ソーシャルメディア上の情報拡散データの リアルタイム問合せ処理評価

榎 美紀†‡　吉田 一星†　小口 正人‡

†日本アイ・ビー・エム(株) 東京基礎研究所
‡お茶の水女子大学理学部情報科学科 〒112-8610 東京都文京区大塚 2-1-1
E-mail: †enomiki@jp.ibm.com

あらまし Twitter に代表されるマイクロブログサービスでは，リアルタイムにメッセージが多く発信され，他のユーザーへの返信や再共有(リツイート，RT)等のユーザー間のやりとりも多く存在する．メッセージが多くのユーザーに RT されて情報が拡散すると，その拡散規模や内容によっては，現実社会に与えるインパクトも大きい．それゆえ，ソーシャルメディア上で今何が広く拡散しているのか，を知ることは，企業や団体にとって重要である．本論文では，情報拡散データをストリームデータとして処理してリアルタイムに分析するためのシステムのフレームワークを構築し，代表的な問合せパターンの性能評価を行う．また，ツイートの拡散収束のタイミングを，拡散モデルを用いて推定してインメモリデータストアをメンテナンスする手法を提案し，有効性を評価する．

キーワード ソーシャルメディア, 情報拡散,インメモリデータベース, Twitter

Performance Evaluation of Real-time Query Processing of Information Diffusion on Social Media

Miki ENOKI†‡　Issei YOSHIDA†　and Masato OGUCHI‡

†IBM Research - Tokyo

‡Ochanomizu University　2-1-1 Otsuka, Bunkyou-ku, Tokyo, 112-8610, JAPAN
E-mail: †enomiki@jp.ibm.com

1. はじめに

1.1. ソーシャルメディアの情報拡散

Twitter に代表されるマイクロブログサービスでは，リアルタイムなメッセージ発信や，他のユーザーへの返信(Reply) や再共有(リツイート， RT) 等のユーザー間のやりとりも多い．メッセージが多くのユーザーに RT されて情報が拡散すると，短時間でソーシャルメディア上のユーザーにその情報が共有されることになる．その拡散規模や内容によっては，現実社会に与えるインパクトも大きい [1].

それゆえ，ソーシャルメディア上で今どんな情報が広く拡散しているのか，をリアルタイムに知ることは，企業や団体にとって炎上防止や流行把握のために重要である．これらは現状では，人手で人海戦術によるモニタリングを実施するか，あらかじめ登録したキーワードのバーストを発見して異常検知する商用サービスを利用することが近年の企業のソーシャルメディア活用の傾向である[2].

ソーシャルメディアの情報をリアルタイムに分析してバーストやイベントを検知する研究は多く存在する．Twitter のメッセージ内容を解析して特徴的なキーワードを抽出し，その頻度のバースト性により，今何がトレンドとなっているかをモニタリングする[3,4]. イベント情報を検知するためには位置情報やメッセージ内容を分析して，そのキーワードやツイート発信場所のバースト性により，今どんなイベントが発生しているかを発見する[5]. これらのサービスや研究は，各ツイッターのメッセージに出現する「キーワード」の増減の情報を基にした分析である．必ずしもツイート間には繋がりはなく，同じキーワードを話題にしているという状態を分析対象にしている．

対して，我々が分析対象とするのは，メッセージの再共有で広がっていく情報拡散である．あるツイートが多数のユーザーに RT されて広く拡散したメッセージは，多くのユーザーが興味をもち，インパクトを与えた情報であるといえる．また，あるトピックに関する複数のツイートの拡散データを対象にして，それらのツイートをよく RT しているユーザー達や，逆によ

く RT されているユーザー等,「キーパーソン」を見つけることにより,「どのようなユーザーが興味をもっているか」「誰が話題の中心になっているか」「どのようなユーザー間の流れを介して情報が拡散しているのか」という事を発見することが期待される.

発見したユーザーはユーザープロファイリングなどの分析を行うことにより[6], 人となりを深く分析可能になる. 我々はこのような情報拡散データの分析を実現するためのシステム構築を目的としている.

1.2. リアルタイムストリーム処理

Twitter のメッセージはリアルタイムに逐次ユーザーから発信されるため, ストリームデータとして捉えることができる. ストリームデータをリアルタイムに分析する研究がこれまでにも行われてきている[20,22,23,24].

ストリーム処理の特徴は, ストリームデータの数や時間によってウィンドウ幅を設けてデータを区間に区切り, その範囲に含まれるデータを分析対象にする. 拡散データをストリーム処理で扱う場合, 逐次到着する RT を時間ウィンドウで切り, RT 元であるオリジナルツイートの ID 単位でデータを区分化して処理すると, 図 1 のようになる. あるツイート(Tweet1)が投稿された後, 時間経過と共に他のユーザーが RT すると, Tweet1 の RT のスレッドが発生して, RT の情報が蓄積される. Tweet2 も同様に, 他ユーザーから RT されて情報が追加されていく.

