リアルタイム型大規模データ処理基盤Jubatusとその活用事例について

韓　正圭，牧野　浩之，熊崎　宏樹

状況や感想などを簡潔に投稿し，他ユーザと意見を交換するマイクロブログは新たな社会分析手段として注目されている．情報が陳腐化する前に大量の投稿から有意義なデータを分析する必要があり，大規模ストリームデータ処理の必要性が近年増している．しかし，バッチ処理と比較すると，標準フレームワークや活用事例の報告が少ないのが現状であり，ストリーム処理システムの普及が難しい理由の一つになっている．本稿ではストリームデータ処理の事例としてTwitter投稿から特定キーワードのバースト出現を検知するシステムをリアルタイム型データ処理基盤であるJubatusを用いて実装した事例を紹介する．

キーワード：ストリーム処理，Jubatus，分散データ処理，マイクロブログ

1. はじめに

マイクロブログはユーザが身の回りの状況や感想などを簡潔な文章で投稿し，他のユーザと共有し意見を交換するウェブのサービスである．代表的なマイクロブログサービスの一つであるTwitter [1] は全世界で約4億6,500万ユーザが加入し，一日に約1億7,500万件の投稿がある（2012年現在）[2]．ユーザ数や，投稿の多さと生の感想を簡潔に投稿できるマイクロブログの特徴から，ユーザの投稿から趣向を分析し社会現象の分析に利用するための研究やサービスが注目を集めている [3–6]．

これらの分析は母集団が多いほうが，統計的により信頼性のある情報抽出が可能になるため，可能な限り多くのデータを対象に分析するケースが多い．しかし，人気サービスのデータは書き込み頻度が高いため，結果として大量のデータを分析する必要がある．また，既存データ分析と比較してマイクロブログ分析で重要視されている要素として，リアルタイム処理が挙げられる．ユーザの即興的感想をほぼ無加工で瞬時に書き込むマイクロブログの性質上，感想が陳腐化する前に意味のあるデータを獲得するためには，素早い分析が要求される．Hadoop [7] のように，大規模データ分散処理のためのオープンソースプログラムは基本的にバッチ処理を前提にしたものである．バッチ間隔を狭めることで，高精度かつある程度素早い分析が可能であるが，データ格納や素早い処理のためのマシン確保のコストが必要であり，一定時間以下の応答時間の確保が難しい．したがって，リアルタイムでは，粗い粒度でデータを処理し巨視的に把握した後，必要な部分のみをバッチ処理を用い，詳しく分析する方法が大規模データ処理では一般的である．また，バッチ処理と比較してリアルタイムデータ処理の事例と運用結果の報告は現時点では少数である．

われわれは現在Twitterへの書き込みから抽出したユーザの意見の変化が現実社会の出来事にどのように影響するかを分析する研究を行っている．例えば，ある特定の映画に対するユーザの評判の変化と劇場入場者など関連要素の変化の関連性を分析し，マーケティングに応用することが挙げられる．当該研究においてはユーザの意見の変化時刻を素早く特定することが重要なポイントとなる．特定キーワードの出現頻度の急激な変化ポイントを検知するキーワードバースト検知手法は，アルゴリズムが比較的にシンプルで，かつほぼリアルタイムに検知が可能であり，また有効な検知精度を持つため，効率的な手段の一つとして複数のイベント検知研究に用いられている [8, 9]．

本稿はTwitterユーザの関心の変化分析研究の一部として構築した，大容量ストリーミングデータからリアルタイムにバーストキーワードを検知するアーキテクチャについて説明し，リアルタイム処理の実際の適用例と効果を示すことを目標とする．本稿の第2章ではバースト検知の関連研究について述べ，第3章でバースト検知のためのシステム構成を，第4章では当該システムの精度，性能評価結果を分析する．

はん　じゅんぎゅ，まきの　ひろゆき，くまざき　ひろき
NTT ソフトウェアイノベーションセンタ
〒180–8585 武蔵野市緑町 3–9-11 NTT 武蔵野研究開発センタ

