ソーシャルネットワークを利用したパーソナライズ情報検索

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
†† 名古屋大学　〒 464–8601 愛知県名古屋市千種区不老町
††† NTT サイバースペース研究所　〒 239–0847 神奈川県横須賀市光の丘 1-1
E-mail: †miura@ohta.is.uec.ac.jp, †h-suwa@is.uec.ac.jp, ††tori@is.nagoya-u.ac.jp,
†††onizuka.makoto@lab.ntt.co.jp

あらまし 近年増加の傾向を見せているソーシャルメディアの検索では，人の繋がりのような検索者個人に関わる指標を的確に組み合わせることで，検索者に即した検索精度の向上が実現できると考えられる．その中でソーシャルネットワーク上のユーザ関係をドキュメントの検索対象のスコアリングに取り入れる提案やその有効性を示す研究もなされている．本論文では，ソーシャルネットワーク上のユーザー関係をドキュメントのスコアリングに取り入れる事を前提とした上で実際の検索処理を行う際の性能問題を明らかにし，その問題に対する高速化手法を提案する．具体的には，1) 人間関係に基づくスコアの事前計算，2) ドキュメントの作成者情報を保持するよう転置ファイルを拡張，3)Threshold Algorithm に基づく Top-k 検索，の 3 つの手法を提案し性能評価を行ない提案手法の有用性を示した．

キーワード 情報検索，ソーシャルネットワーク，ソーシャルサーチ

Personalized Social Search Using Social Network

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

† University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 182–8585, Japan
†† Nagoya University, Furocho, Chikusa-ku, Nagoya, Aichi, 464–8601, Japan
††† NTT Cyber Space Laboratories, Hikarinooka 1-1, Yokosuka-shi, Kanagawa, 239-0847 Japan
E-mail: †miura@ohta.is.uec.ac.jp, †h-suwa@is.uec.ac.jp, ††tori@is.nagoya-u.ac.jp,
†††onizuka.makoto@lab.ntt.co.jp

1. はじめに

1.1 研究背景と研究概要

従来の Web ページの検索におけるページのスコアリングには，クエリの単語に対するドキュメントの適合度に基づく TFiDF [1] や改良型の Okapi BM25 [2] がよく知られており，ページ間のリンク構造に基づいた PageRank [3] のような指標が提案され幅広く用いられてきた．

しかし，近年爆発的に増加の傾向が見られるソーシャルメディアを対象とする検索では，これらの指標に代表されるようなドキュメントの特性を基準とした検索よりはむしろ，検索者やドキュメントの作成者といった，ユーザの特性に比重を置くような検索要求が見られるようになった．さらに，Web サービスの発達によって性別や年代，好みといったユーザ属性や，Web ページへのアクセスログ，EC サイトでの購買履歴等のユーザ行動の情報はもとより，以前には客観的には検出が困難であった，人の繋がりなどの情報が目に見える形で容易に取得，活用できるようになった．そのような状況も相まって，人の繋がりを情報検索に取り入れる研究も登場してきた．

このような状況の中で一般的な Web の検索エンジンにもパーソナライズ機能を組み込む動きが見られる．Google は *Google Social Search* [4] として，検索者のプロファイルを分析し近いユーザのコンテンツを検索結果として表示する機能や Twitter などの外部サービスから友人がコメントしたコンテンツを表示する機能を Web ページの検索に取り入れた．また，それら踏襲する形で Google+で繋がりのあるユーザのコンテンツを検索結果として表示させることでより検索者にパーソナライズした検索を可能にする *Search plus Your World* [5] を検索エンジンの機能として公開した．

また，ソーシャルネットワークから得られる指標を利用してパーソナライズ検索に役立てる研究も行われている．ソーシャルサーチエンジン Aardvark [6] はクエリに回答出来る可能性

がある人を探すサーチエンジンである．ユーザープロファイルやユーザが発信した情報から，*Social connection* としてソーシャルネットワークを形成しそれをスコアリング計算に用いている．Bender ら [7] はドキュメントのスコアリングの要素として，ソーシャルネットワークに PageRank の考え方を適用した *UserRank* と，ソーシャルネットワーク上のユーザ間の距離を基に計算される *Friendship* という指標を用いる事を提案している．Camel ら [8] は，ユーザ属性の類似度を表す *Similarity-based network* とユーザの親密性を表す *Familiarity-based network* という 2 つの異なるソーシャルネットワークから計算される指標を提案し，異なるユーザ関係によるパーソナライズ指標の解析を行なっている．両研究とも，ソーシャルネットワーク上のユーザ関係をドキュメントなどの検索対象のスコアリングに取り入れることの有効性を示している．