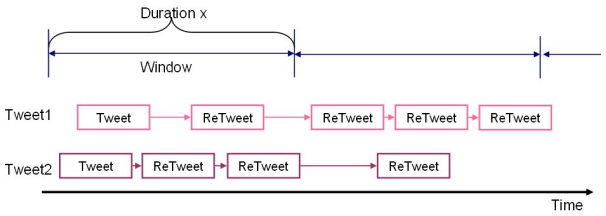

図 1　リツイートのストリーム処理

Figure 1　Stream processing of ReTweet

ここで, ウィンドウ幅を 1 時間として, 各ツイートの RT をカウントした場合, 過去 1 時間の範囲での RT 総数が分かる. しかしながら, それ以前のデータはストリームシステムには残されないため, 各ツイートの拡散ネットワークは時間で分断された状態になる. したがって, あるツイートが発信されたから, RT が広がり, それが収束するまでの, 拡散データ全体を捉えた分析をすることが困難である. ツイートの拡散は, 短期間で爆発的に拡散するものや, 時間経過と共に少しずつ拡散するもの等様々である[7]. 短期間で爆発的に拡散した場合, ウィンドウで切った場合でも頻繁に RT された瞬間を捉えて, 多数 RT されたツイートとして発見できるかもしれないが, 少しずつ時間をかけて RT が広がった場合は, 時間単位の RT 数はそこまで多くならず, 発見されない可能性がある.

そこで我々は, 拡散データをインメモリのデータストアに一時的に格納して, リアルタイムに拡散分析を行うシステムを提案する. ストリーム処理は一般的に事前にクエリを登録して, 逐次到着するデータに対してクエリを発行するが, インメモリに蓄積されているデータストアを対象にすることで, 分析ユーザーがインタラクティブにクエリを発行することも可能になる.

一方で, データストアに格納した場合, サーバーのメモリサイズには限りがあるため, 延々と蓄積し続けることは現実的ではない. また, すっかり RT されなくなった鮮度の低い拡散データなどが残り続けて分析対象になってしまうことが懸念される. そこで, 我々は各ツイートに最適な時間ウィンドウ幅を設定するアルゴリズムを提案する. これにより, 今も RT され続けているアクティブな拡散データはデータストアに出来るだけ格納し続け, 拡散が収束したデータはデータストアから退避するようなメンテナンス処理を実現する.

本論文での我々の貢献は以下である.

- 拡散データをストリームデータとして処理してリアルタイムに分析するためのシステムのフレームワークの生成
- ツイートの拡散収束のタイミングを, 拡散モデルを用いて早期に推定し, 独自の時間ウィンドウ幅を設定してインメモリデータストアをメンテナンスする手法の提案と評価
- Twitter の拡散データを実際に用いて, 代表的な問合せパターンの性能評価. 複雑なクエリを高速化するための事前ビューを導入した時の効果の評価

以降, 2 章では, 我々が提案する情報拡散分析システムについて, 3 章では, 拡散分析の代表的な問合せパターンとデータアクセス高速化について述べる. 4 章ではツイート毎にカスタマイズしたウィンドウ幅を用いた拡散データのメンテナンス手法を提案する. 5 章では実際に Twitter のデータを用いて本システムの拡散データアクセスの性能とメンテナンス手法の評価を行う. 6 章で関連研究について述べ, 7 章でまとめる.

2. 情報拡散分析システム

2.1. 情報拡散データ

ある一つのツイートに対して, それを RT しているリツイートデータを関連付けて蓄積していくと, 1 つの情報拡散データとなる. RT をエッジとし、RT したユーザーとされたユーザーをノードとするグラフ構造

を拡散ネットワークと呼ぶ．拡散ネットワークを可視化すると，拡散の規模や拡散経路が視覚的に捉えられて直感的に理解しやすくなる．

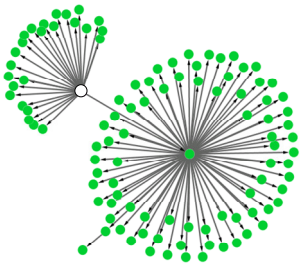

図 2　拡散ネットワーク
Figure 2　Information diffusion network

図 2 はオリジナルのツイートを発信したユーザー(中心が白色のノード)と，そのツイートを RT したユーザー(色の付いたノード)をネットワークのノードとし，エッジは情報の流れを表している．例えば，ユーザー@b がユーザー@a のツイートを RT した時，@a のノードから@b のノードへ向かうエッジがはられる．つまり，1 エッジが 1 リツイートに該当する．