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

2. 関連研究

テキストストリームにおけるバースト検知に取り組んだ初期の研究として，Kleinberg のバーストモデルがある [10]．このモデルは特定メッセージが観察者に継続的に到達するストリーム環境を想定し，当該環境を異なるメッセージ到達間隔を持つ状態で構成された無限の状態を持つオートマトンとしてモデリングし，環境が一定間隔以下の到達時間を持つ状態にある時をバーストと定義した．また，Shasha らは，例えば，1 時間ごと，1 カ月ごとなどの異なる時間解像度でのバーストを定義・検知するための仕組みとして，固定長の期間で構成される階層型期間を導入して，階層ごとにバーストを検知する手法を提案した [11, 12]．しかし，前述の手法はいずれも与えられたデータを基に，状態遷移の履歴を再構築する必要があり，計算コストが高いため，大規模ストリームデータのリアルタイム処理に適用するのは困難である．大規模ストリームデータに適したバースト検知処理として，Weng らは時間経過とともに変化していくキーワードの出現頻度を波長の周波数変化に見立ててウェーブレット変換を行い，ウェーブレットのエネルギーが一定の閾値を超えた状態をバースト状態と判断するアルゴリズムを提案した [13]．

He らは物理の運動法則をバースト検知に応用し，提案手法がストリームデータのリアルタイム処理に適していることを証明した [14]．一定期間の特定キーワードの出現回数を 2 次元上の位置とみなし，観測した位置変化から速度 v，加速度 a を求め，キーワードの重要性を質量 m として捉える．これらを用い荷重 $f = m \times \mathrm{a}$ を計算し，荷重の絶対値が一定の閾値を超えた期間をバーストが始まった期間とみなす．現実世界のノイズを吸収し，最新の状態を計算結果により反映するため，以下のように EMA (Exponential Moving Average) を用いて，量，速度，加速度を計算している．

EMA:

$$x = \{x_t | t = 0, 1, \ldots, \} \tag{1}$$

$$\begin{aligned} \mathrm{EMA}(n)[x]_t &= \mathrm{a}x_t + (1 - \mathrm{a}) \\ &\quad \mathrm{EMA}(n-1)[x]_{t-1} \\ &= \sum_{k=0}^{n} (\mathrm{a}(1-\mathrm{a})^k x_{t-k}) \\ \text{where } \mathrm{a} &= 2/(n+1) \end{aligned} \tag{2}$$

任意のキーワードに対して考えた場合，一定期間ごとに該当キーワードの出現回数を集計したシーケンスを x とすると，式 (1) のように x は測定開始点 ($t = 0$) から現在までの測定値で構成される．t 期間での窓の長さ n の EMA は式 (2) のよう定義される．つまり最近測定された値ほど重みづけされ，平均に反映する．He らのケースでは EMA 計算時の時刻 t は常に現在の時間を示しているため，これ以後 $[x]_t$ 表記は省略する．

MACD:

$$\mathrm{MACD}(n_1, n_2) = \mathrm{EMA}(n_1) - \mathrm{EMA}(n_2) \tag{3}$$

MACD Histogram:

$$\mathrm{signal}(n_1, n_2, n_3) = \mathrm{EMA}(n_3)[\mathrm{MACD}(n_1, n_2)] \tag{4}$$

$$\begin{aligned} \mathrm{histogram}(n_1, n_2, n_3) = \mathrm{MACD}(n_1, n_2) \\ - \mathrm{signal}(n_1, n_2, n_3) \end{aligned} \tag{5}$$

異なる大きさの窓で測定した EMA の差 MACD を EMA の微分値である速度とし，現在の MACD と MACD の EMA 平均との差分を MACD の微分値である加速度として考えている．Dan らはキーワードの重要性 m を定義した場合の，WEMA と WMACD に関しても述べているが本稿では省略する．Du らは He らの研究を基にキーワードの重要性を，Retweet（伝送），Reply（返信），時間経過による劣化等 Twitter の特徴を考慮して定義した手法を考案し，Twitter データに対して有効であることを示した [15]．しかし，He らの研究手法との精度比較が明らかになっておらず，また，適用した Tweet の規模も限定的であることから，本稿で構築したシステムではバースト検知アルゴリズムとして He らが提案した手法を用い，データソースとして Twitter 社から提供する Gardenhose ストリーム API を用いた [16]．

3. システム概要

本研究で用いたバースト検知システムの構成を図 1 に示す．システムの設計にあたって拡張性 (Scalability) と可用性 (Availability) を最も考慮した．

はじめにデータ生成器 (Data Generator) を用いて，バースト検知のためのデータストリームを提供する．データ生成器はファイル形式にローカルに格納されたデータを読み込み，あらかじめ設定された流量でデータストリームを流し続ける疑似データ生成機能と，Twitter から提供するストリーム API を用いて収集したデータをリアルタイムに発行するデータ収集機能がある．本研究ではデータソースとして，Twitter 社から提供される Gardenhose ストリーム API を用いて収集した日本語ツイートを連続したストリームとして出