両研究とも検索対象のスコアを決定する項として，ユーザ間の人間関係および検索対象とユーザの間の関係を利用しているが，前者はユーザ数に依存し後者は検索対象数に依存して検索処理の計算量が増大するという問題がある．Bender ら [7], Camel ら [8] はスコア精度の評価しか行っておらず，このような現実的な計算速度の課題については言及をしていない．よって，この部分の処理コストをできるだけ軽減するような対応策が必要である．詳しくは 1.2 節で説明する．

1.2 検索例と検索システム

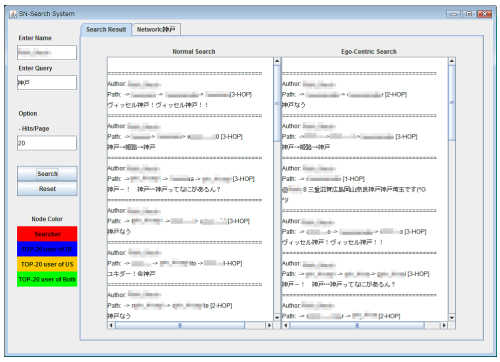

図 1　画面：検索結果ドキュメント表示タブ

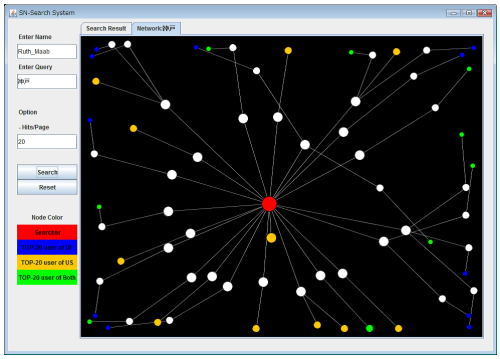

図 2　画面：ネットワーク表示タブ

本論文で対象とするソーシャルサーチシステムについて説明する．検索者は自分の ID と検索したいクエリを入力する．検索結果は図 1 のように，(i) クエリに対する適合度に基づくスコアリングによる結果，(ii) ソーシャルネットワーク上のユーザー関係をスコアリングを含めた結果，の両方が表示される．ドキュメント結果にはドキュメントの作成者 (投稿者) の ID と検索者から作成者へのパスそして本文が表示される．また図 2 のようにネットワーク表示タブには，上位 k 件までに含まれるドキュメントの作成者と検索者が含まれるようなサブネットワークが形成され表示される．ノードの色はユーザの種別を表しており，赤は検索者，青は (i) の検索結果に表れるユーザ，橙は (ii) の検索結果に表れるユーザ，緑は (i) と (ii) 両方の検索結果に表れるユーザを表している．その他は，検索結果には表示されないが検索者とドキュメントの作成者を繋ぐ役割を果たしているユーザである．ノードの大きさは検索者とのソーシャルネットワーク上の距離の大小を表している．

ユーザ関係を検索に取り入れるにあたって必要な処理を具体的に見ていくと，まず新たな指標であるユーザ関係に基づくスコアの計算をしなければならない．次にクエリに合致したドキュメントを抽出した後に，そのドキュメントの作成者を特定する必要がある．そして最後に，新たなスコアを合成して並べ替える操作を経て検索結果の提示となる．これらを単純に実装するだけではそれぞれの処理コストが重なり処理性能が極端に悪くなる．

図 3 はユーザ関係に基づくスコアの計算は事前に行うことを前提として，ドキュメントの作成者の特定と，スコアの合成計算・リランキングの高速化を適用しない場合のヒット数と応答時間の関係を表している．共にヒットしたドキュメントの数だけ各操作が加わるので，応答時間はヒット数に比例して大きくなり，秒オーダー以上かかってしまう．

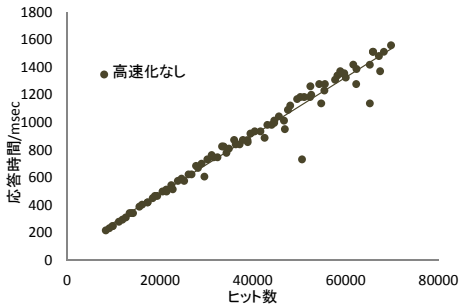

図 3　高速化を行わない場合のヒット数と応答時間の関係

そこで本研究では，Camel ら [8] が提案したユーザ属性の類似度と，ソーシャルネットワーク上のユーザ関係をドキュメントのスコアリングを利用することを前提として，性能問題の要因であるユーザ間の人間関係の計算および検索対象とユーザ間の関係の計算に対して，(1) 全ユーザ間の最短経路の事前計算によるユーザ関係スコアの高速取得，(2) 転置ファイルに作成者情報を埋め込む拡張による検索対象とユーザの間の関係の高速特定，(3) スコアの合成時に Threshold Algorithm [9] を適用することによる検索対象の top-k 検索の高速化，という 3 つの高速化を行う．そして，評価実験を通して高速化の性能を確認する．

