ソーシャルメディアの情報拡散データの リアルタイムストリーム処理性能評価

榎美紀[1,2]　吉田一星[1]　小口正人[2]

概要：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それらが多くのユーザーに再共有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紹介する．従来のストリーム処理とは異なる，ツイートの拡散がはじまって収束するまでをウィンドウ幅として定義する手法を提案する．また，発信されるメッセージ数がシステムで扱いきれないほど急にバーストした場合に，扱うメッセージ数をシステム側で制御することにより，キャパシティをコントロールすることの有効性を評価する．

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

MIKI ENOKI[1,2]　ISSEI YOSHIDA[1]　MASATO OGUCHI[2]

1. はじめに

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，現実社会で起きた災害やイベントについてユーザーが即座に情報を発信したり，逆に，インパクトのある情報がソーシャルメディア上で発生して現実社会に影響を及ぼす現象も多く発生する[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]．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(リツイート， RT)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」という事を発見でき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をリアルタイムに分析するためのシステムを構築している．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．しかしながら，一定のウィンドウ幅でデータを区切った場合，例えばあるツイートが拡散し始めて，それが収束するまでの，拡散データ全体を捉えることが困難になる．逆にウィンドウ幅を長時間に設定した場合，既

1 日本アイ・ビー・エム(株) 東京基礎研究所
2 お茶の水女子大学

に RT されなくなった古い拡散データがシステム内に残り続けてしまう可能性があり，リアルタイム分析システムとして，データの鮮度を保つことができない．したがって，ストリーム処理のウィンドウ幅を，各ツイートが拡散し始めて収束するまでとして定義し，拡散が収束した時点でシステムから退避するメンテナンス手法を導入する．

また，ソーシャルメディア上では，あるトピックが瞬間的に大きく話題になると多くのユーザーが一斉にツイートを発信し，バースト的な状態を起こすことがある．そのような場合，システムは何千何万のツイートを同時に処理することになり，サーバーのキャパシティの限界に達して処理しきれなくなる可能性が生じる．そこで我々は，各ツイートの重要度を計算し，重要度が低いと判断されたツイートをフィルタリングして処理するデータ量をコントロールする手法を提案する．実際にバーストしたツイートデータを用いて，本手法の有効性を確認する．

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紹介する．4 章にて，バースト時のシステムのコントロール手法を提案する．5 章では実際に Twitter のデータを用いて本システムのカスタマイズしたウィンドウ幅を用いたメンテナンス手法とバースト時のコントロール手法の評価を行う．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．

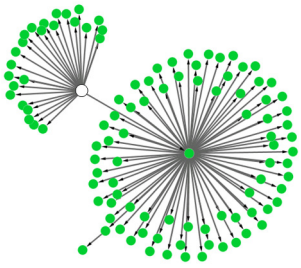

図 1　拡散ネットワーク

Figure 1　Information diffusion network

図 1 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 2 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一時的に格納していく．ツイートの RT の拡散が収束したと判断された時点でデータストアから退避され，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．システムのデータストアには，Key/Value ストアやグラフデータベース等の使用も考えられるが，本研究の代表的なシナリオである，人気のリツイートやユーザーを発見するための集約やソートの演算処理が SQL で記述可能なインメモリデータベースを採用する．

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．

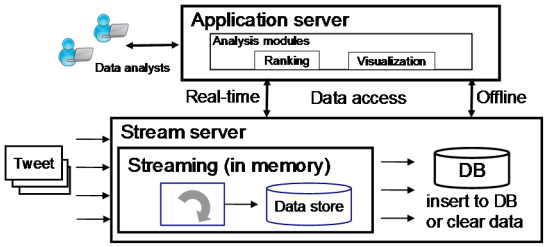

図 2 情報拡散分析システム

Figure 2. Diffusion analysis system.