2.2. 情報拡散分析システム概要

本研究のシステム構成を図 3 に示す．Twitter から発信されるツイートをリアルタイムに取得し，インメモリのデータストアに格納していく．Twitter から日々発信されるツイートは膨大な量であるため，拡散データの収集対象とするツイートをフィルタリングしても良い．例えば，特定のユーザーが発信するツイートの拡散や特定のキーワードが含まれるツイートの RT のみを格納するように指定する．

システムのデータストアには，リレーショナルデータベース，グラフデータベースやキー/バリューストアが候補としてあげられる[8,9]．分析ユーザーが人気のリツイートやユーザーを発見するためには集約やソートの処理が必要であり，キー/バリューストアよりも，SQL を用いて複雑なクエリが実行できるリレーショナルデータベースのほうが分析の幅が広がると考えるため，本研究ではインメモリデータベースを採用する．データストアから退避されることになった拡散データは，HDD のデータベースに格納して，Historical データを対象としたオフラインの分析に利用するか，そのまま破棄する．

アプリケーションサーバーには，拡散ネットワークを分析するための複数のモジュールが入り，分析ユーザーはインタラクティブに分析を実施する．例えば，分析ユーザーは特定のツイートの集合に対して，多く RT されている人気のユーザーやツイートを発見する．定期的にクエリを発行してモニタリング目的に使用したり，分析ユーザーが指定したツイートの拡散ネットワークを生成し，可視化することも可能である．

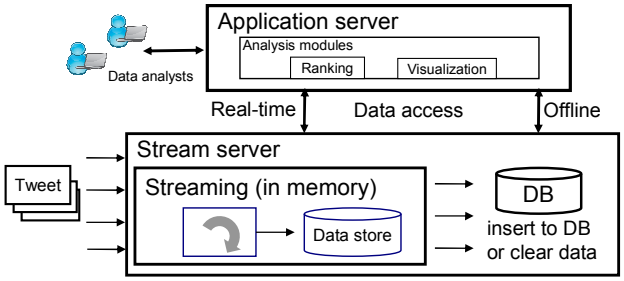

図 3　情報拡散分析システム
Figure 3　Infomation diffusion analysis system

3. 代表的な問合せパターンと高速化

3.1. インメモリデータベース

データベースにはリツイートのデータを格納していく．インメモリのデータベースの実装はいくつか存在し，オープンソースの H2, Apache Derby や，商用の TimesTen, solidDB などがある[21].

データベースには，RETWEET テーブルと，ORIGIN_TWEET テーブルを生成する（図 4）．RETWEET テーブルには，RT のツイートの ID(TweetID)，オリジナルツイートのツイート ID(RTID)，RT を発信したユーザー名(Dst)，RT 元であるオリジナルツイートを発信したユーザー名(Src)，RT 発信時刻(Time)，使用言語(Lang)，位置情報(Location)等を属性として持たせる．1 レコードが 2.1 節の 1 エッジに該当する．

RETWEET table

TweetID	RTID	Time	Src	Dst	Lang	Location	‥
100	1	Time data	u1	u2	Ja	GPS	‥
101	1	Time data	u2	u4	Ja	GPS	‥
‥	‥	‥	‥	‥	‥	‥	‥

ORIGIN_TWEET table

TweetID	Time	User	Msg	RTcount	‥
1	Time data	u1	message	29	‥
2	Time data	u5	message	14	‥
‥	‥	‥	‥	‥	‥

図 4　拡散ネットワークデータベース
Figure 4　Diffusion network database

ORIGIN_TWEET には，ツイート ID (TweetID)，ツイート発信時刻(Time)，発信ユーザー名(User)，ツイートのメッセージ(Msg)，RT された回数(RTcount)がある．RTcount は，対応する RT を受信するたびにカウントされる．ある RT の，RT 元となるオリジナルツイートの情報を取得したい場合は，RETWEET テーブルの RTID

と，ORIGIN_TWEET テーブルの TweetID を Join 結合して問合せを行う．

3.2. 拡散分析のための問合せパターン

本システムにて拡散データを対象にした代表的な問合せパターンを以下に紹介する．

(1) 指定したツイートの拡散ネットワークを取得

拡散ネットワークを生成するため，所望の拡散データを問合せる．これにより，「ユーザー間をどのような経路で RT が拡散していったのか？」といったような情報を得ることができる．また，生成した拡散ネットワークは，クラスタリングや頻出経路発見などのネットワーク分析に応用することも可能となる．この問合せパターンに対応する SQL は以下のようになる：

```
[Query 1]
SELECT      Src, Dst
FROM        RETWEET
WHERE       RTID in (tweet ids)
```