本論文の構成は以下の通りである．2 章では提案するスコアリングの指標とスコアの高い上位 k 件を検索する処理の高速化手法についての説明を行う．3 章では性能評価を行いその結果に対する分析を行う．4 章では関連研究，5 章では結論を述べる．

2. 検索処理の説明と高速化手法の提案

2.1 スコアリングの指標

ドキュメントのスコアリングには，Camel ら [8] が提案したユーザ属性の類似度を表す Similality-based network とユーザ間の距離を表す Famiriality-based network の 2 つのソーシャルネットワークを参考に，ユーザ属性の類似度に基づく *Similality* とユーザ間の距離に基づく *Familiarity* という指標を用いる．またそれらに加えて，ドキュメントとクエリの適合度として *Relevancy* という指標も用いる．総合スコアは，式 (1) によって計算される．検索者を u，ドキュメントの作成者を $author(d)$，クエリを q，ドキュメントを d とする．

$$Score(u,q,d) = R(q,d) \times \{S(u,d) + F(u,d)\} \quad (1)$$

従来のクエリ検索を基本として，新たにソーシャルネットワークに起因するユーザ関係に基づく指標を組み合わせることでパーソナライズ検索を実現する．

Relevancy

TFiDF をベースとした，ドキュメントとクエリに基づくスコア指標を用いる．1 文書中に表れる単語 t の頻度である tf(term frequency) と全文書における単語 t が含まれる文書の出現頻度である df(document frequency) を組み合わせて，クエリに対するドキュメントのスコアリングを行う．N は全ドキュメント数である．

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

本論文が想定している検索対象はテキスト情報であるため，クエリとドキュメントの適合度に基づくこの指標を使用する．

Similarity

ユーザ関係の 1 つの指標として，ユーザ属性の類似度に基づくスコアを用いる．与えられた属性 $a \in A$(A はユーザ属性の集合) について，検索者とドキュメントの作成者が同一のユーザ属性を持つか否かを基に値を算出する．

$$S(u,d) = \frac{1}{|A|} \sum_{a \in A} f(u,d,a) \quad (3)$$

$$f(u,d,a) = \begin{cases} 1 & (u.a = author(d).a) \\ 0 & (u.a \neq author(d).a) \end{cases}$$

この指標はユーザ属性の類似度に基づいている．ユーザ属性とは，性別や年代，趣味などユーザのプロファイルから取得できるような情報を意味している．

Familiarity

ソーシャルネットワーク上のユーザ間の距離に基づくスコアである．ソーシャルネットワーク上の検索者とドキュメントの作成者のホップ数を用いるが, [7], [8] で利用されているユーザ間距離を用いる場合においても，2.2 で高速化手法を同様に適用することが可能である．

$$F(u,d) = \frac{1}{\log\left(Hop(u, author(d)) + 1\right)} \quad (4)$$

この指標はユーザ間の親密度に基づいている．ソーシャルネットワークは一意に定まるものではなく，観点やサービスによってあらゆる種類の人の繋がりが考えられる．例えば Friend 関係や follow/follower 関係は，ユーザがそれぞれの意図に基づいて関係を結ぶことになる．またコメントやリプライからコミュニケーションの有無によって形成されるネットワークなども考えられる．

2.2 ソーシャルサーチ

2.2.1 top-k ソーシャルサーチの定義

式 (1) に基づいてスコアの高いドキュメントの上位 k 件を検索する処理は，SQL 言語を用いた場合(注1)，図 4 のように記述される．

```
SocialSearch(user searcher, query q, integer k)
WITH RECURSIVE friend AS ( -- 検索者からの最短経路長の計算
  SELECT hop-friend.id, hop-friend.friendid, '1' AS hop
  FROM user, user AS hop-friend
  WHERE user.id = searcher
  AND user.friendid = hop-friend.id
UNION ALL
  SELECT user.id, user.friendid, friend.hop + 1 AS hop
  FROM user, friend
  WHERE user.id = friend.friendid)
-- 式 (1) のスコア式に基づく上位 k 件のドキュメント検索
SELECT   d.id, score(d.content, q, friend.hop)
FROM      document AS d, (
-- 検索者からの最短経路長の計算
      SELECT id, friendid, min(hop)
      FROM friend
      GROUP BY id, friendid)
WHERE    relevant(d.content, q)
AND      d.authorid = friend.id
ORDER BY 2  -- SELECT 句の 2 番目の項でソートする
LIMIT k;
```