3. カスタマイズしたウィンドウ幅を用いたインメモリデータストアのメンテナンス

インメモリデータストアに拡散データを格納し，RT の拡散が収束したと判断した時点でデータベースから退避するための，メンテナンス処理の詳細を説明する．

3.1 インメモリデータベースのテーブル構成

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 3). 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 3　拡散ネットワークデータベース

Figure 3　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 情報拡散モデルを用いた拡散収束予測

図 4 はカスタマイズした時間ウィンドウを表している．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．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の全体の拡散データがインメモリデータベースに格納されると期待される．

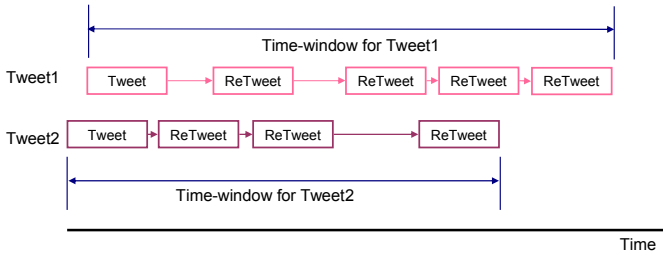

図 4　カスタマイズされた時間ウィンドウ

Figure 4　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[7].

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[8,9,10]. 図 5 はあるツイートが発信されてからの, RT の発信時間の分布の例を示している. 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．

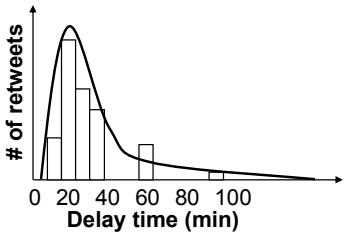

図 5　あるツイートの RT の分布

Figure 5　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. RT 数がある閾値を下回る場合は，1 時間後の時点でほとんど拡散していないと捉え，既に拡散収束したと判断する．

4. データバースト時のキャパシティコントロール手法の提案

ソーシャルメディア上では，あるトピックが瞬間的に大きく話題になると多くのユーザーが一斉にツイートを発信し，バースト的な状態を起こすことがある．たとえば，オリンピックなどのスポーツの試合で盛り上がった瞬間や，大規模な震災が発生した瞬間等である．そのような場合，システムは何千何万のツイートを処理することになり，サーバーのキャパシティの限界に達して処理しきれなくなる可能性が生じる．そこで本章では，各ツイートの重要度を計算し，重要度が低いと判断されたツイートをフィルタリングして処理するデータ量をコントロールする手法を提案する．

図 6 は日本の衆議院議員選挙の開票日(2014 年 12 月 14 日)から翌日にかけての各時間帯の, 政党名を含んだツイー

ト（含リツイート）の発信数を示している．これにより，開票開始時間である 14 日の 20 時に大きくツイート数が跳ね上がっていることが分かる．このように瞬間的なバーストが発生した時，システムを稼動しているサーバーのリソース不足等を引き起こしてしまう可能性がある．

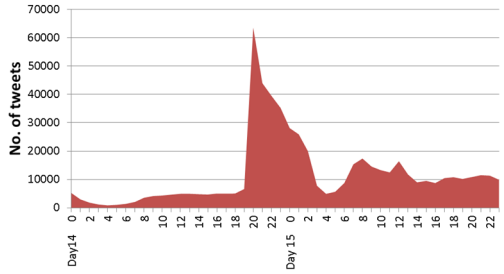

図 6　ツイートのバースト
Figure 6　Bursting of tweets

単純な解決策としては，送信されてくるツイートをランダムにフィルタリングして，処理可能な量のみを扱うようにするという方法が考えられる．しかしながらこの場合，各ツイートのリツイートがランダムフィルタリングされ，図 1 で示したような拡散ネットワークの経路が分断されてしまう．本システムが分析対象とするのは，拡散データであるため，このようにデータが欠損してしまうことは望ましくない．

