情報処理学会論文誌　データベース　Vol.6　No.2　61–84 (Mar. 2013)

文脈的なつながりを考慮したツイート群の効果的な抽出・提示手法の実現

青島 傳隼[1,†1]　坂本 翼[1,a)]　横山 昌平[2]　福田 直樹[2]　石川 博[2]

受付日 2012年9月20日, 採録日 2013年1月3日

概要：Twitter を代表とするマイクロブログサービスでは，スマートフォンなどの小型携帯端末からの投稿および閲覧が行いやすいこともあり，従来のブログにはない即時性を持つことが特長となる．Twitter のように，フォローという関係を用いて複数の発信者の記事を時系列一覧（タイムライン）表示させる閲覧方法では，ユーザが閲覧対象としたい発信者数が増加した場合や共通の話題に対して複数の発信者から類似の記事の投稿が生じた場合などに，特に小型携帯端末の限られた表示領域での閲覧性の改善が課題となる．本論文では，この課題を解決するために，単語の共起とタイムスタンプの情報を用いてツイート間のつながりを発見し，同一話題に言及しているツイート群を抽出する手法を提案する．また，本手法を適用し，小型携帯端末上で効果的にタイムラインを提示可能なシステムの実現について述べる．

キーワード：マイクロブログ，閲覧支援，クラスタリング

Toward Better Extraction and Presentation of Twitter Articles Based on Contextual Connections Among Them

Tsugutoshi Aoshima[1,†1]　Tsubasa Sakamoto[1,a)]　Shohei Yokoyama[2]
Naoki Fukuta[2]　Hiroshi Ishikawa[2]

Received: September 20, 2012, Accepted: January 3, 2013

Abstract: The widespread use of Twitter and other micro-blog services emerge huge importance of effective and efficient processing of those data to effectively use these services and to analyze the produced data for many purposes. Due to rapid increase of users and the produced articles (tweets) posted by the users, it is difficult for the user to grasp the tweets in the timeline and track the context of these tweets. In this paper, we propose a method for extracting contextually related tweets reflecting their contexts by using cooccurrence of words and variance of timestamps of those tweets. In addition, we apply our approach to build a prototype system for smart phones and we show that our prototype system effectively assists users to browse tweets that might quickly flow out from its timeline.

Keywords: microblogging, browsing support, clustering

1. はじめに

マイクロブログとは，チャットとブログの中間に位置するようなサービスである．その代表的なサービスの 1 つである Twitter [39] には，フランスの調査会社 Semiocast によって 2012 年 1 月 31 日に発表された内容 [28] によると 2011 年の末時点で 3 億 8,300 万以上のアカウントが登録されており，そのうち米国では 1 億 770 万アカウント，日

[1] 静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
[2] 静岡大学情報学部情報科学科
Department of Computer Science, Faculty of Informatics, Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
[†1] 現在，株式会社キテラス
Presently with Qteras, Inc.
[a)] gs11022@s.inf.shizuoka.ac.jp

図 1　ユーザ Bob_Mk2 のタイムライン画面．Bob_Mk2 がフォローしている他のユーザ（発信者）のツイートが表示されている．
Fig. 1　Example of timeline in twitter.

本国内は 2,990 万アカウントが利用されているとされる．Twitter に投稿されるツイート数は，公式の発表 [40] では 2011 年 6 月 30 日時点で，全世界で 1 日あたり 2 億件に達している．ここから得られる情報の効果的な活用の必要性が指摘されている [7]．

マイクロブログサービスの特徴の 1 つに，高い即時性がある．従来のブログサービスでは，多くとも 1 日 1 回程度の更新が一般的であるとされる[*1]が，マイクロブログの場合には 1 日に複数回書き込まれることも一般的であり，テレビ番組やスポーツの試合の様子が，あたかも実況中継されるかのように記事として投稿される場合もある．この即時性を支える 1 つの要因として，スマートフォンのような小型携帯端末からの記事の投稿が比較的容易である[*2]という点があげられる．

