ソーシャルビューイングにおけるトピックを考慮したツイート集約化手法

大田垣　翔†　　角谷　和俊††　　牛尼　剛聡†††

† 九州大学大学院芸術工学府　〒 815-8540 福岡県福岡市南区塩原 4-9-1
†† 関西学院大学総合政策学部メディア情報学科　〒 669-1337 兵庫県三田市学園二丁目一番地
††† 九州大学大学院芸術工学研究院　〒 815-8540 福岡県福岡市南区塩原 4-9-1
E-mail: †2DS14087T@s.kyushu-u.ac.jp, ††sumiya@kwansei.jp, †††ushiama@design.kyushu-u.ac.jp

あらまし　近年，Twitter 等の SNS を利用して視聴中の TV 番組の感想を投稿し，他者の投稿を読みながら TV 番組を観る「ソーシャルビューイング」という新たな視聴形態が注目されている．Twitter においては，ハッシュタグを使って目的の番組に関するツイートを収集し行われ,「TV 実況」と呼ばれ親しまれている．実況タイムラインでは，複数のユーザが様々な番組でソーシャルビューイングを行っているが，短時間に大量のツイートが投稿されるため，それら全てを読むことは困難である．この問題を解決するために，本研究では，実況タイムラインを適切に集約化してユーザに提示することによりユーザが快適にソーシャルビューイングを楽しめるようにすることを目的とする．本論文では，実況タイムラインから盛り上がっているトピックを検出し，トピックを伝える代表ツイートのみをユーザに提示することで，ユーザに盛り上がりを効果的に提示可能な集約化手法を提案し，プロトタイプシステムを利用した被験者実験により，有効性を評価する．

キーワード　SNS, ソーシャルビューイング, Twitter, クラスタリング, 要約, bigram

1. はじめに

近年，ソーシャルネットワークサービス (SNS) が世界的に普及した．現在では様々な SNS が利用されており，Twitter [1] は世界中で多くのユーザを持つ代表的な SNS の一つである．Twitter の投稿記事はツイートと呼ばれ 140 文字以内の短文であり，気軽に投稿できるために．ユーザが見たり感じたりしたリアルタイムな情報が投稿される事が多い．こうした特徴から Twitter は現在コミュニケーションや情報収集など，様々な目的で利用されている．

Twitter にはハッシュタグという機能が存在する．ユーザはツイートにハッシュタグを付与することで，投稿の分類を行うことができる．イベント等の際はユーザ間でハッシュタグを決めて投稿する事で，イベント関連ツイートをユーザ間で共有することができる．

また，SNS の登場により新たな TV の視聴形態である「ソーシャルビューイング」が登場した．ソーシャルビューイングとは，SNS を利用し同じ番組を視聴しているユーザと感想を共有しながら TV 番組を視聴するものである．これによりユーザはパブリックビューイングと同様に，同じ嗜好を持つ者と，盛り上がりを共有しながら TV 番組の視聴という一つのイベントを体験する事ができる．Twitter においてソーシャルビューイングを行う場合は，ユーザは番組毎に特定のハッシュタグツイートに付与し投稿を行い，またハッシュタグを検索することで同様の番組を視聴しているユーザの投稿を収集し行う．Twitter におけるソーシャルビューイングは「TV 実況」と呼ばれ，そのツイートを「実況ツイート」，取得されたツイートのストリームは「実況タイムライン」と呼ばれている．しかし，対象とする TV 番組が多くのユーザに注目されていると，取得される実況ツイートの量が多くなってしまい，視聴中にタイムラインを閲覧するユーザはツイートの全てを読むことが困難になる．

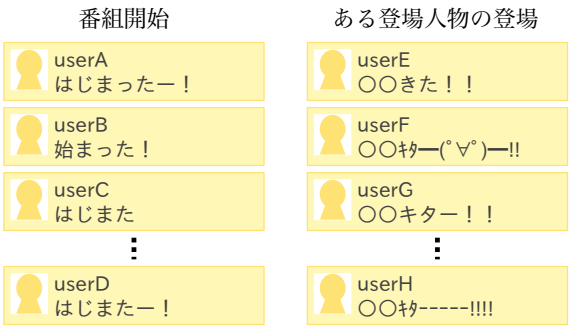

図 1　盛り上がっているツイート例

この問題を解決するために，本研究では，実況タイムラインにおいて同一事象に対するツイートが，複数のユーザによって集中的に投稿される現象 (図 1) に着目する．この現象を，同一トピックに対しての「盛り上がり」と呼ぶ．本論文では，盛り上がりの発生をリアルタイムに検出して，トピック毎に要約することでタイムラインの内容を集約する手法を提案する．本手法では実況タイムラインに対して一定期間毎にツイートの収集解析を行う．期間中の全てのツイートに含まれる bigram の出現頻度を利用して，盛り上がりを検出する．その後，同一トピックに含まれやすい bigram 同士は期間中の出現頻度の時間推移のパターンが類似する点に着目し，トピック毎に，それを表す高頻出 bigram の集合を得る．その時の実況ライムラインから，トピック毎に，それを表す bigram を最も純度高く含ん

でいるツイートを代表として 1 つ選び，トピックの要約とする．以上の手順で，ユーザに提示する為の実況タイムラインの要約をリアルタイムに生成する．そして，提案手法の有効性を被験者実験により評価する．