ツイートには，発信したユーザーの ID，フォロー/フォロワーのユーザー数など，様々な情報が含まれている．そこで，ツイートに付随しているそれらの情報を用いて，ツイートに重要度を表す重み値を付与する．拡散データを分析するにあたり，拡散数が大きいツイートほどインパクトの大きいツイートであり，分析対象とする価値が高いと考えられるため，そのようなツイートの重要度が高くなるようにしたい．したがって，拡散数が大きくなりそうなツイートとそのリツイートのデータはできるだけフィルタリングせずに残すようにする．

以下にツイートの重要度を計算する式を表す．

$$Weight(t) = \begin{cases} getUserInfo(t_{RTID}), & if \quad t \quad is \quad retweet \\ getUserInfo(t_{ID}), & otherwise \end{cases}$$

*Weight(t)*はツイート，もしくはリツイート *t* の重要度を示す重み値である．*getUserInfo* メソッドは，*t* に付随する情報をインプットとして，重み値を返す．本論文では，ユーザーのフォロワーの数を返す．これは，フォロワー数が多いユーザーのツイートは RT される頻度が高いという仮定のもとに設定した．送信されてきたツイートがツイートの場合はそれを発信したユーザーのフォロワー数を，リツイートの場合は，そのオリジナルツイートを発信したユーザーのフォロワー数を参照している．この重み値がある閾値を下回った場合，フィルタリングされる．

この閾値をあげて，多くのツイートをフィルタリングする程，分析結果の精度をさげてしまう可能性が大きくなる．これは，キャパシティのコントロールと分析結果の精度のトレードオフとなる．

5. 実験

Twitter のデータを用いて，拡散速度を考慮したデータメンテナンスとバースト時のキャパシティコントロールの手法の効果を評価する．

5.1 実験シナリオ

(1) カスタマイズしたウィンドウ幅
3.2 節で述べたように，ツイートが発信されてから 1 時間のデータを用いて対数正規分布にフィッティングし，拡散の収束時間を推定する．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評価する．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．

(2) バースト時のツイート数のコントロール
4 章で提案した手法を用いて各ツイートの重み値を計算し，ツイートをフィルタリングする．フィルタリングする重み値の閾値は，1,000 とする．比較対象として，ランダムにツイートをフィルタリングする手法を用いて，フィルタリングの効果と拡散分析結果への精度の影響度を評価する．

実験データは， 2014 年 12 月 の衆議院議員選挙の投票日に，政党名をメッセージに含んでいた日本語ツイートを用いる．該当日のツイート数は図 6 に示す通りである．実験に用いるマシンは 2 x CPU Xeon X5670 (2.93GHz, 6 cores) with RAM 32 GB, OS は Red Hat Linux 5.5 を使用した．対数正規分布へのフィッティングは R[18]を用いた．

5.2 実験結果

5.2.1 カスタマイズしたウィンドウ幅決定の評価

実験データのうち，20 時に発信されたツイートの，総 RT 数上位 100 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 の分布を対象に拡散の収束を推定した．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 数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 2,3,4 時間経過した時(= *t*)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 100 件の結果を平均して 90%のカバー率を達成できていた．これは，それぞれのツイートに対して，拡散データの全体の約 90%をインメモリデータストアに保持することができることを示している．

