パーソナライズドソーシャルサーチのための効率的な top-k 検索アルゴリズムの提案

三浦　大樹† 諏訪　博彦† 鳥海不二夫†† 鬼塚　真†††

† 電気通信大学　〒 182–8585 東京都調布市調布ケ丘 1-5-1
†† 東京大学　〒 113-8656 東京都文京区本郷 7-3-1
††† NTT ソフトウェアイノベーションセンタ　〒 180–8585 東京都武蔵野市緑町 3-9-11
E-mail: †miura@ohta.is.uec.ac.jp, ††h-suwa@is.uec.ac.jp, †††tori@sys.t.u-tokyo.ac.jp,
††††onizuka.makoto@lab.ntt.co.jp

あらまし　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伴い，検索結果が確定するまでの応答時間が遅くなる問題がある．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．一つ目は，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．二つ目は，ユーザ毎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．三つ目は，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．

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 top-k Personalized Social Search

Hiroki MIURA†, Hirohiko SUWA†, Fujio TORIUMI††, and Makoto ONIZUKA†††

† University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 182–8585, Japan
†† University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan
††† NTT Software Innovation Center, 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
E-mail: †miura@ohta.is.uec.ac.jp, ††h-suwa@is.uec.ac.jp, †††tori@sys.t.u-tokyo.ac.jp,
††††onizuka.makoto@lab.ntt.co.jp

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 algorithms for the social search; Single Index algorithm, Social Index Graph algorithm, Hybrid algorithm. We use Twitter data to compare the efficiency of the two algorithms and show those features of performance. From the result we show validity of the change rule on the hybrid algorithm.

Key words　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上の文書を対象とする検索では，検索結果のランキングの際にソーシャルアノテーション(注1)の情報を利用して文書の重要度を決定することでユーザの属性や嗜好に合致するような検索結果を取得出来ることが示されている [1], [7], [16]．このような検索はソーシャルサーチと呼ばれ，その定義は様々である．Carmel ら [4] はソーシャルサーチを，ソーシャルデータを用いた検索プロセスであると定義している．ソーシャルデータとは，ソー

(注1)：コンテンツに対するコメント，お気に入り，ソーシャルブックマーク，Facebook の いいね! 等

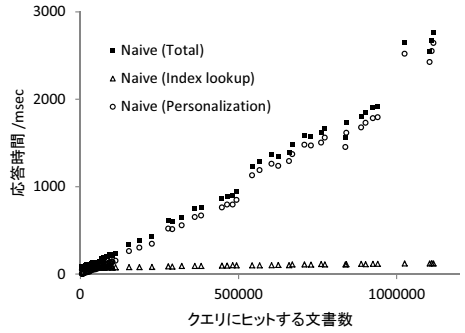

図 1　高速化を適用しない場合の応答時間

シャルブックマーク，wiki，ブログ，フォーラム，SNS から取得できる情報を指している．

また，ソーシャルメディアにおいてユーザはフレンド関係やフォロー/フォロワー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繋がりに基いて情報を取得している [10], [17], [19]．よって，検索者と検索対象の文書の作成者とのユーザ関係を検索結果のランキングに利用することで，従来の検索では対応出来ない，各検索者の嗜好に合った情報や繋がりのある人の情報を重視するような検索が可能である．

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* [2] 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* [4] 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．

図 1 は通常の検索に用いられる転置ファイルを利用した場合のパーソナライズドソーシャルサーチの応答時間を示している．横軸にクエリにヒットする文書数，縦軸にヒット数が変化した場合の応答時間を記載している．第一の Index lookup ステップではクエリにヒットする文書の取得とクエリと文書の適合度の計算を行い，第二の Personalization ステップでは，パーソナライズの操作である，文書の作成者の特定とユーザ関係スコアの計算，スコアの合成計算を行っている．Total は 2 つのステップの合計時間である．応答時間の大半が Personalization ステップに要する時間で絞められており，その時間は文書のヒット数に従って線形に増加する傾向が見られる．

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 3 つの効率的な検索アルゴリズムを提案する．一つ目は，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．二つ目は，ユーザ毎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．三つ目は、Single Index アルゴリズムと Social Index Graph アルゴリズムをヒット数の多少を基準に切り替えるハイブリッドアルゴリズムである．

本論文の構成は以下の通りである．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 章で関連研究を述べ，5 章では本論文の結論と今後の展望を述べる．

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

検索結果のランキングを決定するための文書のスコアは，計 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．検索者事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 $v(d)$ とした場合，以下の式 (1) で計算される．