図 4　top-k social search expressed in SQL statement

(注1)：集合論あるいは関数言語に基づく表記の方が SQL 言語よりも一般的であるが，top-k ソーシャルサーチの処理では SQL の高速化手法である実体化および検索処理における実行順序の最適化の考え方が適するため，ここでは SQL を用いる．

この SQL 文ではデータ構造として，ユーザの主キーと友人への参照をカラムとして有するユーザ表 user(id, friendid)，およびドキュメントの主キーとコンテンツをカラムとして有するドキュメント表 document(id, content) を利用する．1-9 行目は，SQL の再帰構文を用いることで，検索者 searcher を起点として推移的に到達できる全ユーザへのホップ数を計算する操作であり，結果は friend(id, friend, hop) の表として得られる．但し hop とは，検索者から各レコードが表すユーザへのグラフ上でのホップ数を表すカラムである．こうして得られる friend 表を用いて，式 (1) のスコア式に基づく上位 k 件のドキュメント検索は 11-19 行目のように表現される．この処理では，クエリ q に対して適合するドキュメント d を抽出し (17 行目)，ドキュメント d の作成者のレコードを friend 表から特定し (18 行目)，2.1 節で定義したスコアを関数 score を用いて計算し (11 行目)，スコアの高い順に上位 k 件のドキュメントを取得する (19,20 行目)．

2.2.2 top-k ソーシャルサーチの実行方法

図 4 の SQL 文で定義される top-k ソーシャルサーチを実行する方法は，どの部分クエリをどのように実体化するか，またどのような実行順序で検索を実行するかという点において多様な選択肢が考えられるが，ここでは高速な処理が可能だと考えられる 2 つの方法について説明する．1 つ目の方法は，全ドキュメントを検索できる 1 つの転置ファイルを用いる方法である．この方法では，図 4 の 17 行目にある relevant 関数の処理を高速化するため事前に転置ファイルを構築する．全ドキュメント d，全単語 t に対して relevant(d, t) の結果を t 毎に実体化した結果が転置ファイルであり，またクエリ q は複数の単語から構成されるため，図 4 の 17 行目の relevant(d.content, q) の処理は転置ファイルを用いて高速に処理することができる．この手法では，更に 1) ユーザ表から得られるソーシャルネットワークを用いて n 対 n の最短経路を事前に計算することで，任意の検索者に対する friend 表を実体化し，2) 転置ファイルを用いて検索されたドキュメント群に対して作成者を特定する処理 (18 行目の d.authorid) を高速化するため，ドキュメント毎に authorid を付与するよう転置ファイルを拡張する(注2)．n 対 n の最短経路の事前計算については 2.3.1 節，転置ファイルの拡張については 2.3.2 節で詳述する．こうした実体化結果を活用することで，top-k ソーシャルサーチは次のように実行される．

（1） 転置ファイルを利用してクエリ q に適合するドキュメント集合を特定する．

（2） 転置ファイル内に格納された authorid を利用してドキュメント毎に作成者を特定する．

（3） 転置ファイルの結果から $R(q,d)$ の高いドキュメント毎に，特定した作成者情報を利用して実体化した friend 表から $F(d,u)$ (検索者からドキュメント作成者 u へのホップ数) を得て，式 (1) を適用して合成スコアを得る．ここで，TA を用いることで上位 k 番目までのドキュメントの順序が確定した時点で検索処理を打ち切ることができる．TA を用いた高速化については 2.3.2 節で詳述する．

この 1 つ目の方法の特徴は，1 つの転置ファイルで全ドキュメントを検索することができるので，クエリの種類に依存せずに平均的な処理性能を得ることができることである．

2 つ目の方法は，作成者毎にドキュメント群を分割して転置ファイルを構築する方法であり，転置ファイルのアクセスの順は検索者の友人から開始してソーシャルネットワークをたどって探索空間を広げる方法である．この方法は，図 4 の SQL 文において 11-20 行目の friend 表に対する操作を，1-9 行目の friend 表を再帰的に構築する操作内にプッシュダウンすることで実現される．具体的には次のように検索は実行される．

（1） 検索者の友人を特定し (図 4 の 2-5 行目の記述)，各友人 u が作成者であるドキュメントを検索するため各友人毎の転置ファイルを検索してクエリ q に適合するドキュメント集合を特定する．

（2） 友人 u に対する $F(u, author(d))$ を得て，転置ファイルの結果から $S(q,d)$ の高いドキュメント毎に式 (1) を適用して合成スコアを得る．