t=4 の時点で，手法 2 と同等の約 90% をカバーできてい

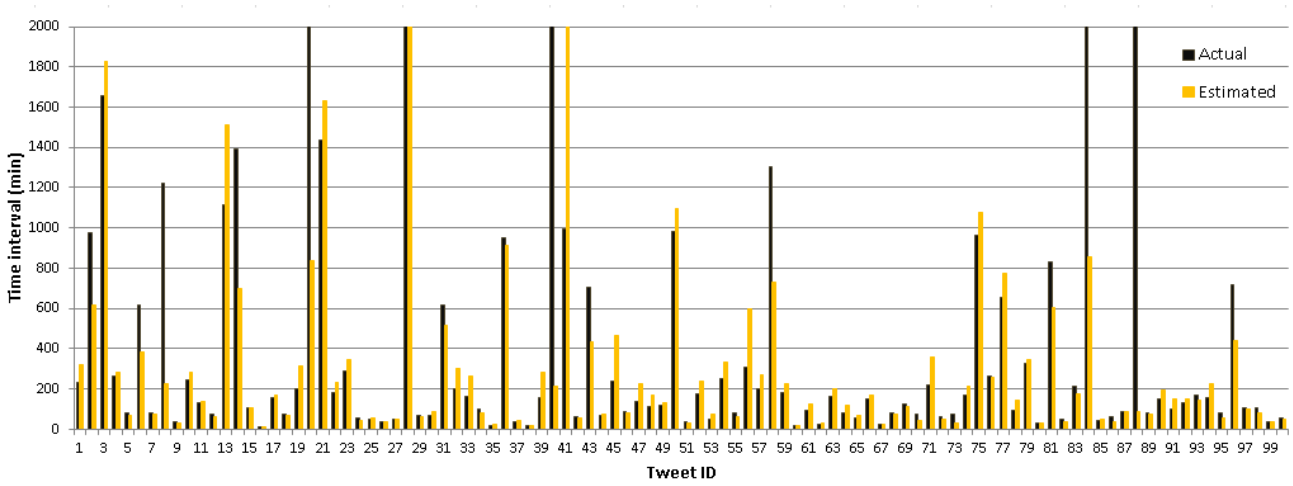

図 7 各ツイートの総 RT 数の 90%の RT 数をカバーした実際の時点と本手法で推定した時点

Figure 7. Actual point of 90% and the estimated point with our method in each tweet.

る．これは，各ツイートが発信されて，一律 4 時間経過するまでのリツイートをインメモリデータストアに保持することを示す．

表 1.拡散の平均カバー率

Table 1. The average coverage of retweet diffusion

t=2	*t*=3	*t*=4	**our method**
82.5%	87.6%	90.4%	**90.2%**

各ツイートが実際に 90%の RT 数をカバーした時点の経過時間と，本手法で推定した 90%点の時間との差は平均で 252 分であった．一方で，一律 4 時間で固定した時の差は平均で 412 分であった．よって，一定のウィンドウ幅で固定するよりも，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．

図 7 は 100 件のツイートそれぞれの，実際に RT 数が総 RT 数の 90%に到達した時点の，オリジナルツイートからの経過時間と，本手法により RT 数が全体の 90%に到達する時間を推定した時点の，オリジナルツイートからの経過時間を表している．我々の手法は，実際の時間と近い結果を取得できていることが分かる．但し，数件のツイートに対しては実際の経過時間が 2000 分をオーバーしており，推定に失敗している．これらのツイートは，数日に渡って数件ずつ RT され続けたためであり，我々の手法はオリジナルツイートが発信されてから 1 時間までのデータを用いて推測するため，推定しきれていない．これらは拡散推定のモデル複雑化する等により改善する可能性があると考えられる．

5.2.2 バースト時のツイート数のコントロール

実験データの開票日である 12 月 14 日の 20 時から 24 時までのツイートをフィルタリングしたツイート数の結果を図 8 に示す．”original”は図 6 の同時間帯のツイート数と同じものを表している．(a)は送信されてくるツイートをランダムに半数フィルタリングした結果であり，(b)は 4 章に示した我々の手法を用いて，重み値が 1,000 未満のツイートをフィルタリングで排除した結果である．本手法により，ランダムで半数をフィルタリングした結果とほぼ同じ数まで削減できている．

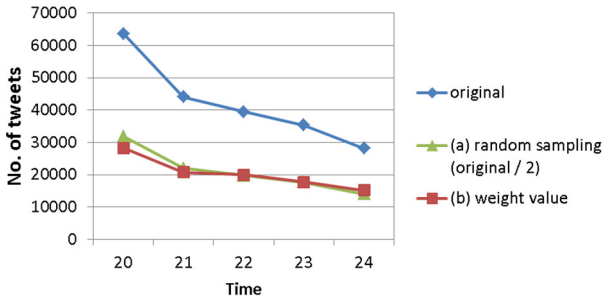

図 8　ツイートのフィルタリング結果

Figure 8. Results of filtering incoming tweets.