インプットはツイート ID であり，この ID は分析モジュールにて指定されるか，もしくはデータベースから取得される．例えば，データベース内の ORIGIN_TWEET テーブルに対して，あるトピックを表す特定のキーワードが含まれるツイートのツイート ID リストを取得し，そのリストを Query 1 のインプットにすることにより，そのトピックに関連する拡散ネットワークを取得する．

(2) 人気のあるツイートを取得

RT 数の多い順のツイートランキングを問合せる．この問合せパターンに対応する SQL は以下のようになる：

```
[Query2]
SELECT      *
FROM        ORIGIN_TWEET
WHERE       TweetID in (tweet ids)
ORDER BY    RTcount   DESC
 FETCH FIRST 10 ROWS ONLY
```

WHERE 句にてツイート ID を指定しない場合は，単純に現在格納している拡散データの中で，最も RT 数の多い上位 10 件が出力される．

(3) キーパーソンとなるユーザーを取得

RT された，もしくは RT した数の多い順のユーザーランキングを問合せる．この問合せパターンに対応する SQL は以下のようになる：

```
[Query3]
SELECT      Src, count(Src)
FROM        RETWEET
WHERE       TweetID in (tweet ids)
GROUP BY    Src
ORDER BY    count(Src)   DESC
FETCH FIRST 10 ROWS ONLY
```

Query 3 にて Src を指定した場合は，指定したツイートの集合の中で，最も RT された総数の多い順のユーザー(＝拡散影響力の高いユーザー)が出力される．一方，Dst を指定した場合は，指定したツイートの集合の中で，最も RT していた総数の多いユーザー(=情報を RT しやすいユーザー)が出力される．

以上のような問合せは，インメモリデータベースで処理されるためディスクベースのデータベースよりもデータ処理が高速になることが期待される．しかしながら，予めインデックスを張ったカラムに対するシンプルな問合せは高速に処理可能であるが，ソートや Join，副問合せ等メモリ上でのデータ演算が問合せコストの多くを占めるような複雑な SQL になるほど，処理能力が HDD のデータベースと同程度まで下がってしまう可能性がある [10].

本システムの場合，Query 3 の問合せ処理は，ユーザー単位での集約，ソートの処理等が必要になるため，やや複雑な SQL となる．

3.3. ランキング計算のためのデータアクセス高速化

Query 3 のようなランキング計算を伴う問合せは，Top-k 計算の最適化アルゴリズムを応用できる．そこで我々は，ランキング計算のための事前処理結果を独自のビューとして保持し，問合せの高速化を実現する．図 5 は，RT されたユーザーのランキングを計算するためのビューである．オリジナルツイートの ID ごとに，RT されたユーザーと RT された回数のペアが．RT 回数の多い順にソートされてリストで保持されている．

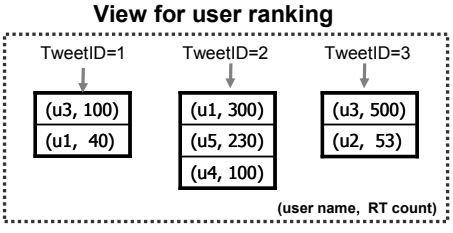

図 5　ユーザーランキング計算のためのビュー

Figure 5　View for user ranking

このようなユーザー単位に集約したビューを用いると，top-k 計算の代表的な効率的な計算アルゴリズムである，Fagin ら[11]の Threshold Algorithm を適用することができる．

以下に，「ツイートの集合を対象にして，RT された回数によるユーザーのランキング計算」を実現する場合を例に問合せ処理の手順を説明する．ランキングは，

上位 k 件を出力するものとする．

[top-k 検索のためのビュー作成]

各オリジナルツイート ID（t とする）に対し，以下の検索ビュー S(t)，検索インデックス R(t) を作成する

S(t):ユーザーと被 RT 数の値の組(u, w)を w 値の降順に並べ，値の大きい順から各組に順次アクセス可能なリスト

R(t):任意のユーザーx に対して、S(t) 内の組 (x, w) を定数時間で取得するためのランダムアクセス検索インデックス．任意のマップ構造で生成可能

図 5 の例では，S(1) = [(u_3, 100), (u_1, 40)], S(2) = [(u_1, 300), (u_5, 230), (u_4,100)], S(3)=[(u_3, 500), (u_2, 53)] となる．

[top-k アルゴリズム]