（3） ソーシャルネットワークをたどって隣接する新たな友人を特定し (図 4 の 7-9 行目の記述)，(1)(2) の処理を繰り返し実行する．ここにおいても，TA を用いることで上位 k 番目までのドキュメントの順序が確定した時点で検索処理を打ち切ることができる．

この方法の特徴は，ヒット数が多いクエリならば，検索者近隣のソーシャルネットワークに探索範囲が局所化されるため，計算コストが低く抑えられることである．しかし，ヒット率が低いクエリだと探索範囲がソーシャルネットワーク全体になってしまうことから，特に大量のユーザ数の際に転置ファイルのフラグメントに起因する性能問題が生じてしまう．例えば，論文のタイトル全体をフレーズ検索するような検索では，クエリ長が長く且つ単語の出現順序が制限されるためヒット数が小さくなり，結果としてソーシャルネットワークを広範囲に探索しなければならなくなる．

ソーシャルサーチのような web 上でのコンテンツ検索では，大量のユーザ数を対象としてクエリの種類に依存せず平均的な処理性能が得られることが望ましいため，本論文では 1 つ目の方法を用いる．しかし，この方法を導入することで，主に 3 つの要因によって処理コストが増加することが考えられる．その対策について 2.3 で詳細に説明する．

2.3 高速化手法

2.3.1 最短経路の事前計算

1 つ目の要因は，ソーシャルネットワーク上のユーザ間距離の計算である．本論文ではユーザ関係を表すスコアである Familiarity の要素として検索者とドキュメントの作成者のホップ数を用いている．

ユーザをノード，繋がりをエッジとした場合，ホップ数はエッジを全て同一に扱いそれぞれに異なる重み付けはしない．よってユーザ数を n とした場合，1 対 n の最短経路コストを求めるには幅優先探索を行う．n 人すべてのホップ数を求めるには，

(注2)：この拡張は実体化結果を非正規化する手法に相当する．

図 5　k=3 の場合の TA 適用イメージ

起点を変化させて n 対 n の最短経路コストを求めればよい．1 回のクエリ検索の度に毎回この計算を行なう事は処理コストが高い．ため，予め全 2 ユーザ間の最短経路を事前計算しておくことで，検索実行時の処理コストを削減する．計算には [10] のアルゴリズムを用いた．

2.3.2　転置ファイルの拡張

2 つ目の要因は，ドキュメントの作成者の特定時のインデックスへのアクセスである．ドキュメントのスコアリングにユーザに基づくスコアを導入することにより，ランキングを決定する前の段階でドキュメントの作成者を特定する必要性が生じることになる．一般的な転置ファイルでは，ドキュメントの通し番号と，スコアの組である，$\langle doc\#, tf,$ 単語の位置情報 $\rangle$ 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 authorid を埋め込み，$\langle doc\#, tf, authorid,$ 単語の位置情報 $\rangle$ 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，高速にドキュメントの作成者が特定できるようになる．

2.3.3　TA による top-k ソーシャルサーチの効率化

3 つ目の要因は，スコアの合成である．本研究では，ドキュメントのスコアを式 (1) によって決定するので，クエリ-ドキュメントベースのスコアである Relevancy とソーシャルネットワーク上のユーザ関係ベースのスコアである Similarity と Familiarity という複数のスコアを考える．ランキングの際にはそれらを合成する操作を行わなければならない．ここでは，Threshold Algorithm(TA) を適用することで，スコア合成時に不必要な計算を省略する．計算は Relevancy スコアの大きい順に行ない，ドキュメントの作成者を特定した後，作成者に関する Similarity スコアと Familiarity スコアを取得し式 (1) を基に合成する．また TA を適用するには，k 番目のスコアの保持と Relevancy スコアを固定した場合の式 (1) の値を最大化するような Similarity スコアと Familiarity スコアの和を予め知っておく必要がある．

図 5 は TA を適用した場合の Top-3 検索の例を表している．常に上位 3 番目のスコアを保持し，スコア合成の計算前に予想される最大値との比較を行う．例えば，ドキュメント d5 の計算を行う時に，まずスコアの合成式である式 (1) に基づいて，予想される最大値の計算を行うとその値は 9.6 となる．d4 まで計算を終えた時点での 3 番目のスコアは 10 であるので，最大値を取ったとしても上位 3 件のランキングが変化することはない．またドキュメントは Relevancy スコア順に並んでいるため，d6 以降のスコアの最大値は d5 のスコアの最大値を超えることはない．よってこの場合 d5 以降のスコアの合成計算を打ち切ってよい．