次に，フィルタリングによる拡散分析結果への影響について検証する．図 9 の黒いバーは，20 時台に発信されたツイートの，RT 数の多い上位 100 件のランキング結果において，フィルタリング無しのオリジナルのデータを用いた上位 100 件のツイートのうち，フィルタリングした後にランキングを計算したら，何%を 100 位以内に維持できていたかを示している．(a) ，(b)の手法は図 8 と同じである．(a), (b) 両者とも 90%前後のツイートを維持することができている．(a)のように半数にフィルタリングしても，RT 数が多いツイートは相対的にフィルタリングされずに残る可能性が高いため，RT 数のランキング結果に大きな影響は生じていないことは妥当であると考えられる．

白いバーは，同様の RT 数上位 100 ツイートに対して，それぞれの拡散ネットワークを生成した時のエッジ数が，フ

ィルタリング無しのオリジナルデータを使用した時の総数に対して，手法(a), (b)でフィルタリングした時に何%のエッジ数を保持できていたかを示している．手法(a)の場合，エッジ数が約 50%失われてしまっているのに対して，我々の提案する手法(b)では，約 86%のエッジを保持できている．この結果により，我々の手法は拡散の大きなデータの情報をより多く残せていることが分かる．

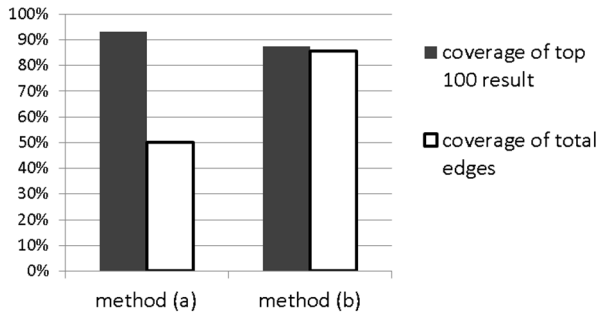

図 9　フィルタリング無しの結果のカバー率

Figure 9. Coverage of original results.

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[11,12]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[13]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[14]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[15]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．

S-Store[16]は，OLTP 処理とストリーム処理を統合したシステムである．SQL をサポートし，ストリーミングデータに対してトランザクションを保証するための研究を行っている．また，Wei ら[17]は，ストリーム処理上のクエリのコストを事前に計算し，ストリームデータがバーストする等で QoS を満たせなくなった場合はデータをサンプリングしてデータ量を削減する．削減率は，クエリコストの情報をもとに決定する．

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，発信されるメッセージ数がシステムで扱いきれないほど急にバーストした場合に，扱うメッセージ数を各ツイートの重要度を基にシステム側でコントロールすることにより，キャパシティをコントロールすることの有効性を示した．

今後は，システムのリソースと，キャパシティコントロールのバランスを考慮して動的にシステム制御できるようにしたい．

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

[8] 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

[9] 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.
[10] 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
[11] 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 .
[12] McKelvey K, Menczer F "Truthy: Enabling the study of online social networks." In: Proc. CSCW 2013.
[13] 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
[14] 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
[15] 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.
[16] Cetintemel, U., Du, J., Kraska, T., Madden, S., Maier, D., Meehan, J. and Zdonik, S, "S-Store: A Streaming NewSQL Sys-tem for Big Velocity Applications." Proceedings of the VLDB Endowment 7.13, 2014.
[17] Wei, Y., Prasad, V., Son, S. H., and Stankovic, J. A. "Prediction-based QoS management for real-time data streams." In 27th IEEE International Real-Time Systems Symposium (RTSS), pp. 344-358, 2006.
[18] R http://www.r-project.org/