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

式 (1) のスコアは，クエリと文書との適合度にユーザ関係スコアを合成することで，検索者に即した検索結果のパーソナライズを行っている．α は，$R(q,d)$ と ユーザ関係スコア，β は $S(u,v(d))$ と $F(u,v(d))$ の線形和を取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 [8] 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．

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

- Similarity の導入の目的は，ユーザ同士の属性は似ているが直接繋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 $v(d)$ とのユーザ属性の類似度に基づいて計算される．利用するユーザ属性は性別や年代，興味・関心・話題等，様々なものが考えられる．検索者 u と文書の作成者 $v(d)$ のユーザ属性集合を U,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．

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

- 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/フォロワー関係や，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

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**

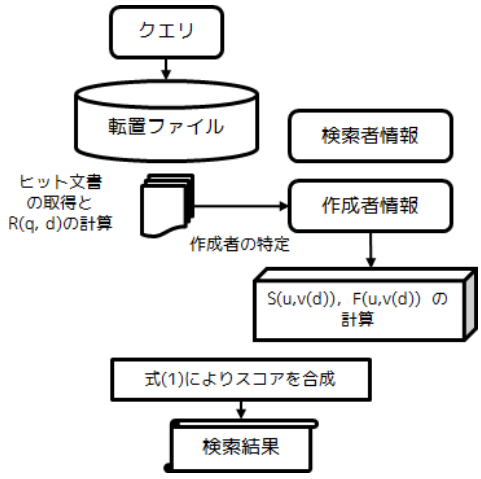

図 2　Single Index アルゴリズム検索処理イメージ

全て 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

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．全ユーザ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．

図 2 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の部分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，<doc#,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，作成者を特定するためにはこの $doc\#$ を基にディスク上のインデックスにアクセスし，作成者情報を取得しなければならない．そこで，転置ファイルに新たに 文書の作成者の情報である authorid を埋め込み <doc#,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 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: $docs \leftarrow$ empty priority queue
2: $Nodes \leftarrow$ empty queue
3: $Visited \leftarrow$ empty list {visited node list}
4: $u \rightarrow Nodes, Visited$
5: **while** $Nodes$ not empty **do**
6: $w \leftarrow Node$
7: **for** $neighbor\ v$ of w **do**
8: **if** $v \notin Visited$ **then**
9: $v \rightarrow Visited, Node$
10: $D(v,q) \leftarrow getDocDivIdx(v,q)$
11: **for all** $d \in D$ **do** {in descending order of $R(q,d)$}
12: $Score(u,q,d) \leftarrow \alpha R(q,d) + (1-\alpha)\{\beta S(u,v) + (1-\beta)F(u,v)\}$
13: $\overline{Score}(u,q,d) \leftarrow \alpha \overline{R}(q,d) + (1-\alpha)\{\beta \overline{S}(u,v) + (1-\beta)F(u,v)\}$
14: **if** $docs.size < k$ **or** $docs.kthscore < Score(u,q,d)$ **then**
15: $docs \leftarrow (d, Score(u,q,d))$
16: **else if** $docs.kthscore > \overline{Score}(u,q,d)$ **then**
17: $return$
18: **end if**
19: **end for**
20: **end if**
21: **end for**
22: **end while**

Algorithm 1 は検索アルゴリズムの詳細な手順を表している．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 毎に作成者を特定する (行 4)．特定した作成者とのユーザ関係スコア $S(u, v(d))$，$F(u, v(d))$ 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，$Score(u,q,d)$ の上限値を計算する (行 6)．ここで，$\overline{Score}(u,q,d)$ を用いて Threshold Algorithm [13] 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 $R(q,d)$ を一度に計算出来ることである．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事ができる．しかし，top-k 文書の作成者の $S(u, v(d))$，$F(u, v(d))$ の値が大きい場合など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ユーザの計算を省略する事が出来ない．

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．全文書を作成者毎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

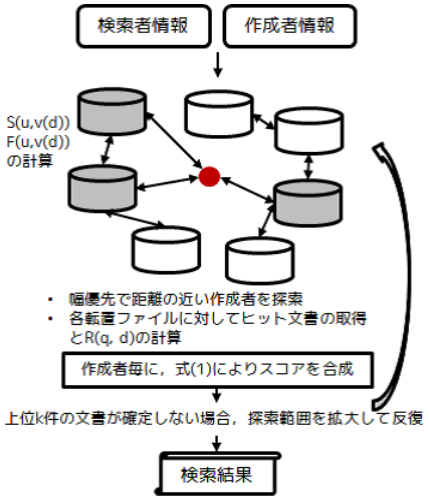

図 3　Social Index Graph アルゴリズム検索処理イメージ

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間の距離の計算をし，訪問した近隣ユーザ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．Familiarity の値が等しいユーザ内では Similarity について降順にソートされ，その順に処理が行われる．