2011 年 3 月 11 日に起こった東日本大震災時には，Twitter が，被災状況の共有や避難所情報の確認などのほか，電話がつながらない，つながりにくい地域に住む被災者の安否確認などにも活用された．これは，Twitter の持つ，次のような仕組みが効果的に働いたからであると考えられる．Twitter では，個々のユーザが発信者となり，そこで投稿されるツイートと呼ばれる記事が，「フォロワー」という関係を持つユーザに配信される．ユーザは他の複数のユーザを「フォロー」することでその「フォロワー」となることができ，それらの「フォロー」した発信者たちのツイートが「タイムライン」というリストに時系列に配置されるようになっている．**図 1** は，ユーザ Bob_Mk2 のタイムライン画面の例であり，中央部の領域がタイムラインである．

このように，Twitter では，個々のユーザによって見ている「タイムライン」は異なっており，それぞれのユーザの関心に応じた最新のツイートを閲覧できると同時に，それらのツイートを自身のフォロワーに伝えたり（リツイートと呼ぶ），そのツイートの一部を引用したツイートを行うことなどにより，ユーザ同士のつながりに沿って，それらのユーザに注目されるような情報がツイートを通じて波のようにすばやく広がっていく現象が起きる．この仕組みにより，ユーザの関心を引くような情報を含んだツイートが，取捨選択をされながらも，非常に速いスピードで広がっていく現象が起きていると考えられる．一方で，Twitter ではこのような仕組みを持つがゆえに，その閲覧性について次のような課題も生じる．

Twitter の場合では，多くの発信者が同一の話題に対して言及したツイートを投稿した場合，非常に近い内容のツイートが，自身が閲覧対象とする最新のタイムラインに多数表示されることになる．ある 1 つの話題について書かれたツイートがタイムラインの多くを占めると，それ以外の話題について触れたツイートを見落としがちになる．この現象は，特に表示領域の限られた小型携帯端末上で閲覧する場合に顕著となると考えられる．

本論文では，同一の話題について言及するツイート間のつながりを発見し，また，それらを 1 つにまとめ，ユーザに対して効果的なタイムラインの閲覧を実現するシステムの実現について考える．この目的のために，本研究では，単語共起とタイムスタンプ情報を効果的に用いたツイート間のつながり抽出と，それに基づくツイート集約手法を提案する．提案手法が，スマートフォンなどの小型携帯端末から利用可能とするためのシステムのアーキテクチャおよび実装上の制約を満たすものであることを，プロトタイプシステムの実装により示す．

本論文の構成は次のとおりである．2 章では，本研究の背景および関連研究に対する本研究の位置づけを示す．3 章では，同一の話題に言及するツイート群の抽出方法について述べる．4 章では，抽出方法の実験と評価を行い，その有効性を示す．5 章では，提案した手法に基づくシステムの実装について，詳細を述べる．6 章では，本論文のまとめと今後の課題について述べる．

2. 研究の背景

本章では，Twitter およびそれらを代表とするマイクロブログサービスの特徴と固有の課題について概観するとともに，その課題に対する既存のアプローチの適用可能性を議論し，本研究の位置づけについてまとめる．

2.1 Twitter の特徴と閲覧性の課題

マイクロブログサービスでは，1 つの投稿に対して強い

[*1] State of the Blogosphere [29] の 2009 年の調査によると，1 日に 1 回以上ブログを更新する利用者の割合は全体の 32.9%で，1 日に 3 回以上更新する利用者の割合は全体の 10.5%である．
[*2] アスキー総研の 2009 年 12 月の調査 [42] によると，利用者の約 39%が，スマートフォンから Twitter にアクセスしている．

字数制限が設けられる場合があり，Twitter では 140 字に制限される．この字数制限により，ある 1 つの出来事に対して 1 回の投稿ですべての内容を網羅的に述べること（たとえばある商品を購入し，使用した感想を細かく述べる，など）は困難となる．すなわち，従来のブログであれば 1 度の投稿としていた内容が，たとえば，「買った」，「使ってみます」，「使ってみた感想は…」の 3 回に分けて投稿される．