2.4　top-k ソーシャルサーチの計算量

検索処理の全体の計算量は，全ドキュメントの単語数を t，クエリに含まれる単語数を w，ヒット数を h，top-k 確定のために必要なスコア計算をするドキュメント数を l とすると，表 1 のように書ける．転置ファイルアクセスは，全単語の中からクエリに含まれる各単語を検索し，ヒットする文書を転置ファイルから検索する．スコア合成については，合成計算を行ったものを逐次，スコアに関する優先度つきキューに格納することでランキングを保持している．ただし，実際には $w \log t << h$ なので，全体の計算量は $h + l \log l$ となる．また，TA の効果はクエリや k の値によって変化するため，TA の効果が全く無いようなケースでは $l = h$ となる．

表 1　各フェーズと全体の計算量

	計算量
転置ファイルアクセス	$O(w \log t + h)$
スコア合成	$O(l \log l)$
全体	$O(w \log t + h + l \log l)$

3.　性 能 評 価

性能評価は主にクエリに対して，その検索結果が決定するまでの応答時間を測定することで行う．本論文で提案した転置ファイルへのユーザ情報を保持する拡張とスコア合成時の TA の適用のそれぞれの効果と有用性を確認する．なお，インデックスの拡張に関する提案手法については，拡張前後での転置ファイルのサイズと転置ファイルアクセス時間の比較も行う．

性能測定環境は表 2 の通りである．

表 2　性能測定環境

CPU	Intel Core2 Duo 2.66GHz
Memory	4GB
OS	Windows Vista
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

3.1　データセット

性能評価には表 3 のデータセットを用いた．2011 年 3 月 6 日から 2011 年 3 月 17 日までに約 5 万人のユーザーによって Twitter に投稿された 1500 万ドキュメントである．ソーシャルネットワークについては，このユーザー集合が閉じたネットワークを形成しているとして，実際のソーシャルネットワークの特性を模した人工的に作成したものを利用した．また，OS のファイルキャッシュの効果が得られるよう，応答時間は各ク

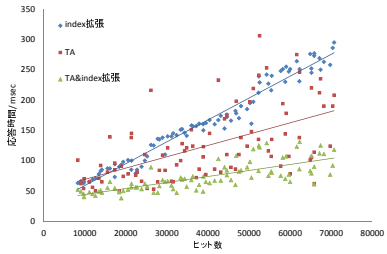

図 6　Top-10 検索における高速化の効果

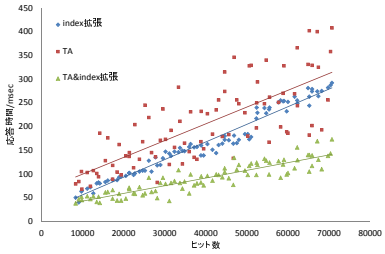

図 7　Top-100 検索における高速化の効果

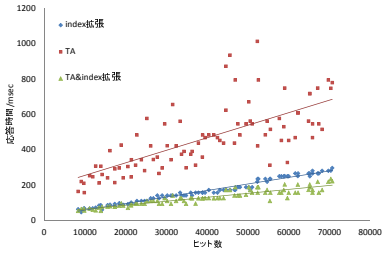

図 8　Top-1000 における高速化の効果

エリ同条件で 10 回計測した内の最善値を選択した．性能評価には用いたデータセット中で 7000 から 70000 ヒットする単語を 90 個選択し，各単語を 1 クエリとして使用した．

表 3　Twitter データセット

ユーザー数	エッジ数	エッジ数の平均	文書数
50011 人	2262486 本	45.2 本	15168808

3.2　提案手法の効果と分析

インデックスの拡張は転置ファイルにユーザ ID を埋め込むため，その分のファイルサイズの増加が見られる．ユーザ ID はすべて整数値であるため爆発的なサイズの増加は見られないが，表 4 が示すように拡張前後で約 16 %のファイルサイズの増加，また約 14 %のアクセス時間の増加が見られた．なおアクセス時間は，ユーザ関係を取り入れる操作を除いた場合の全 90 クエリの平均の応答時間である．

表 4　拡張前後の転置ファイルのサイズとアクセス時間

	拡張前	拡張後
ファイルサイズ	5.6 GB	6.5 GB
アクセス時間	34.9 msec	39.8 msec

図 6,7,8 は k=10，100，1000 検索において，2.3.2 で提案したインデックスの拡張のみを適用した場合 (index 拡張)，2.3.3 で提案した TA のみを適用した場合 (TA)，両方適用した場合 (TA&index 拡張) それぞれの応答時間を比較した．インデックスの拡張を行ったものについてはヒット数と応答時間の間に線形性が見られる．一方で TA のみ適用した場合はヒット数と応答時間の間の関係に大きなばらつきが見られる．これは TA を適用した場合足切りが行われるので，ヒット数と実際にスコア合成の計算を行ったドキュメントの数は異なり，ヒット数ではなくスコアの分布に依存するためである．スペースの都合のため詳細は割愛するが，別途行った実験の結果，実際に計算した数と応答時間の間には線形性が見られる．2.4 で述べたように，検索処理の計算量は $O(h + l \log l)$ で表されるが，TA の効果が見られる場合は l の値が小さくなり $l \log l < h$ となるので，提案手法がヒット数 h に比例する特徴が確認できる．