図 3 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 ホップのユーザの転置ファイルを探索する．2) 探索したユーザ 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，更に探索範囲を広げて反復する．

Algorithm 2 は Social Index Graph による検索処理の詳細を表している．訪問予定ノードと訪問済みノードを格納する $Nodes$，$Visited$ を用意し，ソーシャルグラフを幅優先で探索する (行 1-7)．ノード v が未訪問ならば $Visited$ に格納し，ユーザ 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全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ユーザ間の距離に基づくスコアである $F(u, v(d))$ を検索時に高速に計算出来る事である．検索者を中心に幅優先でグラフを探索すると同時に $F(u, v(d))$ の計算を行う．また，予めユーザ毎に文書が分割されているので文書毎の作成者の特定を必要としない．そのため $R(q, d)$ を計算する際には全文書に対する IDF の値を使用する必要があるが，転置ファイルへのユーザ情報の付加といった拡張は不要である．Threshold Algorithm は $S(u, v(d))$，$F(u, v(d))$ を基準に行われ，検索者の近隣ユーザから順に効率的な top-k 文書の検索が可能である．

表 1　各アルゴリズムの計算量

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

2.2.3　ハイブリッドアルゴリズム

このアルゴリズムは，2.2.1 で説明した Single Index アルゴリズムと 2.2.2 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事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連れ，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有ることに起因する．Social Index Graph は，ヒットする文書が多い程，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連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取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り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することが可能になる．

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ユーザ数を 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毎にサイズが 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)．ヒットした文書毎にスコ

表 2　Twitter データの詳細

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

表 3　各アルゴリズムのインデックスのサイズ

通常の転置ファイル	Single Index	Social Index Graph
14.8 GB	17.4 GB	20.0 GB

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 ユーザ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 によりユーザ m 人分となる．h が大きい時，$m << n$ となり，全体の計算量が小さくなる．

3. 評価実験

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り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*) を基に有向のソーシャルネットワークを構築する．ユーザ A が @ユーザ B の含まれる投稿した場合，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．

インデックスとは，適合文書の取得と Relevancy を計算するための転置ファイル，Similarity を計算するためのユーザプロファイル，Familiarity を計算するためのソーシャルネットワーク，のサイズの合計を表している．ユーザプロファイルは 150 MB，ソーシャルネットワークは 15.8 MB である．また，各アルゴリズムで用いる転置ファイルのサイズを表 3 で示す．通常の転置ファイルに比べて，Single Index アルゴリズムの転置ファイルのサイズが大きいのは，転置ファイルにユーザ情報を埋め込む拡張をしているためである．また，Social Index Graph アルゴリズムの場合，ユーザ情報の埋込みは行なっていないが，ユーザ毎に転置ファイルを分割したことによるサイズの増加が見られる．

3.2 実験条件と実験環境

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 100 ユーザを選択し，検索クエリは全文書中から 5000〜100 万ヒットするような単語 100 語を選択する．例として，表 4 にヒット数上位 20 件の単語を示す．スコア式 (1) における重みは $\alpha, \beta = 0.5$ とする．本実験では，ユーザ属性集合を，ユーザの直近 200 件の投稿中に含まれる単語の内，出現頻度が多い上位 100 単語と定める．このユーザ属性集合を基に Similarity を計算する．また，実験に用いたハードウエア及びソフトウエアの環境は表 5 の通りである．

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)

順位	単語						
1	時間	6	拡散	11	帰る	16	心配
2	日本	7	食べる	12	仕事	17	問題
3	被災	8	話	13	ツイート	18	携帯
4	大丈夫	9	福島	14	電話	19	最近
5	明日	10	避難	15	確認	20	報道