Java ら [18] は，Twitter に多くの記事が投稿される要因として，従来のブログサービスと比較したときに，記事の投稿にかける時間の違いから，投稿内容について考える負担が下がっている可能性を指摘している．

奥村 [25] は，従来のブログとは異なるマイクロブログ固有の特徴として，「今」現在の実世界に関する情報を発信している点，および，1 つの投稿に対する長さの制約などの要因により，これまでのテキストとはかなり異なった言語使用が行われている点を指摘している．

Twitter の持つ，少ない文字数で手軽に書けるという特徴は，文字入力などに制約のある携帯電話やスマートフォンなどの小型携帯端末との親和性を高めることになる．一方で，表示領域の限られたこれらの端末上における，ツイートの閲覧性・検索性の確保が課題となる．

Twitter では，この文字数制限により，ツイート内に必ずしもその話題の内容を代表するキーワードが文字列として含まれるとは限らない．Kwak ら [12] は，この課題に対し，話題に対応するハッシュタグをツイートに付与することで，140 字という限られた文字数の中でも関連する話題のツイートを見つけやすくなる点を指摘している．一般に，ハッシュタグは，「#hashtag」のように「#」＋「タグ名」として，ツイートに文字列として付与される．これにより，たとえば「#soccer」のハッシュタグで検索を行うことで，「サッカー」という単語を直接含んではいないツイートもサッカーの話題に対する検索結果として得ることができる．一方で，ハッシュタグのみによるツイートの閲覧性・検索性の改善には，限界がある．たとえば，同日の話題に対して複数のハッシュタグが並列して用いられる場合[*3]や，異なる話題に対して同一のハッシュタグが用いられてしまう場合[*4]への対応が課題となる．また，話題とハッシュタグ名との関係を何らかの手段で事前に知っていなければ，それを検索に用いることが困難である．さらに，特定のハッシュタグに対するツイートが多数存在した場合には，ハッシュタグ以外の方法で候補の絞り込みを行う必要が出てくる．

Twitter では，閲覧対象の絞り込みの手段として，各ユーザごとにタイムラインに表示したい他のユーザ（発信者）をフォロワーとして指定することで，それぞれに対するタイムラインの表示を構成できるようにしている．

Hannon ら [11] は，Twitter のユーザが，フォロー対象としている発信者のツイート内容から自身に最適なタイムラインを形成する傾向があるとし，一般的にフォロー，アンフォロー（フォローを外す）を繰り返しながらその調整をしていく傾向があることを指摘している．

この特性により，互いにフォロワーの関係にある Twitter ユーザ同士であっても，閲覧しているタイムラインの内容は異なったものとなる場合があるため，自身のタイムライン上にあるツイートを自身のフォロワーに対して再配信したいというニーズがあり，この再配信の操作をリツイートと呼ぶ．

リツイートは，前述の事情により，Twitter のユーザの間で，他のツイートの内容を引用してその先頭に発信者 ID を付与した，「RT@発言者 ID 内容」のように記述したツイートを行うことから自然発生したと考えられる．この操作を同一のツイートに対して多数の発信者が行った場合，ほぼ同一の内容が複数の発信者からツイートされることになる．これは，タイムラインがほぼ同一のツイートで埋め尽くされてしまう状態を作る 1 つの典型的な要因となる．

Twitter の持つ公式のリツイート機能を用いてリツイート操作を行った場合には，多数の発信者からの単一のリツイートは 1 つのものとしてタイムライン上に表示されるようになっているが，リツイートでは，単に引用のみでなく，そこにコメントを加える形で「コメント RT @発信者 ID 内容」のようにツイートされる例もあり[*5]，この操作に相当する公式なリツイート機能は本論文執筆時点では用意されていないため，リツイート操作による閲覧性の低下は必ずしも解決されない．また，コメントとして付与された情報が少ない文字数であってもユーザの閲覧ニーズに応えるものであった場合には，それは閲覧対象となるようにしたい．