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

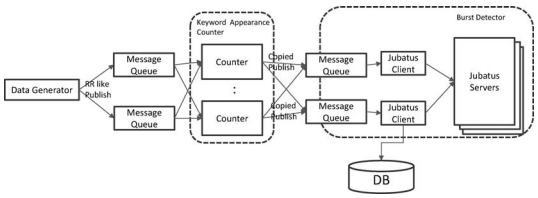


図 1　バースト検知システム構成

力する．

ツイートの本文は集計器 (Counter) により単語単位に分析され，一定期間ごとの出現頻度を単語ごとに集計する．具体的には図 2 で示すようにおのおののツイート本文を，形態素解析器を用いて形態素に分解し，投稿された期間ごとに形態素集合を作成する．形態素毎に必要な単語のみをフィルタリングし，残った単語おのおのに対して出現頻度を集計して出力する．形態素解析プログラムとしてオープンソースソフトウェアである MeCab [17] を利用した．集計器の処理は状態を保持しない処理であるため，拡張性のため複数配置を可能にした．また，データ生成器と集計器の処理速度の差を吸収するため，集計器の前にメッセージキューを配置し，データの受け渡しをメッセージキュー経由で行う．可用性を確保するためキューは 2 個以上を配置し，個々のツイートデータはラウンドロビンに近い方法で，データ生成器からおのおののメッセージキューへ伝送され，集計器はすべてのキューからデータを取得する．

バーストキーワードの判定モジュールには，オープンソースソフトウェアである Jubatus [18] を用いた．Jubatus はリアルタイム処理タスクが精度と処理性能のトレードオフ関係にあることに注目し，機械学習において，ある程度の学習モデルの精度を保ちつつ処理性能を優先させる学習モデル共有方式を実装したオンライン機械学習フレームワークである．現時点（2012 年 8 月）では，フレームワーク上で機械学習手法である分類，回帰，統計，推薦，グラフマイニングが提供されている．本実験では，このフレームワークを応用し，EMA 関連アルゴリズムを実装した．Jubatus を用いた理由は，ストリーム処理は大規模バッチ処理と異なり Hadoop などの事実上の標準がまだないうえ，ストリーム機械学習のフレームワークを提供しているため，構築期間の短縮と次の研究である機械学習による分析への拡張性を考慮したためであった．図 3 にバースト検知器のワークフローを詳細に示す．集計器から作成された集計結果はバースト検知用のキューを介し

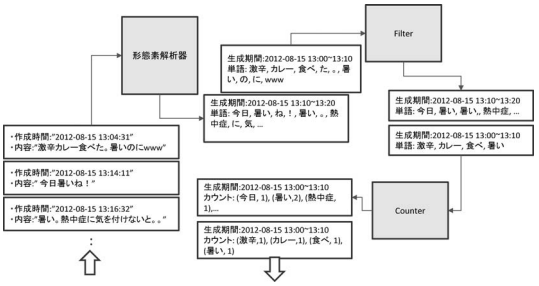


図 2　集計器ワークフロー

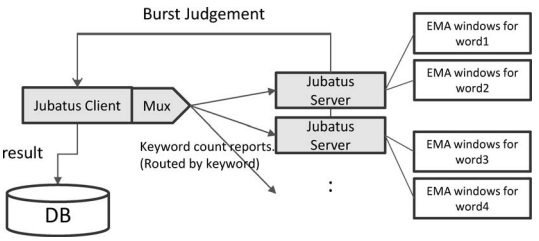


図 3　バースト検知器ワークフロー

て Jubatus クライアントへ伝送される．Jubatus クライアントはそれぞれのキーワードを担当する Jubatus サーバへの振り分け処理を行い，サーバの処理結果を用い該当キーワードのバーストを判定し，外部データベースに記録する．Jubatus サーバはキーワードに対しての荷重計算を担当しており，そのために必要となるキーワードごとの EMA 計算窓をメモリ上で管理し，新たな集計情報を得るたびに新しい荷重を計算し，クライアントに通知する．チューニングの結果，集計器は 10 分毎に集計し，式 (5) の n_1, n_2, n_3 はそれぞれ 7, 9, 5 とした．

4. 評価

各構成モジュールの性能および拡張性とシステム全体の長期安定性とバースト検知精度を評価した．表 1 に示す構成のマシン複数台を評価に用いた．

表 1　測定マシン詳細

項目	詳細
CPU	Intel Xeon X3430 2.4 GHz (4 core)
Memory	8 GB
Network	Gigabit Ethernet
HDD	500 GB
OS	CentOS 5.5