2. 関 連 研 究

近年，Twitter を用いたソーシャルビューイングが注目されるに従い様々なサービスが登場している．SONY 社の液晶 TV「BRAVIA」は，Twitter 連携 [3] を提供し，ユーザは TV 画面上で番組と Twitter を同時に閲覧することができる．また PC などのブラウザから利用可能な Livetter.com [4]，スマートフォン・PC 両環境で利用できる「つぶあに」[5] 等の SNS において実況投稿を気軽に行え，また閲覧できるアプリケーションや Web サービスが提供されている．これらによりソーシャルビューイングの利用が容易になっているが，どのサービスの機能も実況タイムラインをそのまま表示するものであり，ツイートの大量取得によるユーザの負担に配慮した機能は実装されていない．

SNS の世界的な普及に伴い，ソーシャルビューイングに着目した研究も多くある．中澤ら [6] は TV 番組の放送に合わせてリアルタイムに行われるユーザの実況ツイートは番組の放送内容と関連性が高いと考え，TV 番組に関するツイート数の変動から重要なシーンを検出し，ツイート内容から各シーンのイベント内容を表すラベルを生成する手法を提案している．しかしこの手法は，放送終了後の番組に適応することを前提としているためリアルタイム性は考慮されていない．

またストリームからツイート数の増大期間に着目し，リアルタイムに要約の生成を行っている研究に，久保ら [7] と坂本ら [8] の研究がある．久保らは増大期間のイベントを最も適切に表現しているユーザのツイートを，事前に用意した説明性の高い単語を登録した辞書を基に見つけ出し，期間の要約として速報という形でユーザに提示する手法を提案している．この手法はリアルタイム性を考慮しているが，事前に用意した辞書の適用範囲のみにしか有効ではない．坂本らはツイート数の増大期間に対して，ユーザの投稿速度が一定でないために過去の増大期間のイベントの情報が現在のイベント情報にも混在してる可能性を考慮した，期間の適切な重要語群を要約として生成する手法を提案している．しかし，これらの研究はツイート数の増大に着目しており，ツイートの内容の盛り上がりの検出とその要約が目的である本研究とは異なる．

これらを踏まえ，本研究ではリアルタイム性を考慮した，ストリームから盛り上がっているトピックを検出，要約を目的とし，且つ対象となる番組を限定しない手法の開発を目指す．

3. 実況タイムラインの特徴

実況タイムラインには，不特定多数のユーザによって TV 番組内の出来事に対しての感想がリアルタイムに投稿されている．そのような実況タイムラインでは，複数のユーザが同一の事物に対しての投稿をし，それが短期間に集中することがある．それは番組内のシーンセリフに対しての言及であったり，番組の開始終了 CM 等々についてであったり様々である．これを本研究では実況タイムライン上における，あるトピックに対する盛り上がりと定義する．

また，複数のトピックの盛り上がりが短期間に集中する状況もある．図 2 にそのような状況の具体例を示す．これは TV 番組「ガンダムビルドファイターズトライ」(2015 年 2 月 4 日 18:00〜18:30 放送分) の冒頭 1 分間の実況タイムラインのツイートを 5 秒間毎に集計したものである．更にその中から定義によるトピック毎の盛り上がりで主要なものを人手で発見し，各トピックに分類したツイートの記事数を同じく 5 秒間毎にプロットしてある．期間中の実況タイムラインでは開始 15 秒間に置いて番組開始ついて言及するツイートが増え，「はじまた」等のフレーズが多く見られた．また開始から 20 秒後から番組内の状況についてのツイートが増え，「修羅場」のキーワードを含むツイートが多く見られた．それぞれトピック「はじまた」，「修羅場」の盛り上がりとしている．

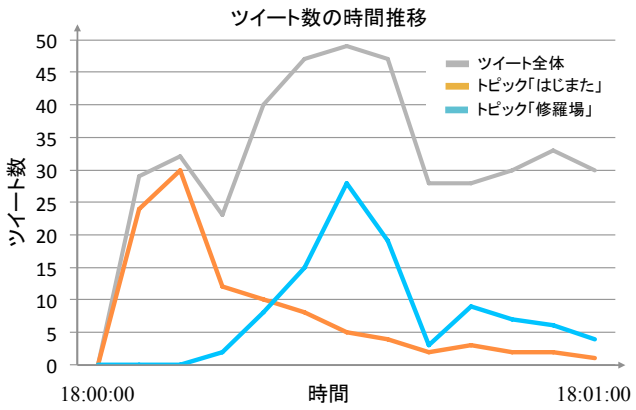

図 2　TV 番組「ガンダムビルドファイターズトライ」2015 年 2 月 4 日放送の番組開始 1 分間のツイートの 5 秒間ごとの時間推移グラフ

本研究では関連研究のようなツイート数の増大期間ではなく，このトピック毎の盛り上がりの発生期間をリアルタイムに検出する．また盛り上がりが短期間に集中した場合でもトピック毎に検出可能な手法を提案する．