ツイートの集合(TweetID のリスト)を {t_1, t_2, …, t_m} として，被 RT 数の多いユーザー上位 k 件を出力する手順を以下に示す．

```
Input  S(t), R(t), Tweetid_list = [t1, t2, …, tm],k
Output result_candidate  //top k user list
1.   result_candidate = [];
2.   For each i=1,...,m, Do
   1.   If !S(ti).hasNext() Then continue;
   2.   Retrieve (ui, wi) from S(ti);
   3.   If result_candidate.contains(ui) Then continue;
   4.   V := sum of w in each R(tj) for ui // j=1,...,m
   5.   Vk := sum of w in the k-th candidate in
        result_candidate;
   6.   If (V > Vk) Then update result_candidate      by
        (ui, V).
   7.   V' := sum of the minimal value of w in S(ti)
        that have been retrieved so far // i=1,...,m
   8.   If (V' >= Vk) Then break;
3.   End For
```

ビューを導入することにより，問合せ処理の高速化を実現できるが，その分サーバーのメモリ領域を消費することになり，トレードオフの関係となる．クエリの頻度などを参考に，どのランキング用のビューを作成するべきか考慮する必要がある．

4. データストアメンテナンスのための，情報拡散モデルを用いた拡散収束予測の提案

1.2 節で述べたように，インメモリのデータストアを採用することでツイートの拡散データを格納しておくことができる．しかし逆に，古くなった拡散データをデータストアから退避させてメンテナンスする必要がある．本章にて，ツイートごとにカスタマイズした時間ウィンドウを設定する手法を提案する．これにより，ツイートが頻繁に RT されている間はデータストアに拡散データが格納され続け，RT が収束してきたらデータストアから退避される．

図 6 はカスタマイズした時間ウィンドウの例を表している．時間ウィンドウ幅を，各ツイートが発信されて，RT が続いて，それが収束するまでに設定することで，データストアには全体の拡散データが格納されると期待される．

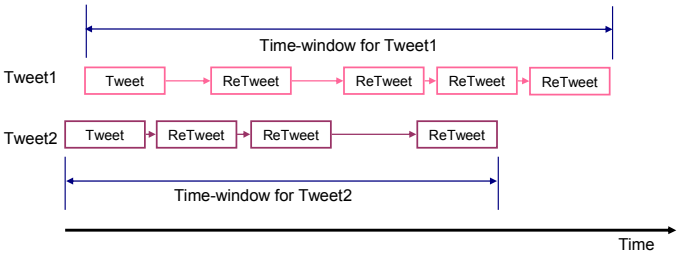

図 6　カスタマイズされた時間ウィンドウ

Figure 6　Customized time-window

ウィンドウ幅を決める単純な手法としては，あらかじめある時間幅を一つ設定して，どのツイートも発信時刻からその時間が経過した時点て，拡散が収束したとみなすことが考えられる．しかしながら，拡散が収束するまでの時間はツイートによって様々であるため，各ツイートの拡散度合いを考慮したほうが望ましい[28].

ツイートの拡散のモデル化はこれまでも研究されてきており，時間経過の拡散の分布は主に対数正規分布に従うとされている[7, 26,27]．図 7 はあるツイートが発信されてからの，RT の発信時間の分布を示している．そこで我々は，ツイートが発信された時刻を基準にして，そこから短時間内の RT の拡散度合いの情報を対象にして対数正規分布にフィッティングする．この分布をもとに，各ツイートの拡散がいつ頃収束するかを推定する．

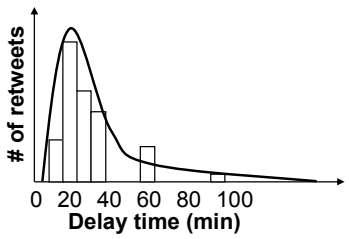

図 7　あるツイートの RT の分布

Figure 7　Retweet distribution of a tweet

具体的な手順は以下のようになる．

対数正規分布の確率密度関数は，パラメータ μ と σ

を用いて以下のように表される．

$$f(x)=\frac{1}{x\sigma\sqrt{2\pi}}e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}},\qquad x>0$$

オリジナルのツイートが発信されてから 1 時間経過した時点での RT の分布に対して，確率密度関数の μ と σ を推定する．得られた確率密度関数の，上側確率の 90%に対応する点を，そのツイートがほぼ収束する時間であるとみなす．これにより，ツイートが RT されなくなるまでを全体とした 90%程度をカバーする時間を推定できると期待する．