表 5　性能測定環境

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

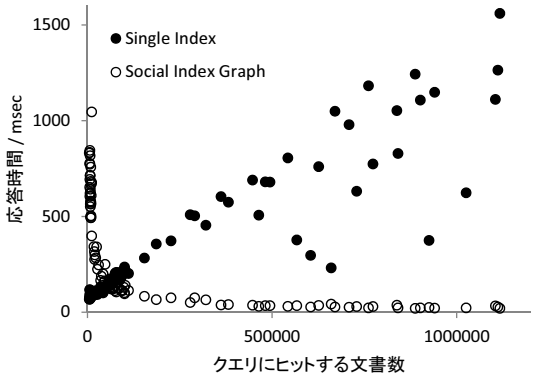

図 4　2 つのアルゴリズムの応答時間とクエリにヒットする文書数

3.3 結　　果

図 4 は，$k = 100$ の場合の Single Index アルゴリズムと Social Index Graph 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全て取得し，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伴って，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，近隣のユーザ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，近隣ユーザ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．

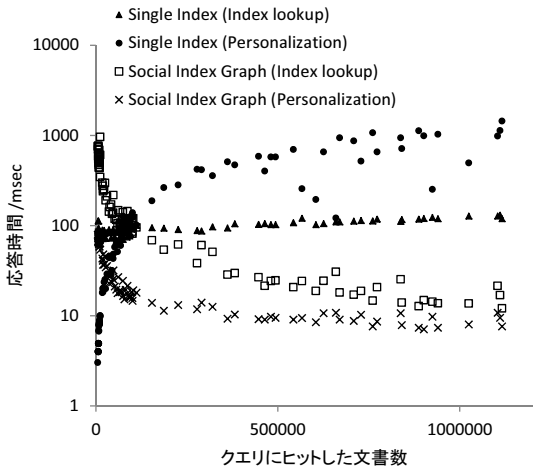

図 5　各アルゴリズムが Index lookup と Personalization に要する応答時間

図 5 は全体の応答時間を，クエリにヒットする文書を取得と $R(q,d)$ の計算を行う *Index lookup* と，$S(u,v(d))$ と $F(u,v(d))$ 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．

図 6 は k=10，図 7 は k=100，図 8 は k=1000 と，k の値を変化させた場合のヒット文書数と応答時間の関係を表している．一般に k の値が大きくなると，それだけスコア計算をする文書数が多くなるため応答時間は増加する．

3.4　ハイブリッドアルゴリズムにおける切り替え基準ヒット数の検証

図 4 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事が確認できた．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り替え基準ヒット数を h_s と置き，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 0.863 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，回帰分析による切り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言える．

表 6　各グループで推定された h_s と的中率 r の値

	1	2	3	4	5	
h_s	60134	61251	59712	59334	58112	
r	0.867	0.866	0.863	0.863	0.863	
	6	7	8	9	10	平均値
	57223	62561	57331	58311	59912	59388.1
	0.860	0.867	0.860	0.862	0.862	0.863

4.　関 連 研 究

第 4. 章では本研究に関連する研究について述べる．パーソナライズドソーシャルサーチに関連する研究は，文書のスコアを決定する新たな指標を提案し検索精度を評価する研究と，応答時間の短縮や転置ファイルの容量の削減などの検索性能を評価する研究に分類する事ができる．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　パーソナライズドソーシャルサーチに用いられるスコア

Bender ら [2] は，ユーザ間の距離に基いて計算される *Friendship*，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*，タグの共起頻度から計算されるタグの類似度である *TagSim* を組み合わせることで，ユーザ，コンテンツ，タグの関係を用いたスコア指標を提案している．また，Carmel ら [4] は 2 つの異なるソーシャルネットワークを用いて文書のスコアを計算する手法を提案している．一つ目は，ユーザ属性の類似度を表す *Similarity-based network* であり，二つ目はユーザ間の親密度を表す *Familiarity-based network* である．2 つのネットワークから計算されるスコアによって検索結果のパーソナライズを行い，そのスコアの有効性を検証している．Kashyap ら [9] は，Social Aware Search graph model と呼ばれるネットワーク利用した指標である *SonetRank* を提案している．そのネットワークは以下の 3 つの観点から構築されている．1 つ目は，過去の検索において同時に検索した単語の組み合わせを基にした Individual Preferences，2 つ目は，似ている，または同じグループに属している他のユーザの検索結果を基にした Group Preferences，3 つ目は，他のグループのユーザによって関連があると判断された検索結果に基づく Query Preferences，である．

本論文で扱うパーソナライズドソーシャルサーチは，従来取り組まれていなかった，ユーザ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，複数のスコアの合成，転置ファイルを用いた適合文書