4. 提 案 手 法

本研究では大量のツイートが流れるタイムラインを読むユーザの負担軽減を目的とする．そのために 3. で述べた「盛り上がり」に着目し，リアルタイムに「盛り上がり」を検出，要約しユーザに提示する実況タイムラインの集約手法を提案する (図 3)．本手法は一定期間ごとに Twitter からハッシュタグを利用し番組の実況ツイートを収集し，以下大別して 2 段階の処理を実行する．

a. 収集ツイートからのトピック別の盛り上がりの検出
b. 盛り上がり別の要約の生成

4.1 トピック別の盛り上がり検出

本研究では実況タイムラインにおいて盛り上がっているトピックを，リアルタイムに発見する．そのため実況タイムライ

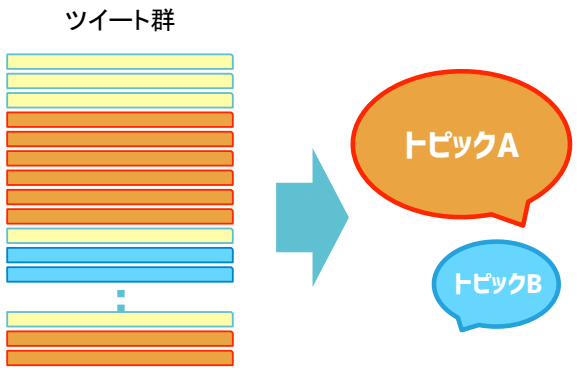

図 3　提案手法の概要

ン上のトピック抽出とトピックの変遷を把握する必要がある．

従来，ドキュメント集合からトピック抽出を行うには TFIDF 法による重要単語の抽出や潜在的ディリクレ配分法（LDA: Latent Dirichlet Allocation）によるトピック抽出が行われてきた．これらを利用し特定期間毎に生成されたトピックの追跡を行いトピックの変遷を把握する研究は，TFIDF 法を利用した手法では水落ら [9]，LDA を利用した手法では芹澤ら [10] の研究等がある．しかし，これらの手法では特徴語となる単語を抽出する必要があり，そのために MeCab などを利用した形態素解析を行う必要がある．しかし，Twitter におけるツイートは表記ゆれが多い，加えてソーシャルビューイングにおけるツイートでは，キャラクター名や作品内造語等々の辞書に無い単語が多く存在する可能性が高い為，適用することが難しい．

本手法では上記の理由から，文字 bigram を用いたトピックの盛り上がり抽出とその変遷の把握を行う．そのために対象時間区間ごとに以下の 2 つの処理を行う．

a. 対象時間区間中の頻出度の高い bigram を，DBScan を利用してトピック毎にクラスタリングする

b. コサイン類似度を利用して，複数の時間区間をまたぐ同一トピックの盛り上がりを示す bigram クラスタを追跡する

4.1.1　対象とする時間区間中の bigram のクラスタリング

対象とする時間区間中の実況タイムラインからトピック別に盛り上がりを発見する為に，本手法では時間区間中のツイートに bigram の出現頻度の時間変化に着目する．実況タイムラインにおいて，あるトピックについての盛り上がりが発生している場合，そこには集中的に使用されているフレーズやキーワードが存在する．その期間の bigram の出現頻度の時間推移を 1 秒間毎に集計すると図 4 の様になる．これは図 2 で用いた TV 番組「ガンダムビルドファイターズトライ」の期間中の実況タイムラインのツイートに含まれていた bigram の 1 秒間毎の出現頻度を集計し，過去 5 秒間のデータを用いた移動平均で平滑化したものである．トピック「はじまた」の盛り上がりでは bigram「はじ」「じま」「また」の推移波形が大きく変化しており，トピック「修羅場」でも同様であるが，こちらは 2 つの bigram の波形が完全に一致している．このように盛り上がりが発生するとその期間は，特定の bigram の出現頻度が上昇し，フレーズやキーワードに含まれやすい bigram は類似した推移波形をとる．これを利用し，対象時間区間中の実況タイムラインからトピックの盛り上がりの検出を行う．

n 番目の対象時間区間中のツイートを $T_n = \{t_n^1, t_n^2, ..., t_n^k\}$ する．また T_n に存在した Bigram を $B_n = \{b_n^1, b_n^2, ..., b_n^l\}$ とする．それぞれの bigram の n 番目の時間区間中 の出現頻度を $freq_n(b_n^l)$ とする．まず期間中にトピックの盛り上がりが発生しているかどうかを検出するために，出現頻度が閾値 $Amin$ 以上の bigram を得る．

$$B'_n = \left\{ x | x \in B_n^l, freq(b_n^l) \geqq Amin \right\} \qquad (1)$$

$|B'_n| > 0$ であった場合に盛り上がりが発生していると判定する．それらから推移パターンが類似している bigram のクラスタを発見し，トピックを表す bigram の集合とする．bigram の系列データ $x[k], y[k](k = 1, 2, ..., n)$ があるとして，推移パターンの類似性を，ピアソンの相関係数を利用して下記の式 (2) のように定める．

$$distance = 1 - Pearson(x, y) \qquad (2)$$

式 (2) を利用し，高頻出の bigram 集合 B'_n を DBScan [11] を用いてクラスタリングを行う．これにより，一つの要素が盛り上がっているトピックを表す bigram のクラスタである，クラスタ集合 $C_n = \{c_n^1, c_n^2, ..., c_n^h\}$ を得る．通常の DBScan では条件から外れたノードを，クラスタに含めない外れ値 Border Point として扱う．しかし，実況タイムラインでは 2 文字のフレーズで盛り上がりが発生することもあるため為，本手法では Border Point として判定された bigram も単独でトピックを表す bigram のクラスタとして扱う．DBScan における到達可能半径 Eps 及び最低密度 $MinPts$ の最適値に関しては後の章で検証を行う．