リツイートという操作によらない場合であっても，同時刻に複数の発信者から非常に類似したツイートがなされる場合もある．典型的な例としては，サッカーの試合を観戦している発信者が，ゴールの瞬間に「ゴーーール！」などとツイートする場合があげられる．

このように，特定の話題に関するツイートが，タイムライン上でバースト的に発生した場合には，時系列表示に基

[*3] たとえば，セマンティック Web に関する著名な国際会議である International Semantic Web Conference 2012 では，語彙やオントロジーの統一の問題を扱うはずの研究者の間でも複数のハッシュタグが統一できないでいた点が話題となった．

[*4] たとえば，2012 年に行われた人工知能学会全国大会では，オノマトペに関する話題を扱うハッシュタグと，あるソフトウェア会社の製品に関する話題を扱うハッシュタグが意図せず衝突してしまった点が話題となった．

[*5] 現在は，Twitter の機能を用いて記事をリツイートすることを公式 RT，記事の内容を引用しつつそれにコメントを付けてツイートすることを QT（Quote Tweet）と呼ぶ傾向がある．旧来のように RT を自分で付与してツイートすることは，どちらかといえば QT に近い．文書の改竄が可能か否かが公式 RT か非公式 RT（QT，旧来の RT）との違いの 1 つであり，情報の信憑性を重要視する場合はが公式 RT を行うことが一般的であるが，本論文執筆時点ではまだ非公式 RT も多く見られる．

づくタイムラインでは，同じ話題に関するツイートが溢れ，他の話題の閲覧が困難になってしまう．

2.2　既存の閲覧支援手法の適用可能性

大量の情報から必要なものを取捨選択して効果的に閲覧できるようにするための既存の技術としては，情報フィルタリング [13] に関して幅広い研究が行われてきており，凝集法や k-means 法 [17] などのクラスタリング手法，半教師付き学習 [3] を用いた制約付きクラスタリング [43]，Support Vector Machine [5] などの機械学習を用いた記事分類手法 [14] などがあげられる．

文書中の単語の出現頻度から求めた $tf \cdot idf$ をもとに文書分類やクラスタリングを行う手法としては，たとえば，Joachims [19] の，PrTFIDF と呼ばれる確率的分類法，徳永ら [38] の重み付け IDF（WIDF）による手法がある．分類手法を検索結果の提示方法の改善に用いた例としては，たとえば村松ら [24] の，$tf \cdot idf$ と条件付き特徴量を用いた手法があり，検索結果の記事集合にクラスタリングを適用することで，ユーザのニーズに沿った検索結果の提示を試みている．

これらの手法の適用対象を一般の文書集合からブログ記事に拡張してその適用を試みた事例としては，戸田ら [37] の品詞の重み付けに着目したクラスタリング手法などがあるが，マイクロブログ固有の特徴として奥村 [25] が指摘する，これまでのテキストとは異なった言語使用がなされる点への対処を考えれば，これらを単純に適用するのみでは必ずしも期待した性能を発揮しないことが考えられる．

ツイートのような短い文章の記事を分類やクラスタリングする方法として，Phan ら [26] は，Wikipedia [45] の記事を学習対象として，短い記事を分類する手法を提案している．Phan らの手法では，Latent Dirichlet Allocation（LDA）でモデルを生成し，Maximum Entropy（ME）で分類器を作ることで，学習データが少なくても比較的高い精度で短い文章の記事が正しく分類されることを示している．Hu ら [15] は，短い文書をクラスタリングする際に問題となる，単語のまばらさを補完するために，Wikipedia と WordNet [23] を用いる手法を提案している．