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

モジュール別性能

バースト検知システムはおのおののモジュールが担当タスクを処理し，結果を次のタスクの入力にする連鎖処理であるため，最も処理性能が低いモジュールがシステム全体の性能を決定する．本システムでは，データ生成器，メッセージキュー，集計器，バースト検知器が該当するが，本稿ではシステムのコアモジュールである集計器とバースト検知器の性能に関して評価する．個別モジュールの性能限界とは当該モジュール前段から入ってくるストリームを滞りなく処理する（キューに滞留が発生しない）ことである．したがって，集計器の性能測定のため 3 台のマシンを準備し，うち 1 台をデータ生成器，残り 2 台にはそれぞれメッセージキューと集計器を配置し，生成器のデータ生成速度を調整しながら 30 分間連続運転し，キューが滞留するデータ生成速度を調査した．測定の結果，2 台合わせて秒間 3,000 件（1 台の場合，1,500 件）の日本語ツイートを処理可能であった．1 日に約 1 億 7,500 万件のツイートが投稿されており [2]，これは秒に換算すると約 2,000 ツイートであるため，2 台で平均的な全日本語ツイートをリアルタイムに処理することが可能である．また，状態を保持しない処理であるため，台数を追加することにより，性能はほぼ線形にスケールする．バースト検知器は一定期間毎に送られてくる集計結果を用いて個別の単語に対して式 (5) のヒストグラムを計算し，単語が異常出現状態かどうかを判定する．したがって，次の集計結果が送られてくる前に，すでに受け取った結果に対して判定処理を終了することが，バースト検知器が最低限保証すべき性能となる．また，バースト検知対象は個別単語ごとであるため，計算する単語数により限界性能は異なる．

バースト検知器の性能測定のため 2 台のマシンを準備し，うち 1 台にはデータ集計器とバースト時刻記録用データベース，もう 1 台にバースト検知器を配置し，計算すべき単語数を増加させながら 12 期間分の集計結果（1 期間 10 分，合計 2 時間分のデータ）の処理を実施し，おのおのの処理に費やした時間を測定，平均値を求めた．結果を図 4 に示す．

監視単語数が 5,000 個の場合，約 3 秒で処理可能であった．10,000 個と 25,000 個の場合それぞれ約 6 秒，約 16 秒と処理時間は監視単語数に比例している．ただし，監視単語数が 50,000 個の場合，約 24 秒と比例した場合よりも低い値になっている．これは入力データとして実際のツイートデータを用いたためと考えられる．用いたツイートデータは，10 分間に約 40,000 単語を含んでいた．したがって，50,000 単語を集計しようとした場合でも実際は約 40,000 個が集計される．このため実際は約 40,000 個の単語に対する集計結果が伝送され，この伝送時間がバースト検知器の性能の支配項であると考えられる．また，日本人成人は約 50,000 単語の語彙力があると考えられており [19]，一般的にイベントを識別可能な単語は認知度の高い固有名詞と特定名詞群であるため，冗長化を考慮しない場合であれば 1 台のマシンでも Twitter のリアルタイムなバースト検知が可能であると考えられる．