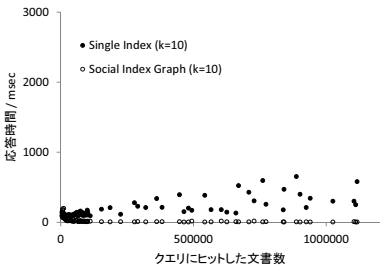

図 6　ヒット文書数と応答時間の関係 (k=10)

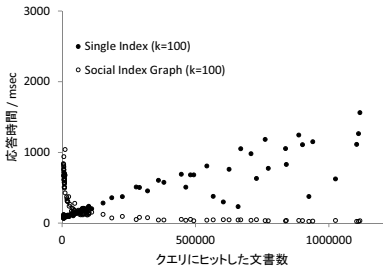

図 7　ヒット文書数と応答時間の関係 (k=100)

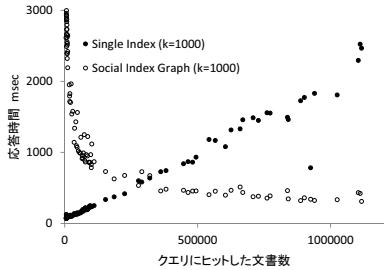

図 8　ヒット文書数と応答時間の関係 (k=1000)

の取得，ユーザ関係スコアの計算と取得，の 3 つに分類することが出来る．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 [14] は，複数の単語による全文検索を行う際に考えられる 2 つの戦略を比較している．各文書毎に単語の包含判定を行う *document-at-a-time* と，文書集合に対して最初の単語による絞込みを行い，結果の部分集合に対して順次，残りの単語を適用していく *term-at-a-time* の 2 つのアプローチの比較を行っている．その結果，*term-at-a-time* アプローチを用いる際に，各単語に適合する候補集合内のスコア上位の文書に対して，残りの単語による評価を行う *max-score optimization* が最もコストを削減できる事を示している．Brown [3] は，文書スコアへの寄与率が高い文書の情報を転置ファイルのヘッダファイルに保持し，検索結果の候補集合を絞込む処理を初めに実行することで，検索の応答時間が短縮出来る事を示している．

スコアの計算順序を考慮することはソーシャルサーチにおいても有効であるが，ユーザ関係スコアは，クエリや文書の内容に依存しない事に加え，値が検索者毎に異なるため，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事は，計算量や容量の観点から現実的ではない．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，本研究では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依らずに高速なソーシャルサーチを実現している．

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 [18] 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など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転置ファイルを作成する場合の格納順が重要であると述べている．一般的な転置ファイルは単純に文書 ID の順に構築される．Persin [12]，Wong ら [15] は文書頻度順に転置ファイルを構築する手法を提案している．Chen ら [5] 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．また，Garcia ら [6] はクエリ履歴を用いて，アクセス頻度順に自動的に転置ファイルを構築し直す手法を提案しており，約 25-40%のクエリ評価の時間を短縮できる事を示している．Lee ら [11] は，ソーシャルタグを利用した情報検索を行う際にの検索処理を効率的に行える Social inverted index を提案している．転置ファイルを構築する際に，ページに対してタグ付けを行ったユーザの情報をリストとして合わせて格納することで，タグが付けられたページのスコアを計算する際にそのリストを辿ることでユーザに関するスコア取得できる．

ユーザ関係スコアは，適合度や寄与率，新規性等のすべてのユーザに等しい指標ではなく，検索者によって異なる値であるため，一元的な文書の優先順位によって転置ファイルを構築する技術は利用できない．本研究の Social Index Graph アルゴリズムでは，ソーシャルグラフに従って接続された転置ファイルを用いており，繋がりのあるユーザの探索と同時に，そのユーザの文書から優先的に検索を行う手法を提案している．

5. 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 3 つの検索アルゴリズムを提案した．提案したアルゴリズムは以下の 3 つである．1 つ目は，Single Index アルゴリズムであり，全ユーザの全文書から構築した 1 つの転置ファイルを利用する．転置ファイルによって適合文書の取得と適合度の計算を行い，適合度について降順に並べられた文書に対して順にユーザ関係スコアを合成する．転置ファイルに対してユーザ情報を埋め込む拡張を行うことで作成者の高速な特定を実現している．

2 つ目は，Social Index Graph アルゴリズムであり，ユーザ毎に分割された文書から構築された転置ファイルを利用する．これらの転置ファイルはソーシャルネットワークのユーザ関係に基いて接続されている．検索者の近隣ユーザを幅優先探索で発見し，ユーザ間の距離が近い順にそのユーザの文書に対して適合文書の取得を行う．この動作を検索結果が確定するまで繰り返し行う．ヒット数が多い場合は，近隣ユーザに検索範囲が限定される可能性が高いため，高速な検索が可能である．