Twitter に対して適用した事例としては，たとえば，竹中ら [34] によるハッシュタグの自動付与に関する研究がある．竹中らは，ハッシュタグが付与されたツイート群から，ベイジアンフィルタを生成し，学習データ内に存在しない単語があり出現確率の積を用いるフィルタは有効に働かない場合であっても，加算法と既知語限定処理を組み合わせスムージング処理を行うことで精度を向上させている．Go ら [10] は，ツイート内の感情に関する語や顔文字から特徴量を求めることで，ツイートのポジティブ・ネガティブ分類を行っている．

マイクロブログを含めたブログ記事には，記事としてのテキストの内容以外にも，それが投稿された時間情報や複数の記事間での関連付けに関する情報も付与されており，これらの利用も有益であると考えられる．たとえば，鎌田ら [20] は，ブログ記事を対象にした場合に，$tf \cdot idf$ のような出現単語そのものの頻度情報などは用いずに，トラックバックなどといった記事に付与された情報の考慮や出現単語の品詞レベルでの頻度情報などを用いた場合でも，ブログ記事の分類を効果的に行える場合があることを示している．

ツイートの分類などへの適用としては，たとえば，Sriram ら [30] の，Twitter ユーザのプロフィール情報からの特徴抽出による手法がある．Sriram らの手法では，ユーザプロフィールに加え，時事イベントや，感情語などの 8 つの特徴量を用いることで，高い精度で，ツイートをニュース・イベント・意見・詳細・プライベートメッセージの 5 つに分類できることを示している．Cha ら [2] は，ユーザの発信者としてのツイートの影響力に対する，フォロワー数，リツイートされた回数，他のユーザとの会話数などとの関係を指摘し，従来の現実世界におけるそれぞれの人が持つネットワークの関係から導き出せるような「影響力」を，Twitter のフォロー関係からも導き出せることを指摘している．Duan ら [8] は，ユーザ間の，フォローしている・されているという関係を用いたツイートのランキング手法を提案している．

時間情報を持つ対象に対しては，Burst 検出法 [9], [21] などの時系列情報に着目した手法の適用も考えられる．時系列的に連続した文書をまとめる方法として，石川ら [16] は，クラスタリング手法に対し，時系列に着目した「忘却（forgetting）」の概念を導入し，そこで定義した忘却係数を文書間の関連度を求める際に用いることで，k-means 法によるクラスタリング精度が向上することを指摘している．角谷ら [31] は，Web 上のニュース記事を対象として，ユーザの要求にあった価値の高い情報を提示するための時系列クラスタリングを適用している．角谷らの手法では，ニュースの続報に特に着目し，ニュース記事間のつながりを 3 パターンに分類している．この分類に基づいて続報を検出するために続報リストを生成し，新着ニュース記事と既存の続報リストとの関連度を測ることで，記事間のつながりの発見を試みている．

時系列情報をマイクロブログに適用したものとしては，高村ら [33] のイベント要約に関する研究がある．高村らは，時間とともに動的に変化する特定の話題に関する記事エントリの集合をマイクロブログストリームと呼び，そのストリーム中の特筆すべきイベントの要約を行っている．高村らは，たとえばサッカーの試合に関する記事のような，特定の話題に関連して収集されたエントリ集合に対し，代表的な記事エントリ（代表エントリ）を抽出し，そこに代表記事に内容的に被覆されるような記事を関連付ける

ことで，トピック中のイベント（たとえば，サッカーであればゴールシーンなど）の要約を行っている．坂本ら [27] は，Twitter に投稿されるツイート群を一種のストリーム情報として扱い，着目した話題内での内容の変遷に対して，Burst 法を用いて要約されるべきイベントをリアルタイムに検知し，そのイベントに関するキーワード集合を得る手法を提案している．

