation Strategies and optimizations, *Information Processing Management*, Vol.31, No.6, pp.831–850 (1995).

[4] Brown, E.W.: Fast Evaluation of Structured Queries for Information Retrieval, *Proc. SIGIR*, pp.30–38 (2009).

[5] Jones, K.S.: A statistical interpretation of term specificity and its application in retrieval, *Journal of Documentation*, Vol.28, No.1, pp.11–21 (1972).

[6] Jansen, B.J. and Spink, A.: An Analysis of Web Documents Retrieved and Viewed, *Proc. ICOMP*, pp.65–69 (2003).

[7] Ronald, F., Lotem, A. and Naor, M.: Optimal aggregation algorithms for middleware, *Proc. PODS*, pp.102–113 (2001).

[8] Hotho, A., Jaschke, R., Schmitz, C. and Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking, *Proc. ESWC*, pp.411–426 (2006).

[9] Yanbe, Y., Jatowt, A., Nakamura, S. and Tanaka, K.: Can social bookmarking enhance search in the web?, *Proc. JCDL*, pp.107–116 (2007).

[10] Bao, S., Xue, G., Wu, X., Yu, Y., Fei, B. and Su, Z.: Optimizing web search using social annotations, *Proc. WWW*, pp.501–510 (2007).

[11] Zobel, J. and Moffat, A.: Inverted files for text search engines, *ACM Computing Surveys*, Vol.38, No.2 (2006).

[12] Anh, V.N., de Kretser, O. and Moffat, A.: Vector-space ranking with effective early termination, *Proc. SIGIR*, pp.35–42 (2001).

[13] Anh, V.N. and Moffat, A.: Impact transformation: Effective and efficient web retrieval, *Proc. SIGIR*, pp.3–10 (2002).

[14] Chen, C., Li, F., Ooi, B.C. and Wu, S.: TI: An Efficient Indexing Mechanism for Real-Time Search on Tweets, *Proc. SIGMOD*, pp.649–660 (2011).

[15] Horowitz, D. and Kamvar, S.D.: The anatomy of a large-scale social search engine, *Proc. WWW*, pp.431–440 (2010).

[16] Brandes, U.: A Faster Algorithm for Betweenness Centrality, *Journal of Mathematical Sociology*, Vol.25, pp.163–177 (2001).

三浦 大樹

2013 年電気通信大学大学院情報システム学研究科社会知能情報学専攻博士前期課程修了．同年，日本電信電話（株）入社．現在 NTT ソフトウェアイノベーションセンタ所属．

諏訪 博彦　（正会員）

1998 年群馬大学社会情報学部卒業．2006 年電気通信大学大学院情報システム学研究科博士後期課程修了．博士（学術）．現在，電気通信大学大学院情報システム学研究科社会知能情報学専攻助教．

鳥海 不二夫

2004 年東京工業大学大学院理工学研究科機械制御システム工学専攻博士課程修了．同年名古屋大学情報科学研究科助手，2007 年同助教，2012 年東京大学大学院工学系研究科准教授，現在に至る．エージェントベースシミュレーション，人工市場，ソーシャルメディア等の研究に従事．電子情報通信学会，人工知能学会，日本社会情報学会各会員．博士（工学）．

鬼塚 真　（正会員）

1991 年東京工業大学工学部情報工学科卒業．同年，日本電信電話（株）入社．2000～2001 年ワシントン州立大学客員研究員．現在，日本電信電話（株）ソフトウェアイノベーションセンタ特別研究員，電気通信大学客員教授，博士（工学）．大規模グラフデータのデータ処理に関する研究開発に取り組んでいる．ACM，電子情報通信学会，日本データベース学会各会員．

（担当編集委員　豊田 正史）