本手法では，ツイート発信後 1 時間の RT の情報を基に推定しているが，この時点で総 RT 数が少ない場合はうまくフィッティングできない可能性がある．このように，RT 数がある閾値を下回る場合は，ツイートは，1 時間後の時点でほとんど拡散していないと捉え，既に拡散収束していたと判断する．

5. 実験

Twitter のデータを用いて，拡散分析システムの問合せ性能と，拡散速度を考慮したデータメンテナンス手法の効果について評価する．

5.1. 実験データ

実験データは，期間 2014/1/19 – 2014/2/11 における，2014 年東京都知事選挙の立候補者が発信したツイートを用いる．この期間は，2014 年の都知事選実施期間が含まれている．2013 年より，日本では「インターネット選挙活動」が認められ，候補者が積極的にソーシャルメディアを活用して有権者へアピールするようになってきている．都知事選候補者のうち，Twitter を利用して頻繁に情報を発信していたユーザーA, B, C, D, E を本実験の対象とする．各候補者が発信したツイートの拡散データを実験対象とする．期間中の合計ツイート数は 1611，合計総 RT 数は 76,593 である．

5.2. 実験シナリオと環境

1. 問合せ処理性能

拡散データを対象にして 3.1 節で紹介した Query 1, 2,3 のパターンをインメモリデータベースに問合せ，所要時間を測定する．Query 3 については，3.2 節で述べたランキング前処理のビューを追加した場合の処理時間も測定して比較する．

各回とも，10,000 回問合せた時の 1 問合せあたりの平均処理時間を計算する．(ただし，最初の 1,000 回は平均計算には含めない)

2. 拡散の推定モデルを用いた時間ウィンドウ幅の決定

実験データの候補者ユーザーA が発信したツイートで，RT された数が多い順上位 50 件のツイートの拡散データを対象に，それぞれの拡散の収束時間を推定する．4 章で提案した，ツイートの拡散収束推定手法を用いる．

実験に用いるマシンは 2 x CPU Xeon X5670 (2.93GHz, L1=32KB, L2=256KB, L3=12MB, 6 cores) with 32 GB of RAM, OS は Red Hat Linux 5.5 を使用した．システムは Java (IBM J9 VM JRE 1.7.0)で実装した．データベースは，H2 v1.4.184 をインメモリモードで使用し，RETWEET テーブルと ORIGIN_TWEET テーブルの，オリジナルツイートの ID (TweetID) と，RETWEET テーブルの，オリジナルツイートを発信したユーザー(Src)にインデックスを張った．対数正規分布へのフィッティングは R[25]を用いた．

5.3. 実験結果

5.3.1. 問合せ処理性能評価

図 8 に Query 1, 2, 3 の問合せ処理時間の結果を示す．Query 1 は，WHERE 句で TweetID を 1 件指定した時の結果である．Query 2, 3 は TweetID を重複無しで 100 件指定した時の結果である．TweetID は 1611 件の中からランダムに選択されてる．平均処理時間は Query 1 が 0.2ms，Query 2 が 0.4ms であり，高速に問合せ処理ができている．一方で，Query 3 は 5.6ms であり，前者のクエリと比較して問合せ処理に時間が大幅にかかっている．これは集約演算のオーバーヘッドによるものであると考えられる．

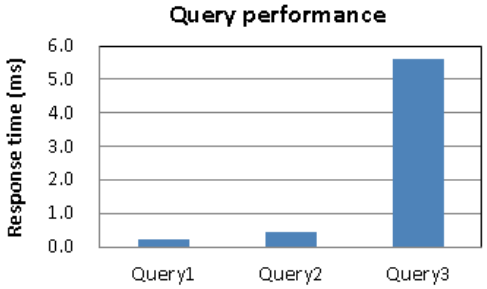

図 8　問合せ性能結果

Figure 8. Query performance results

次に，Query 3 にランキング前処理のビューの導入前後の性能比較結果を図 9 に示す．WHERE 句で指定する TweetID の数を 100 から 500 に変化させたときの問合せ処理時間を測定した．ビューを導入することにより，問合せ処理時間を大幅に短縮できている．

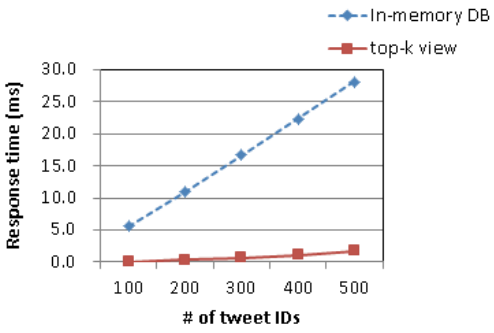

図 9　ビュー導入前後の性能比較

Figure 9. Performance comparison with/without view

5.3.2. メンテナンスのための収束判定

50 件のツイートに対して，ツイート発信時間から 1 時間以内の RT の分布を対象に，4 章の手法を適用する．推定された拡散収束時間の時点で，各ツイートの総 RT 数のうち何%の RT をカバーできていたかを計算する．総 RT 数は各ツイートが発信されてから約 1 週間経過した時点での RT 数を使用する．比較として，各ツイート発信時間から，それぞれ 1, 10, 11, 12 時間経過した時の RT 数のカバー率を測定する．