2.3　本研究の位置づけ

本研究では，ツイートのタイムライン表示の閲覧支援にも適用できる手法について考えたい．そのための課題は大きく 2 つある．

課題の 1 つは，Twitter におけるタイムライン表示の特徴を考慮した手法の実現である．Twitter においてタイムラインに表示されるツイート群は，たとえば高村らが文献 [33] で扱うような，特定の話題に関連して収集されたエントリ集合であるとは限らない．Hannon らが文献 [11] で述べるとおり，ユーザ自身の手で最適なタイムラインを形成するためにフォロー関係の更新が随時行われているが，その選択は対象となる情報であるツイートではなくその発信者の単位での選択である．ある発信者のツイート内容が複数の話題を含む場合には，タイムラインにも必然的に複数の話題が含まれてくることになる．一方で，タイムラインに表示されるツイート群は，ユーザ自身のタイムラインの形成にともなってそれぞれ個別の偏りを持つことが想定される．これは，たとえば特定の新聞社が発行する特定の期間内の特定の分野（たとえば，政治・経済など）の記事全体といったような，ある分野の内容について平均的に集められた文書集合とは限らない．さらに，Cha らが文献 [2] で指摘する点を逆説的に考えれば，特定の発信者をサポートする目的など，社会的なつながりからその発信者のフォロワーとなる場合もあると考えられ，タイムライン表示の最適性のみを考えてフォロワーが選択されるとは限らない．このことが，タイムラインの高速化と呼ぶような，自身のタイムライン上の大量の情報のフローの発生を起こすことの要因の 1 つとも考えられる．

また，その話題の生起も，必ずしも事前に予測可能なものであるとは限らない．たとえば，100 年に 1 度の震災を事前に正確に予測することは容易ではなく，それに関連した重要な情報が，その震災が起きてから新たに現れた発信者（たとえば政府関係組織により新たに開設されたアカウント）や，それまではその話題に関係した発信を行ってこなかった発信者（たとえば普段は記者クラブ問題を扱っているジャーナリストや IT 長者など）から出てくるようなことを事前に予測することは難しい．すなわち，事前に規定された分類クラスへの分類のみでは十分ではない場合がある．Twitter にも，フォロワーとは独立に閲覧対象となる発信者を絞り込めるリスト機能があり，特定のクライアントソフトウェアの持つツイートの絞り込み機能などの利用も可能ではあるが，これらの多くは事前にユーザ自身の手で設定が必要であり，必ずしも話題の生起に即時的に対応できるような自動化がなされているわけではない．

もう 1 つの課題は，Twitter の利用環境としての，スマートフォンなどの小型携帯端末上での動作を実現するための実装やアーキテクチャ上の制約への対応である．実装上の制約として，スマートフォンなどの小型携帯端末は，クライアント PC やサーバと比較して，搭載メモリ量や処理能力の面以外に，バッテリ駆動であることを前提とした消費電力の削減にも配慮する必要があり，常時増え続けるタイムライン上のツイートに対して，それらの端末上ですべての処理を行わせることは現実的ではない．一方で，サーバサイドですべての処理を行うような Twitter 閲覧サービスを構成した場合には，操作の際に通信のレイテンシをともなうという課題以外に，Twitter クライアントとしての各種機能をサーバサイドに実装する必要が出てくるため，実装および運用保守上の負担が大きいことや，そのサーバに障害が起きたときに Twitter の閲覧などの操作が行えなくなるなどの点が課題になる．この課題を回避するには，たとえばユーザの持つ小型携帯端末上で Twitter クライアントソフトウェアが単体で動作可能でもあり，そのクライアントソフトウェアがサーバサイドから提供される補助情報に基づく簡易な処理によって閲覧支援の機構を実現できるようにする，といった方法がある．

また，小型携帯端末には，その大きさから，表示領域上の制約がどうしても生じる．たとえば，共通の話題に言及しているツイートが 10 件あった場合，それを 1 つのツイート群としてまとめることで，空いた 9 件分のスペースに他の話題のツイートも表示できるようにし，それらも見逃さずにすむようにしたい．一方で，共通の話題に言及している 10 件のツイートについても，単にそれらの要約を示すだけとするのではなく，その 1 つ 1 つのツイートも必要であれば閲覧できるようにしたい．ここでいう「まとめる」という用語は，主に集約を意味し，「特定の話題に関して触れているツイートを，ただ 1 つのまとまりとして集め，他の話題のツイートなどを表示できるような余地を作ること」を目的とする．