3 つ目は，ハイブリッドアルゴリズムであり，ヒット数の多少によって Single Index アルゴリズムと Social Index Graph アルゴリズムを切り替えて利用する．切り替えの基準となるヒット数は，検索対象のデータ内から，検索者とクエリをサンプリングし，各アルゴリズムの性能を示す回帰直線の交点から算出する．

性能評価では Twitter のデータを用いて各アルゴリズムの性

能特性を確認した．Single Index アルゴリズムはヒット数が小さい場合に高速な検索が可能であった．これはヒット数が少ないほどスコア計算の対象になる文書数が少なくなるためである．対して Social Index Graph アルゴリズムはヒット数が多いほど高速な検索が可能であった．ヒット数が多いほど，そのクエリに適合する文書を持つユーザが多く，検索者の近隣ユーザを検索するだけで検索結果が確定する可能性が高くなるである．また，交差検証法を用いて，切り替え基準ヒット数の妥当性を確かめた．その結果，約 86 % の的中率でより高速な方のアルゴリズムの選択が行える事を示した．よって，本研究の提案するハイブリッドアルゴリズムを用いることで，ヒット数の多少にかかわらず高速な top-k パーソナライズドソーシャルサーチが高い精度で実現できることを示した．

提案した Social Index Graph アルゴリズムは，ヒット数が多い場合に高速な検索が可能なアルゴリズムであるが，反面ヒット数が少ない場合には応答時間が指数関数的に遅くなる．これは検索対象のユーザが増加し，各ユーザの文書を検索する際に発生する転置ファイルの切り替えのオーバーヘッドが原因である．そのため，ユーザ毎ではなく，ユーザクラスタごとの文書を格納する転置ファイルを用いるようなアルゴリズムがあれば，ヒット数に依存しないような安定した性能でパーソナライズドソーシャルサーチを実行する事が可能なのではないかと考える．ユーザの分割には，トピックやネットワーク構造に基づくクラスタリング等が考えられる．

謝 辞

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

文　　献

[1] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *Proceeding of WWW*, pp. 501–510, 2007.

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

[3] E. W. Brown. Fast evaluation of structured queries for information retrieval. In *Proceeding of SIGIR*, pp. 30–38, 2009.

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

[5] C. Chen, F. Li, B. C. Ooi, and S. Wu. Ti: An efficient indexing mechanism for real-time search on tweets. In *Proceeding of SIGMOD*, pp. 649–660, 2011.

[6] S. Garcia, H. E. Williams, and A. Cannane. Access-ordered indexes. In *Proceedings of ACSC*, pp. 7–14, 2004.

[7] A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies:search and ranking. In *Proceeding of ESWC*, pp. 411–426, 2006.

[8] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[9] A. Kashyap, R. Amini, and V. Hristidis. Sonetrank: leveraging social networks to personalize search. In *Proceeding of CIKM*, pp. 2045–2049, 2012.

[10] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proceedings of SIGKDD*, pp. 611–617, 2006.

[11] K.-P. Lee, H.-G. Kim, and H.-J. Kim. A social inverted index for social-tagging-based information retrieval. *Journal of Information Science*, 38(4):313–332, 2012.

[12] M. Persin, J. Zobel, and R. Sacks-Davis. Filtered document retrieval with frequency-sorted indexes. *Journal of the American Society for Information Science*, 47(10):749–764, 1996.

[13] F. Ronald, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of PODS*, pp. 102–113, 2001.

[14] H. Turtle and J. Flood. Query evaluation strategies and optimizations. *Information Processing Management*, 31(6):831–850, 1995.

[15] W. Y. P. Wong and D. L. Lee. Implementations of partial document ranking using inverted files. *Inf. Process. Manage.*, 29(5):647–669, 1993.

[16] Y. Yanbe, A. Jatowt, S. Nakamura, and K. Tanaka. Can social bookmarking enhance search in the web? In *Proceeding of JCDL*, pp. 107–116, 2007.

[17] J. Yang and S. Counts. Comparing information diffusion structure in weblogs and microblogs. In *Proceedings of AAAI*, pp. 351–354, 2010.

[18] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.

[19] 風間一洋. Twitter における情報伝播. 人工知能学会誌, 27(1):35–42, 2012.