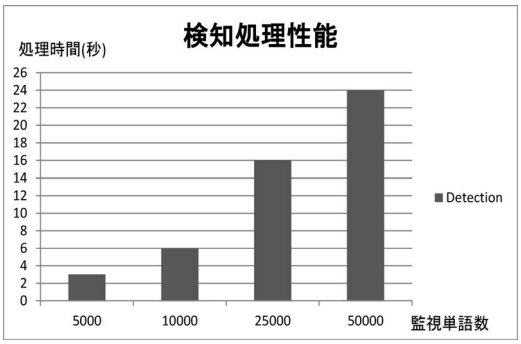


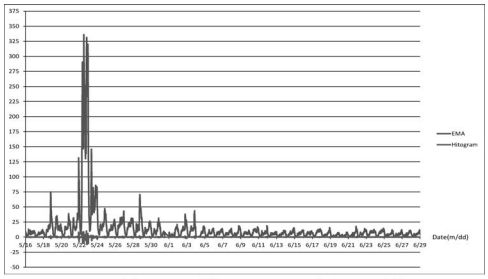
図 4　バースト検知処理性能

試験運転

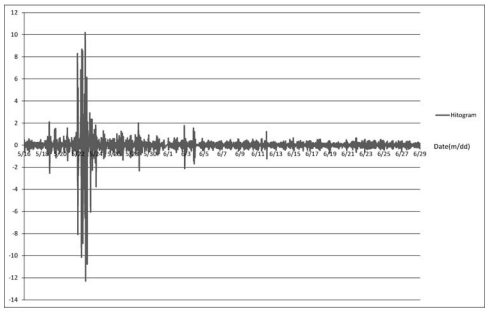
試験運転として，2012 年 5 月中旬から 6 月末まで約 1 カ月半 Twitter の Gardenhose ストリーミング API を用いて収集したストリームデータをリアルタイムに処理した．システムはデータ生成器 1 台，メッセージキューと集計器ペア 2 台，メッセージキューとバースト検知器ペア 2 台と，結果記録用データベース 1 台の計 6 台構成とした．また，Gardenhose はツィートの流量の 1/10 を獲得可能としているため，全日本語ツィート処理をシミュレートするためにデータ収集器から Gardenhose 流量の 10 倍の疑似ストリームを生成し，1 週間動作させた．試験期間中システムは安定動作した．試験期間中いくつかのキーワードのバーストを検知した．例として一部の頻度集計グラフと MACD ヒストグラムを図 5 と 6 に示した．図 5 によると 5 月 22 日にキーワード“スカイツリー”がバーストしていることがわかる．

実際に該当日はスカイツリーのオープン日であった．また，6 月 24 日の深夜にはキーワード“ダークナイト”のバーストを検知している．図 6 の (b) は 6 月 24 日時から 25 日 24 時までの式 (5) のヒストグラムである．21 時，23 時，翌日 1 時と 3 回のバーストが起きていることがわかる．実際，該当日の 21 時から 23 時頃

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

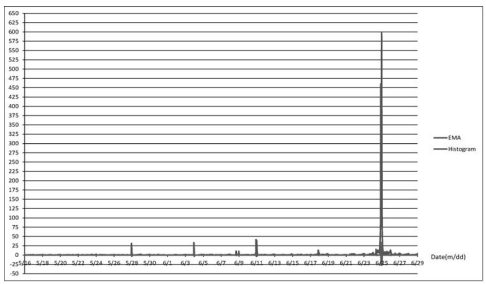


(a)　EMA(7), Histogram(0 付近の線)

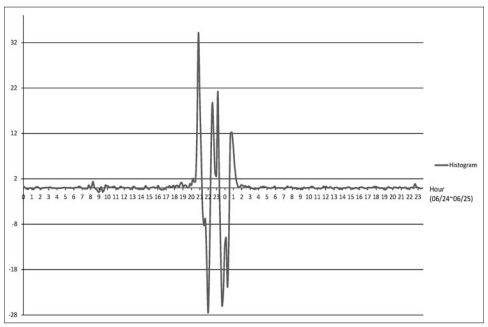


(b)Histogram

図 5　スカイツリー



(a)EMA(7), Histogram(0 付近の線)



(b)Histogram

図 6　ダークナイト

まで該当の映画が TV で放映されていた．表 2 はバースト付近で投稿されたキーワード“ダークナイト”を含むツイート群を検索し，戸田ら [20] が提案した手法を用い代表的なツイートを抽出した結果を一部引用したものである．1 回目の 21 時付近のバーストは，TV

表 2　代表ツイート

期間	ツイート
6/24 20:50 ～ 21:20	solt_yamori，“ダークナイトなうう．ゲイリーwktk”，2012-06-24 21:03:46 MASA031008，“ダークナイト，開幕カットした?”，2012-06-24 21:03:48 nktkskr，“ダークナイトってなんですか”，2012-06-24 21:03:48 marodotto，“ダークナイト見るよ !! ”，2012-06-24 21:03:49 sasuke3k，“ダークナイト撮るの忘れてた死んだ”，2012-06-24 21:03:51 hir0k0，“うおおおお銃社会すげえええええ#ダークナイト”，2012-06-24 21:03:52 redbicycle，“ダークナイトみてる．人質がいる中でシャッガン使うかフツー”，2012-06-24 21:03:54
6/24 22:40 ～ 23:20	viroosyana，“RT @nunonofuku123: 実のとこ，ダークナイトの制作費がバカ高くなったのはぶっ壊した 1000 万の IMAX 映写機とぶっ壊したこの本物のランボルギーニ．#tvasahi #ダークナイト”，2012-06-24 22:45:37 vino7，“RT @ryohgo_narita: ダークナイト豆知識．この病院爆破，ＣＧじゃない．爆発が起こらなくてジョーカーが振り返ったりしてるのは，爆破のタイミングが遅れたというトラブルに対するヒースのアドリブ．ちなみに NG だったらもう一回病院建てて爆破するつもりだったとかなんと …”，2012-06-24 22:46:19
6/25 00:40 ～ 01:00	rikusiki，“ダークナイトは最後の船の下りがどうしても納得いかない．”，2012-06-25 00:40:03 AyhyO，“ジョニーデップにはまり映画全般にはまってた頃にダークナイト上映してて，すごく見たかったの思い出した．ヒースさんもっと色んな作品に出てほしかったな 残念”，2012-06-25 00:44:29 usadog，“ダークナイトも見れなかったし作業もできてないし，さいやく．何時間寝てん．”，2012-06-25 00:44:32 redmu10，“RT @nozu8: ダークナイトの地下室とF/Z にでてきた地下室 気になったから比較してみた http://t.co/Qmos58UG”，2012-06-25 00:44:52