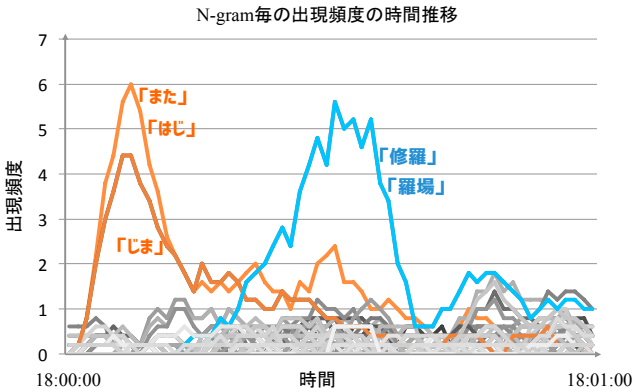

図 4　TV 番組「ガンダムビルドファイターズトライ」(2015 年 2 月 4 日 18:00〜18:30 放送) の番組開始 1 分間の実況タイムライン上のツイートに含まれる bigram について，1 秒間毎にその出現頻度を集計し，過去 5 秒間の移動平均をとり平滑化した時間推移グラフ

4.1.2 時間区間をまたぐ盛り上がりの追跡

提案手法 4.1.1 で，n 番目の時間区間における 実況タイムラインから，トピックの盛り上がりを検出し，トピック毎にそれを表す bigram クラスタの集合 C_n を得た．しかし，設定した時間区間を超えて，トピックの盛り上がりが発生することがある．クラスタリング結果 C_n は時間区間毎に独立しているため，長く盛り上がっているトピックが存在する場合は，各時間区間の C_n 中から同一のトピックを表すクラスタを発見する必要がある．そのために本手法では，ひとつ前の対象時間区間のクラスタリング結果 C_{n-1} を利用し，トピックの追跡を行う．

n 番目の対象時間区間におけるクラスタリング結果 C_n のトピック毎の *bigram* 集合を $C_n = \{c_n^1, c_n^2, ..., c_n^h\}$ とし，含まれる bigram の個数は $|c_n^h|$ とする．この時，$C_{n-1} = \{c_{n-1}^1, c_{n-1}^2, ..., c_{n-1}^i\}$ の各クラスタに対して C_n の各クラスタとコサイン類似度で集合類似度を得る．

$$sim(c_n^h, c_{n-1}^i) = \frac{|c_n^h \cap c_{n-1}^i|}{\sqrt{|c_n^h| * |c_{n-1}^i|}} \quad (3)$$

これを組み合わせ毎に行い，閾値 $Bmin$ 以上かつ式 (3) が最大値になる c_{n-1}^i と c_n^h 組を，同一のトピックを示す bigram のクラスタとして同定する．また，c_{n-1}^i と同一トピックと判定された C_n の要素が複数ある場合は，それら C_n の要素の和集合をとる．図 5 のように C_n の各要素を修正した結果を，n 番目 の時間区間において盛り上がっているトピックを表す bigram のクラスタ集合 $C_n = \{c_{n-1}^1, c_{n-1}^2, ..., c_{n-1}^m\}$ とする．

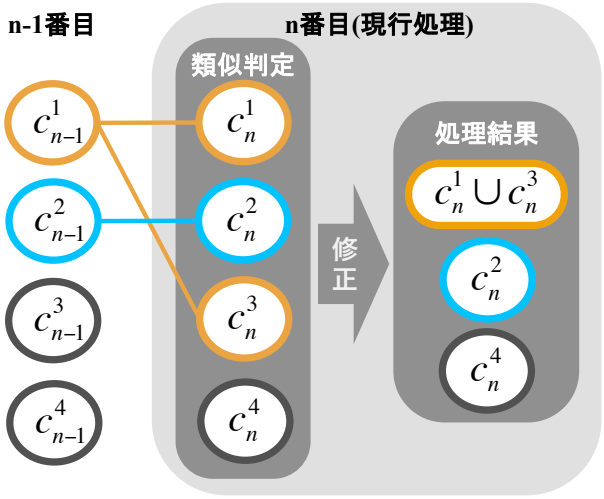

図 5　4.1.2 の処理の流れ

4.2 盛り上がり別の要約生成

4.1 で n 番目 の対象時間区間における実況タイムラインから，トピックの盛り上がりを検出し，トピック毎にそれを表す bigram のクラスタの集合 C_n を得た．これを利用してユーザに提示するための盛り上がりの要約を生成する．要約には n 番目の対象時間区間 の実況タイムラインに存在するツイート T_n からトピックを端的に示すツイートを一つ代表として抽出し，そのツイートをトピックのラベルとする方法をとる．

C_n の各トピックを表す bigram のクラスタに対して，各 bigram の区間中の出現頻度を利用した以下の式 4 による，ツイート t_n^j のスコアリングを行う．この時，t_n^j に含まれる bigram を $b_{t_n^j}$ とする．

$$score(t_n^j, c_n^m) = -|b_{t_n^j} \cup c_n^m - c_n^m| + \sum_{x \in b_{t_n^j} \cap c_n^m} freq_n(x) \quad (4)$$