表 1 に結果を示す．我々の手法は約 90%のカバー率を達成している．

表 1　拡散の平均カバー率

Table 1. The average coverage of retweet diffusion

t=1	t=10	t=11	t=12	**our method**
55.0%	89.1%	89.9%	90.6%	**89.9%**

各ツイートが発信されて 1 時間経過した時点では，平均して総 RT の約 55% までしか到達していないことが分かる．11 時間経過した時点で，手法 2 と同等の約 90% をカバーできている．

各ツイートが実際に 90%の RT 数をカバーした時点の，オリジナルツイートからの経過時間と，本手法で推定した 90%点の時間との差は平均で 368 分であった．一方で，11 時間で固定した時の差は平均で 535 分であった．よって，本手法のほうが高い精度で 90%の点を推定できていることが分かる．

6. 関連研究

情報拡散分析に関する研究はこれまでにも多く存在する．

Truthy[12,13]は，Twitter 上でのアメリカの政党に関連する情報をリアルタイムにトラッキングして，各党が実施するイベントや政策等についての話題を分類したり，情報の拡散を可視化したりするウェブサービスである．Gupta ら[14]は，Twitter で盛り上がったイベントに関するツイートに対して，それらに言及したツイートの信頼度を自動で計算するアルゴリズムを提案し，信頼度順のランキングを表示するシステムを構築した．

TweeQL[15]は，Twitter のストリーミングデータに対して問合せを行うための，SQL-like な問合せ言語である．これにより，キーワードや位置情報，ユーザーID などを指定してストリーミングデータにフィルタリングをかけることができる．ウィンドウ単位での集約処理によるイベント検知等もサポートされている．TwitInfo[16]は，ユーザーが指定したイベントに関する様々な情報をダッシュボードで表示するシステムである．必要な Tweet の情報は前述の TweeQL で収集する．イベントの盛り上がる瞬間を検知したり，ツイートの感情表現を分析して，ユーザーの感想を表示したりする．

これらの関連研究は Twitter 上で情報拡散に関連する分析を行っているが，分析アルゴリズム等をメインにしており，問合せ処理の性能に関して言及しているものではない．我々は，これらのような分析に必要なデータを扱うミドルウェアのシステム構築と，データアクセスの最適化を目指している．

また，top-k 計算に関しては，リレーショナルデータベースでの top-k 計算を高速化するために，あらかじめ materialized view を生成しておき，それを利用する研究がある[17, 18]．また，データベースが更新されたときの効率的なメンテナンス手法についても提案されている[19]．これらのデータ格納形式と本手法は異なるが，効率の良いビューの更新手法等について関連研究を参考にしていきたい．

7. まとめと今後の課題

Twitter のようなリアルタイム性の高いソーシャルメディアでは，今どのような情報がユーザーの間で広く拡散しているのかを知ることが，企業や団体にとって炎上防止や流行把握のために重要である．あるツイートが多数のユーザーに再共有されて広く拡散したメッセージは，多くのユーザーが興味をもち，インパクトを与えた情報であるといえる．

そこで我々は，情報拡散データをストリームデータとして処理してリアルタイムに分析するためのシステムのフレームワークを構築した．リツイートを，オリジナルツイートの ID ごとに拡散収束を判断し，インメモリデータストアをメンテナンスする手法を提案し，評価した．また，Twitter の拡散データを実際にシステム内のインメモリデータストアに格納して，代表的な問合せパターンの処理性能を評価し，高速な処理が出

来ることを示した．

今後は，システム全体の RT の流量を考慮したシステム制御を評価したい．

参 考 文 献

[1] 企業を襲う インターネットの“炎上” http://www.nhk.or.jp/ohayou/marugoto/2014/04/0416.html

[2] ソーシャルメディア運用・分析・監視ツール http://dmc-navi.sendenkaigi.com/keyword/view/3

[3] M. Mathioudakis and N. Koudas, “TwitterMonitor: trend detection over the twitter stream.” In Proceedings of the 2010 international conference on Management of data, pages 1155–1158. ACM, 2010.