で映画の放映が始まったため，2 回目の 23 時付近のバーストは映画のクライマックスのシーンで，3 回目の翌日 1 時付近のバーストは映画の感想について盛り上がっている．

5.　おわりに

本稿では，大容量ストリーミングデータからリアルタイムにバーストキーワードを検知するアーキテクチャについて検討し，リアルタイム処理の実際の適用例と効果を検証した．バースト検知アルゴリズムを Jubatus フレームワーク上に実装することにより，スケーラブルでリアルタイムなバーストの検知が可能となった．今後の課題として，より高度なバースト検知を実現するため，バースト同士の関連性検出や動的な集計窓の変更への対応，などの機能拡大が挙げられる．

参考文献

[1] Twitter, http://twitter.com
[2] Twitter2012, http://www.blogherald.com/2012/02/22/twitter-2012-infographic/
[3] B. J. Jansen, M. Zhang, K. Sobel, and A. Chow-

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

dury, “Twitter Power: Tweets as Electronic Word of Mouth,” *Journal of the American Society for Information Science and Technology*, **60** (11), 2169–2188, 2009.
[4] M. Naaman, H. Becker and L. Gravano, “Hip and Trendy: Characterizing Emerging Trends on Twitter,” *Journal of the American Society for Information Science and Technology*, **62** (5), 902–918, 2011.
[5] Sproutsocial, http://sproutsocial.com/
[6] クチコミ@係長, http://www.hottolink.co.jp/kakaricho
[7] Hadoop, http://hadoop.apache.org
[8] M. A. Cameron, R. Power, B. Robinson and J. Yin, “Emergency Situation Awareness from Twitter for Crisis Management,” *WWW'12 Companio*n, 695–698, 2012.
[9] J. Yao, B. Cui, Y. Huang and X. Jin, “Temporal and Social Context Based Burst Detection from Folksonomies,” *Proc. of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.
[10] J. M. Kleinberg, “Bursty and Hierarchical Structure in Streams,” *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD02*), 90–101, 2002.
[11] X. Zhang and D. Shasha, “Better Burst Detection,” *Proc. of the 22nd International Conference on Data Engineering* (*ICDE '06*), 2006.
[12] Y. Zhu and D. Shasha,“Efficient Elastic Burst Detection in Data Streams,” *Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and data mining* (*KDD 03*), 2003.
[13] J. Weng, Y. Yao, E. Leonardi and F. Lee, “Event Detection in Twitter,” http://www.hpl.hp.com/techreports/2011/HPL-2011-98.html
[14] D. He and D. S. Parker, “Topic Dynamics: An Alternative Model of ‘Bursts’ in Streams of Topics,” *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD10*), 2010.
[15] Y. Du, Y. He, Y, Tian, Q. Chen, and L. Lin, “Microblog Bursty Topic Detection Based on User Relationship,” *Information Technology and Artificial Intelligence Conference* (*ITAIC*), 2011
[16] Gardenhose API, https://dev.twitter.com/docs/streaming-apis/streams/public
[17] MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html
[18] Jubatus: Distributed Online Machine Learning Framework, http://jubat.us/
[19] 森岡健二, “義務教育終了者に対する語彙調査の試み,” 国立国語研究所年報 2, 95–107, 1951.
[20] 戸田浩之, 北川博之, 藤村 考, 片岡良治, 奥 雅博, “グラフ分析を利用した文書集合からの話題構造マイニング,” 電子情報通信学会論文誌, J90-D (2), 292–310, 2007.

 Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.