これにより各トピックに対して，それを表す bigram をより多く含み，且つその他の bigram を含まないツイートのスコアが高くなる．式 4 が最大となるツイートを，各トピックを端的に表す代表ツイートして選択する．また c_n^m が $n-1$ 番目の時間区間において既に検出されていたトピックであるならば，過去に選択された代表ツイートとスコアを比較し，処理時点で最大のものを c_n^m のトピックの代表ツイートとする．

以上を対象時間区間ごとに実行し，リアルタイムに盛り上がっているトピックを発見し，トピック毎に内容を端的に表すツイート 1 つを要約として抽出する．

5. 実　　験

提案手法の有効性を評価するために，以下の 2 つについて検証実験を行った．

a. 提案手法 4.1 の手法が，実際にトピックの盛り上がりを検出できるのか．その妥当性の検証と最適な手法内パラメータの最適値の発見．

b. 実験 (a) により得た最適なパラメータを用い，生成された要約の妥当性の検証．

5.1 データセット

Twitter の UserStreamingAPI の filter [2] を利用して番組のハッシュタグを検索し，下記の 2 つの TV 番組の実況タイムラインを収集した．更にその中からツイート量が多かった 10 分間を選び，実験用データセット (表 1) を用意した．

表 1　実験用データセット

番組名	ハッシュタグ	収集期間	総ツイート数
下町ロケット	#下町ロケット	2015/12/20(日) 21:28〜22:38	2376
ワンパンマン	#onepunchman, #ワンパンマン	2015/12/21(月) 01:05〜01:15	6675

プレビューを稼ぐ目的等で注目度の高いハッシュタグを大量に付与して投稿されている検索妨害ツイートや，番組内容に同期しない可能性のあるツイートは，解析に際しノイズとなる．データセットでは，それらのノイズを除去するために，以下の条件に該当するツイートを収集の際に除去している．

- ハッシュタグが 5 つ以上付されているツイート
- リツイート (他ユーザのツイートを再投稿する機能で投稿されたツイート)

また収集の際に以下の条件でツイートを解析用の文字列に変換する処理を行っている．

- ハッシュタグ，URL，空白，改行の除去

・ 半角を全角カナへ変換，ひらがな，カタカナ共に大文字へ統一
・ 英数字は半角に統一，英字は小文字に統一

5.2 パラメータ試行実験

提案手法 4.1 で生成されるトピック別の頻出 bigram のクラスタリング結果がどれだけ人の直感に適合するかを検証した．この時 4.1.1 における DBScan における Eps 及び $MinPts$ の適切な値と，4.1.2 におけるコサイン類似度の適切な閾値 $Bmin$ を変化させ，よりよいクラスタリング結果を得るパラメータの最適値を検討した．

5.2.1 実験手順

検証にあたって以下の 2 条件を設定した (表 2)．$Amin$ は盛り上がっている bigram の検出力に関わり，条件 A，B は盛り上がりの判定条件が緩いものと厳しいものの 2 条件を，予め決定し，用意した．各条件，各サンプルに対してクラスタリング

表 2 設定条件

	条件 A	条件 B
$Amin$	2 秒に 1 度以上出現	1 秒に 1 度以上出現
処理間隔	番組開始から 5 秒間隔	
対象時間区間	処理時点から過去 10 秒間	

結果の精度を F-尺度により評価する．そのために，各条件で生成された対象時間区間毎の高頻出 bigram のリストを，サンプル毎に被験者に，トピック毎に人手でクラスタリングしてもらった．これを提案手法の結果と比較する正解データとした．まず提案手法 4.1.1 における DBScan の $MinPts$ と Eps の最適値を求めた後に，4.1.2 の $Bmin$ を求めるという形でクラスタリング結果の評価とその最適値の発見を行う．

F-尺度による評価は以下の手順で行う．n 番目 の対象時間区間における，あるパラメータで実行した提案手法によるクラスタリング結果 $C_n = \{c_n^1, c_n^2, ..., c_n^m\}$ と正解のクラスタリング結果 $A_n = \{a_n^1, a_n^2, ..., a_n^p\}$ がある．各トピック要素に含まれる bigram の個数はそれぞれ $|c_n^m|, |a_n^p|$ とする．C_n, A_n 内に存在する全 bigram の数は共に データ量 $N_n = |B'_n|$ である．この時手法により得られたクラスタ c_n^m と正解クラスタ a_n^p に対する再現率 R_{mp} と精度 P_{mp} を以下のように求める．

$$R_n^{mp} = \frac{|a_n^p \cap c_n^m|}{|a_n^k|} \tag{5}$$

$$P_n^{mp} = \frac{|a_n^p \cap c_n^m|}{|c_n^m|} \tag{6}$$

これら R_{mp} と P_{mp} の調和平均をとることで，c_n^m と a_n^p に対する F-尺度 F_n^{mp} が求まる

$$F_n^{mp} = \frac{2R_n^{mp}P_n^{mp}}{R_n^{mp} + P_n^{mp}} \tag{7}$$

さらに n 番目 の対象時間区間のクラスタリング結果に対する F-尺度 F_n は，a_n^p に対して，F_n^{mp} が最大になるような m を求めて F_n^{mp} を算出し，各 p に対して重み付き平均をとったもので表される．