[4] S. Asur, B. A. Huberman, G. Szabo, and C. Wang, “Trends in social media - persistence and decay.” In 5th International AAAI Conference on Weblogs and Social Media, 2011.

[5] Lee, C.-H, “Mining spatio-temporal information on microblogging streams using a density-based online clustering method”, Expert Syst. Appl., Vol. 39, No. 10, pp. 9623-9641, 2012.

[6] 那須川 哲哉,西山 莉紗, 金山 博,吉田 一星,大野 正樹, “一人称所有格を用いたプロフィール推定” 言語処理学会 第 19 回年次大会, 2013

[7] 松澤 有,セーヨー サンティ, 鳥海 不二夫, 陳 昱, “リツイート時系列の 3 パラメータ混合対数正規分布による分析”, 人工知能学会全国大会, 2013

[8] HBase http://hbase.apache.org/

[9] Neo4j http://www.neo4j.org/

[10] インメモリー時代の DB 構築 http://coin.nikkeibp.co.jp/coin/sys_ranking10/img/sample3_2.pdf

[11] Ronald Fagin, Amnon Lotem, and Moni Naor. “Optimal aggregation algorithms for middleware. “ In Proceedings of the twentieth ACM SIGMODSIGACT SIGART symposium on Principles of database systems, pp.102–113, 2001.

[12] Ratkiewicz, J., Conover, M. Meiss, M. Gonc,alves, B. Patil, S.; Flammini, A.; nd Menczer, F.. Truthy: Mapping the spread of astroturf in microblog streams. In Proc. 20th Intl. World Wide Web Conf. (WWW) 2011 .

[13] McKelvey K, Menczer F “Truthy: Enabling the study of online social networks.” In: Proc. CSCW 2013.

[14] A. Gupta and P. Kumaraguru, “Credibility ranking of tweets during high impact events,” in Proceedings of the 1st Workshop on Privacy and Security in Online Social Media,. PSOSM, 2012

[15] Marcus, Adam, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. “Processing and Visualizing the Data in Tweets.” ACM SIGMOD Record 40, no. 4, 2012

[16] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. “Twitinfo: aggregating and visualizing microblogs for event exploration.” In Proceedings of the 2011 annual conference on Human factors in computing systems (CHI). ACM, New York, NY, USA, 227- 236.

[17] A. Gupta and I. S. Mumick, editors. Materialized Views: Techniques, Implementations and Applications. MIT Press, June 1999.

[18] V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: “A system for the efficient execution of multi-parametric ranked queries.” In Proc. of the ACM SIGMOD 2001.

[19] K. Yi, H. Yu, J. Yang, G. Xia, and Y. Chen. “Efficient maintenance of materialized top-k views.” In Proceedings of the Nineteenth International Conference on Data Engineering, Bangalore, India, March 2003.

[20] Jain, N., Mishra, S., Srinivasan, A., Gehrke, J., Widom, J., Balakrishnan, H., Cetintemel,U., Cherniack, M., Tibbetts, R., and Zdonik, S., “Towards a streaming SQL standard.” Proc. VLDB, 2008.

[21] インメモリデータベース http://ja.wikipedia.org/wiki/%E3%82%A4%E3%83%B3%E3%83%A1%E3%83%A2%E3%83%AA%E3%83%87%E3%83%BC%E3%82%BF%E3%83%99%E3%83%BC%E3%82%B9

[22] N. Polyzotis, S. Skiadopoulos, P. Vassiliadis, A. Simitsis, and N.-E.Frantzell, “Supporting streaming updates in an active data ware-house,” IEEE International Conference on Data Engineering (ICDE), pp. 476–485, 2007.

[23] Shivnath Babu , Jennifer Widom,”Continuous queries over data streams”, ACM SIGMOD Record, v.30 n.3, September 2001

[24] Arvind Arasu , Shivnath Babu , Jennifer Widom, “The CQL continu-ous query language: semantic foundations and query execution”, The International Journal on Very Large Data Bases (VLDB), v.15 n.2, p.121-142, 2006

[25] R http://www.r-project.org/

[26] Galuba, W., Chakraborty, D., Aberer, K, Despotovic, Z., and Kellerer,W., “Outtweeting the Twitterers – Predicting Information Cascades in Microblogs.”, In 3rd Workshop on Online Social Networks (WOSN), 2010.

[27] Asur, S., Huberman, B. A., Szabo, G., and Wang, C., “Trends in social media: persistence and decay.” In Proceedings of the fifth International AAAI Conference on Weblogs and Social Media (ICWSM), 2011

[28] Haewoon Kwak, Changhyun Lee, Hosung Park, Sue B. Moon, “What is Twitter, a social network or a news media?” pp. 591-600, WWW 2010.