いずれもインデックスの拡張と TA の両方を適用した場合が最も短い応答時間であり，k=10，100 のケースではヒット数が大きくても 100msec 前後の応答時間を実現しているため，実用的に十分高速な性能を達成している．

図 9 は性能評価に使用した全クエリの応答時間の平均である．3 つの高速化パターンについて k=10，100，1000 の場合を比較している．また，検索フローは大きく転置ファイルアクセス (Step1) とスコア合成・リランキング (Step2) の 2 つに分けて考えることができる．本論文で提案した手法は主に Step2 に関する高速化手法である．

Step1 は高速化手法や k の大きさに依存せず，すべてのパターンについて等しい時間がかかる．Step2 は主に k の増加に依存して処理時間も増大するが，TA と Index 拡張の両方を行うことで，k=10 の場合 Step1 の約 0.8 倍，k=100 でも 1.3 倍のコストで処理が可能である事がわかる．

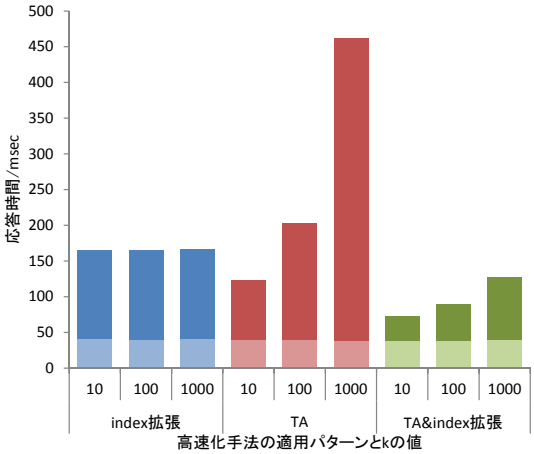

図 9　適用する高速化手法と k の違いによる平均応答時間の比較
薄色:転置ファイルアクセス，濃色:スコア合成・リランキング

図 10 は k の違いによる TA の効果を表している．厳密には TA の効果は上位 k 件周辺のスコアの分布に依存するが，ヒット数における k の割合が小さいほど TA の効果が高いことが見て取れる．k の割合が大きくなるに連れ TA の効果が小さくなり，クエリによっては，TA を適用したとしてもヒットしたドキュメント全てに対してスコア合成の計算を行わなくてはならないケースもあった．

また，Step2 の計算量は $O(l \log l)$ で表されるが，図 10 からも l は k の値に依存していることが分かる．よって，実質的に Step2 の応答時間に $O(k \log k)$ の傾向が見られる．

4.　関 連 研 究

4.1　パーソナライズドサーチ・ソーシャルサーチ

Camel ら [8] はソーシャルサーチを，ソーシャルデータを用いた検索のプロセスであると定義している．ソーシャルデータ

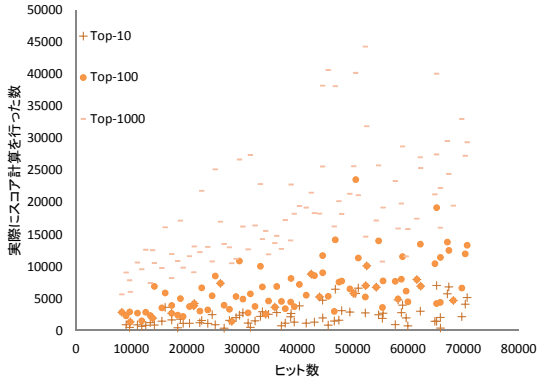

図 10　Top-k 検索における k の違いによる TA の効果

とはソーシャルブックマーク，ブログ，SNS 等から収集された情報のことである．*SBRank* [11] は，ソーシャルブックマーキングサービスから，そのページをブックマークしたユーザの数を抽出しその値をページのランキングに取り入れることを提案し，*PageRank* との比較・分析を行った．

また，ソーシャルサーチエンジン *Aardvark* [6] は本研究とは異なり，人を検索対象としているが，スコアリング指標の 1 つに *Social connection* として共通の友人や所属等からソーシャルネットワークを形成し人物のランキングに取り入れている．