$$F_n = \sum_{p=0}^{m} \frac{|a_n^p|}{N_n} \max_p F_n^{mp} \tag{8}$$

これにより対象時間区間毎の F-尺度を算出し，対象時間区間毎のデータ量 N_n に対して，重み付き平均をとることで，あるパラメータで実行した提案手法 4.1 におけるクラスタリング結果の F-尺度 F とした．この時，対象時間区間毎のデータ量 $N_n = 0, 1$ の時は計算から省いている．

$$F = \sum_{n=0}^{n} \frac{N_n}{\sum_{n=0}^{n} N_n} F_n \tag{9}$$

これを各条件，各サンプルにおいて，パラメータを変化させながら算出し，クラスタリング結果を評価した．最も良い評価となるパラメータとそのクラスタリング精度をした．

5.2.2 結　　果

表 3 パラメータ試行実験結果

	下町ロケット	ワンパンマン
実験条件	条件 A	条件 B
DBScan の Eps	0.7	0.1
DBScan の $MinPts$	1	
$Bmin$	0.3〜0.4	
F-尺度	0.86924	0.894201

各サンプルにおいて最も良い結果となったパラメータ群を下記に示す (表 3)．最適なパラメータで実行した提案手法 4.1 によるトピックのクラスタリング結果は，両サンプルとも F-尺度による評価が 0.9 近い高い精度で行えていた．しかし，両サンプルに効果的な $Amin$ の値と Eps の値の最適値は一致しなかった．図 6 は各時間区間毎に，正解データと最適なパラメータで実行した提案手法 4.1 が発見したトピッククラスタの数である．$Amin$ は盛り上がりの検出する閾値であり，DBScan の Eps はクラスタリングを行う上での，bigram の出現頻度の推移パターンが類似していると見なす閾値である．「下町ロケット」は，ツイート量と盛り上がっている話題数が全体的に「ワンパンマン」より少なかった為に，$Amin$ はより盛り上がりを検出できる様に低く，Eps はクラスタリングの過分割が減るように大きく設定する必要があった．

トピックの追跡に関しては，それぞれのサンプルにおいて最適なパラメータで実行した提案手法 4.1 において,「下町ロケット」で 28,「ワンパンマン」で 127 のトピックを発見できた．

5.3 評価実験

パラメータ試行実験 5.2 により得た結果を元に，要約を生成する提案手法 4. の実行パラメータを以下に定めた (表 4)．各サンプルに対して，提案手法 4. を適用し，対象時間区間毎にトピック毎の要約として選ばれたツイートが，区間の実況タイムラインの要約として妥当かを被験者実験により評価した．またツイートを用いる要約の提示形態がどの程度効果的なのかを，坂本ら [8] 等が用いている要約としてキーワード群を生成する提示形態を，ベースラインとして比較評価する．

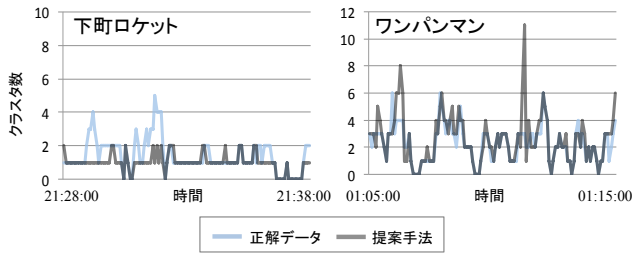

図 6　サンプル毎に各対象時間区間において存在したトピッククラスタの数の推移グラフ．黒い線がサンプル毎に最適なパラメータ群で実行した提案手法 4.1 が生成した結果，青い線が正解データ

表 4　各サンプル毎に設定した，提案手法 4. の各種パラメータ

パラメータ＼サンプル	下町ロケット	ワンパンマン
処理間隔	番組開始から 5 秒間隔	
対象時間区間	処理時点から過去 10 秒間	
$Amin$	2 秒に 1 度以上	1 秒に 1 度以上
Eps	0.7	0.1
$MinPts$	1	
$Bmin$	0.3	

5.3.1　ベースライン手法

提案手法と同様の処理間隔と時間区間を与える．

処理間隔：　番組開始から 5 秒間隔

時間区間：　処理時点から過去 10 秒間

時間区間内の全ツイートに対して MeCab [12] による形態素解析を行い，名詞，動詞，形容詞の形態素を抽出する．この時，ストップワード等の除去を行っていない．n 番目の対象時間区間中に発生した単語集合を W_n とする．この時各単語は $W_n = \{w_n^1, w_n^2, ..., w_n^m\}$ で表される．何かのトピックが盛り上がっている場合，そこに含まれるフレーズやキーワードの数が多くなると考え，区間内の全ツイートから w_n^m を含むツイートの数を求めた．量が多い順に 10 件の単語を抽出し，それを n 番の時間区間を要約したキーワード集合とした．

5.3.2　実 験 手 順

表 4 のパラメータを用いて，サンプル毎に提案手法 4. を適用した．生成された対象時間区間毎の要約で，「ワンパンマン」の要約の一部を付録に示している．サンプル毎に対象時間区間をランダムにトピックの要約として選ばれたツイートが，各対象時間区間の実況タイムラインの要約として妥当かどうかを，下記の手順で評価してもらった．被験者数は 20 代の男女 20 名である．

(1)　サンプル別に実況タイムライン上から，提案手法による盛り上がり検出と要約が行われている，重複しない数十秒間を無作為に 5 つ選び，被験者に閲覧してもらう．

(2)　その期間に含まれる連続した 2 つ対象時間区間の要約を，ベースラインと提案手法別に，被験者に提示する．

(3)　被験者は提示された要約が，自身が閲覧したタイムライン上の盛り上がりの要約として適切かどうかを，上は「6: 適切である」，下は「1: 適切でない」の 6 段階で評価してもらう．

5.3.3　結　　果

それぞれのサンプルについての結果を示す (図 7)．タイムライン上の盛り上がりの要約として，提案手法により選ばれた代表ツイートが妥当かどうかの評価に対して，全てのサンプルで平均 4 以上の評価を得た．評価値の平均は，「下町ロケット」では 4.57，「ワンパンマン」では 4.94 だった．またベースライン手法によるキーワード群の提示が盛り上がりの要約として妥当かどうかの評価はに対して，全てのサンプルで平均が提案手法を下回った．評価値の平均は，「下町ロケット」では 3.36，「ワンパンマン」では 3.4 だった．全てのサンプルに対して，提案手法とベースラインの間にマン・ホイットニーの U 検定で $p < 0.001$ 以下の有意差があった．

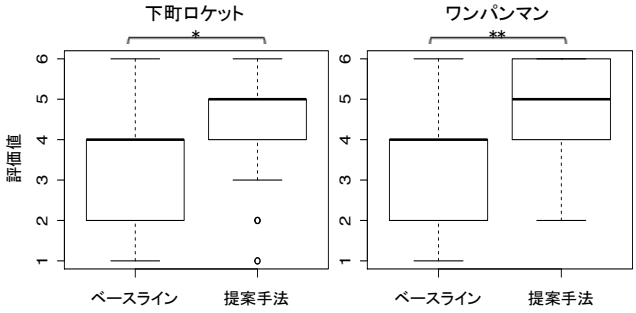

図 7　提案手法とベースライン手法の評価値比較．ボックス内の太いバーは第 2 四分位点を表し，ボックスの上辺は第 3 四分位点，下辺は第 1 四分位点を表す．マン・ホイットニーの U 検定により，*，**の p 値共に 0.001 以下の有意差が確認された．

提案手法 4. によって生成された盛り上がりの要約として選ばれた代表ツイートが，実際の実況タイムラインの盛り上がりの要約として妥当であると示された．また簡便なベースライン手法よりも提案手法のほうが有効であるという結果が得られた．こちらは，ベースライン内でキーワードの選別のために「する」等々の要約として取得するべきでない言葉を除去する等を行っていないので，より洗練された手法との比較を行う必要がある．

また本手法の盛り上がり検出及び要約の生成は，リアルタイム用いる事が可能な処理の流れで実装している．言語は python を使用している．OSX(10.10.5)，プロセッサ 2.5GHz Intel Core i5，メモリ 4GB の環境で，表 4 のパラメータ設定で，それぞれの対象時間区間毎に要約を生成するのに下記の時間を要している．これはリアルタイム処理が現実的に行える可能性を示している．

表 5　サンプル毎の，対象時間区間の盛り上がり検出と要約生成に要した平均処理時間 (s)

下町ロケット	ワンパンマン
1.4104181	0.7930074

6. プロトタイプシステム

実験 5. により得たサンプル別の適切なパラメータを用いて提案手法を実装し，生成された各時間区間の要約を生成した．生成されたトピック毎の盛り上がりの要約をユーザに視覚的に伝える為のプロトタイプシステムを作成した (図 8)．

生成された時間区間毎の盛り上がりの要約が，その時間の映像とともに一覧できるインターフェイスを実装している．再生時間を変更すると，変更された時間に相当する要約が自動的に可視化される．

図 8　サンプル「ワンパンマン」を用いたプロトタイプシステムの動作例

6.1　盛り上がりの可視化

動作プロセスを図 9 に示す．可視化の方法については，トピック毎にボールド体の文字列を生成している．対象となる時間区間におけるトピック毎の盛り上がりの割合によって，生成される文字列の大きさを変えている．これにより，今何が盛り上がっているか，どの盛り上がりが多く発生しているかという事を視覚的に提示する．上記の処理を実現するために，提案手法より新たな処理層を設けている．期間中の盛り上がりの中で式 4 により得たスコアの高いツイートが選ばれているトピックほど文字の大きさが大きく現れる処理を行い，文字列を生成している．