Bender ら [7] は *del.icio.us* と *Flickr* において，ユーザのタグ付行為の類似度を表す *TagSim* とソーシャルネットワークに PageRank の考え方を適用した *UserRank*，ソーシャルネットワーク上のユーザー間の距離を表す *Friendship* を組み合わせた指標を検索に用いることを提案している．Carmel ら [8] もまた，ユーザ属性の類似度を表す *Similarity-based netwok* とユーザ間の親密度を表す *Familiarity-based network* の 2 つの異なる基準によって導き出されるソーシャルネットワークをドキュメントのスコアリングに取り入れる事を提案している．

また検索要件に対して最適化されたインデックスの方式に関する研究もなされている．Anh ら [12], [13] はスコアの高いドキュメント順に転置ファイルを構築することによって，複数単語が選言的に指定された top-k クエリ検索の際に，転置ファイル探索の処理を足切りをすることで高速な検索技術を提案している．，Chen ら [14] は Twitter において情報の新規性の観点に立った効率的なインデックス方式を提案している．クエリと適合度の他にユーザの PageRank やタイムスタンプ，リプライ構造などの指標を利用してランキングを行うために，それらの情報を転置ファイルに埋め込んでいる．またランキングにおいて情報の新規性に重点を置くことから，タイムスタンプを基に転置ファイルを分割しブロック単位で検索を行う方式をとっている．

5. おわりに

本論文では，ソーシャルネットワーク上の検索者とドキュメントの作成者の関係を検索に取り入れることを前提として，以下の 3 つの高速化に対する提案を行った．(1) 全ユーザ間の最短経路の事前計算によるユーザ関係スコアの高速取得，(2) 転置ファイルに作成者情報を埋め込む拡張による検索対象とユーザの間の関係の高速特定，(3) スコアの合成時に Threshold Algorithm を適用することによる検索対象の top-k 検索の高速化，これらを考慮した検索システムを構築し，評価実験を行った結果実用的に十分高速な性能が達成されていることが確認できた．

今後は企業内 SNS と Twitter のデータを用いて被験者実験を行うことで，情報検索にソーシャルネットワーク上のユーザ関係を考慮することの有効性を示す．さらに，ソーシャルネットワークを用いたパーソナライズ検索の観点から，ユーザ関係スコア (特に Familiarity) の導入効果を検証する．また，ソーシャルネットワークを直近の友人のみに限定する場合や，ソーシャルグラフ全体の人の繋がりを利用した場合等との比較によって，考慮するソーシャルネットワークの範囲による有効性に対する考察を行うことを考えている．

謝　辞

実験データを提供して頂いた兼山元太氏 (クックパッド株式会社) に感謝する．

文　献

[1] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. In *Journal of Documentation 28*, 1972.

[2] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at trec-3. In *Proceeding of TREC*, 1994.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceeding of WWW*, 1998.

[4] Google. The official google blog - introducing google social search: I finally found my friend's new york blog! howpublished = http://googleblog.blogspot.com/2009/10/introducing-google-social-search-i.html, year=.

[5] Google. The official google blog - search, plus your world. http://googleblog.blogspot.com/2012/01/search-plus-your-world.html, 2011.

[6] Damon Horowitz and Sepandar D. Kamvar. The anatomy of a large-scale social search engine. In *Proceeding of WWW*, 2010.

[7] Matthias Bender, Tom Crecelius, Mouna Kacimi, Sebastian Michel, Thomas Neumann, Josiane Xavier Parreira, Ralf Schenkel, and Gerhard Weikum. Exploiting social relations for query expansion and result ranking. In *Proceedings of ICDE Workshops*, 2008.

[8] David Carmel, Naama Zwerdling, Ido Guy, Shila Ofek-Koifman, Nadav Har'el, Inbal Ronen, Erel Uziel, Sivan Yogev, and Sergey Chernov. Personalized social search based on the user's social network. In *Proceeding of CIKM*, 2009.

[9] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proceedings of ACM PODS*, 2001.

[10] Ulrik Brandes. A faster algorithm for betweenness centrality. In *Journal of Mathematical Sociology 25*, 2001.

[11] Yusuke Yanbe, Adam Jatowt, Satoshi Nakamura, and Katsumi Tanaka. Can social bookmarking enhance search in the web? In *Proceeding of JCDL*, 2007.

[12] Vo Ngoc Anh and Alistair Moffat. Impact transformation: effective and efficient web retrieval. In *Proceeding of SIGIR*, 2002.

[13] Vo Nagoc Anh, Owen de Kretser, and Alistair Moffat. Vector-space ranking with effective early termination. In *Proceeding of SIGIR*, 2001.

[14] Chun Chen, Feng Li, Beng Chin Ooi, and Sai Wu. Ti: An efficient indexing mechanism for real-time search on tweets. In *Proceeding of SIGMOD*, 2011.