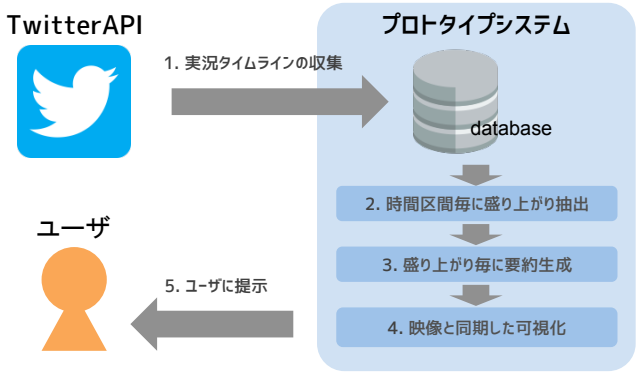

図 9　プロトタイプシステムの処理プロセス

7. ま と め

本稿では Twitter におけるソーシャルビューイングに参加するユーザの支援を目的に，実況タイムライン上で発生するトピックの盛り上がりに着目し，そのリアルタイムな検出，要約によって実況タイムラインを集約化する手法を提案した．被験者実験によって手法がトピック毎の盛り上がりを高い精度で取得でき，トピックの効果的な要約が生成できていることが確認さてた．今後は，実装したプロトタイプシステム 6. を使用した被験者実験を行い，実際にユーザの負担を軽減できるかどうかを検証していく予定である．

本手法は提案手法のパラメータを全てのテレビ番組に対して一般的に適用しうるパラメータを発見できなかった．しかし，盛り上がっている bigram をどの程度検出しうるかの *Amin*，またクラスタの粒度を調節しえる *Eps* 等は最終的なプロトタイプシステムにおいて，ユーザが任意で調整できるパラメータとして実装し得るものである．ユーザが視聴したい番組のソーシャルビューイングに合わせてシステムパラメータを可変することで，より良い結果をもたらすかどうかも今後の研究で検討する必要がある．

文　献

[1] Twitter, https://twitter.com/

[2] Twitter Developers, “Public API POST statuses/filter”, https://dev.twitter.com/streaming/reference/post/statuses/filter (2016-01-9)

[3] SONY, ”Twitter 連携 | ネットサービスを楽しむ | 液晶テレビ BRAVIA ブラビア | ソニー”, http://www.sony.jp/bravia/technology/internet/twitter.html (2015-12-26)

[4] Livetter.com, ”Twitter で実況しよう” , http://livetter.com/ (2015-12-26)

[5] tomstay, ”つぶあに - アニメの視聴管理・実況アプリ”, https://play.google.com/store/apps/details?id=com.tsubuani.android2 (2016/01/09)

[6] 中澤昌美, 帆足啓一郎, 小野智弘, ”Twitter による TV 番組の重要シーン検出及びラベル付加手法”, 全国大会講演論文集 vol.2011,no.1,pp.517-519, 2011-03-02

[7] 久保光証, 笹野淳平, 高村大也, ” ”良い実況者”に着目した Twitter からのスポーツ速報生成”, 言語処理学会第 19 回年次大会, 2013-03

[8] 坂本翼, 廣田雅春, 横山昌平, 福田直樹, 石川博, ”Twitter ストリームの断続性に着目したキーワード抽出”, DEIM Forum 2012 C7-3, 2012

[9] 水落大史, 井上悦子, 吉廣卓哉, 村川猛彦, 中川優, ”新聞記事集合に対する時系列のトピック抽出”, DEIM Forum 2010 D6-3, 2010

[10] 芹澤翠, 小林一郎, ”文章内のトピック数を考慮したトピック追跡の試み”, 言語処理学会第 18 回年次大会, 2012-03

[11] M.Ester, H.-P.Kriegel, J.Sander, and X.Xu ”A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, KDD1996

[12] MeCab, http://mecab.sourceforge.net/

付　　録

「ワンパンマン」のサンプルに対して，提案手法 4. を適用し，生成された要約の一部を掲載する．

表 6　ワンパンマンのサンプルに提案手法 4. にかけて生成した，時間区間の要約を表すツイートの一部．同一のトピックと判別されたものには通し番号を振り分けている．

01:09:26〜01:09:35	57	なかなか楽しませてくれるじゃねか
01:09:31〜01:09:40	58	爺さん生きてた
	59	肩こりが取れたぜ
01:09:36〜01:09:45	58	じいさん生きてた
	59	肩こりが取れたぜ
	60	どこを見ている
	61	おい
01:09:41〜01:09:50	59	肩こりとれた
	60	どこを見ている
	62	アトミック斬
01:09:46〜01:09:55	62	アトミック斬
	63	生意気な
01:09:51〜01:10:00	62	アトミック斬
	63	斬！
01:09:56〜01:10:05	62	アトミック斬
01:10:01〜01:10:10	64	ある！
	62	アトミック斬
01:10:06〜01:10:15	64	ある！
	65	王手じゃ
	66	くそおお
01:10:11〜01:10:20	65	王手じゃ
	66	くそおお
01:10:16〜01:10:25	65	王手じゃ
	66	くそおおお
	67	かっこ e
01:10:21〜01:10:30	65	王手じゃ
	67	かっこ e
	68	余計なことするんじゃねえよ
01:10:26〜01:10:35	67	かっこいい
	68	余計な事するんじゃねえぞ
	69	爺さん強かった
01:10:31〜01:10:40	67	かっけえ …
	70	ひとまず勝利だ
01:10:36〜01:10:45	70	ひとまず勝利だ
01:10:41〜01:10:50	70	ひとまず勝利
01:10:46〜01:10:55	71	無免ライダー
01:10:51〜01:11:00	72	無免さん一行
	71	無免ライダー
01:10:56〜01:11:05	73	なにしてんだ
	74	op
	71	無免ライダー