すなわち，小型携帯端末上での閲覧支援への適用を考えた場合，その閲覧支援のための手法そのものを単体で考えるのではなく，実際にそれをシステムとして動作させることも加味して，前述のようなシステム設計・実装上の制約などへの対処も同時に考慮したような手法を考えたい．

本論文で提案する手法の特徴的な部分は，1 つには，自然言語処理の分野で行われるトピック検出およびトピック追跡に相当する動作（たとえば，文献 [4], [32] など）を，Twitter のツイートという個別にタイムスタンプを持った短い文章を主体に構成された対象に対して，動的にかつ効

果的に行えるようにするという点である．もう 1 つの特徴的な点は，文献 [4] などでのトピック特徴量に相当するものを単語クラスタとしてサーバサイドで計算させ，その単語クラスタを用いたツイートに対するトピックとの関連づけに相当する操作をクライアント端末上の少ない計算資源でも行えるようにすることで，システムのアーキテクチャや実装上の制約に見合ったものとしている点である．

3. ツイートのつながりに基づく抽出手法

本章では，本論文で提案する手法で中心となるツイートのつながりという概念を示したあと，その具体的な計算方法を示し，その改善のための制約の導入について述べる．

3.1 提案手法の概要

本論文では，ツイートに含まれる語どうしのつながりに着目し，それらの語のつながりに基づいてツイートをまとめることで，個別に細分化された大量のツイートの流れに対する「まとめ処理」を効果的に処理する手法を実現する．また，その手法の実現にあたり，その処理が実装上の制約に見合ったものとなるようにする．

この手法について，本節で提案を行う．そのために重要となる概念の 1 つが，ツイート間の「つながり」である．本論文中で扱う「ツイートのつながり」では，複数のツイートの間に何らかの文脈的なつながりがあったときに，そのツイート間の文脈的なつながりの持つ意味内容そのものには着目せず，その文脈的なつながりが存在するか否かに着目する．

Twitter でよく起きる現象として，まったく同じ内容のツイートを書こうとしていても，その発信者やその場の状況によって，その書き方が異なったり，複数のツイートに分割されたりすることがあげられる．たとえば，「うちの猫が餌欲しいって鳴いてる，今無いから買ってくるか」といった内容を，2 つのツイートに分けて投稿する発信者もいる．ここで，それらのツイートに共起する単語がない場合でも，同一発信者によって投稿されたツイートが，時系列的に連続したことについて触れているのであれば，「あるツイートに含まれる複数の単語は，別のツイートでも同様に共起しているか，またはその出現するツイート間の投稿時刻の差は小さくなる」と仮説が立てられる．

また，「○○始まった」，「○○終わった」のように，特定の単語を含むツイートが，一定の間隔で投稿される場合，それらは，時系列的には遠いが，内容としてはつながりを持つと考えられる．このように，一定の間隔で投稿されたツイート間がつながりを持つとすると，「それらのツイートに含まれる一部の単語は，ある一定間隔で出現する」という仮説が立てられる．

我々は，この 2 つの仮説に基づき，単語間のつながりの度合い（以下，関連度）を，共起情報とタイムスタンプに基づいて算出し，それをツイートのつながりを検出するために用いる．なお，各ツイートに含まれる単語は，それぞれ，そのツイートの持つタイムスタンプと，そのツイートを投稿した発信者情報を持つこととする．

3.2 出現単語間の関連度計算

3.1 節に述べたように，ツイート間のつながり抽出を実現するために，本研究ではツイートに出現する単語間のつながりに着目する．ツイートに出現する単語間のつながりを得るために，以下に示す単語間の関連度を定義し，その利用を試みる．

3.2.1 単語の共起に基づく関連度

2 つの単語，w_i と w_j の共起に基づく関連度（以下，共起関連度とする）$Cooc(w_i, w_j)$ は，Tanimoto 係数[*6]を用いて，次のように定義する．

$$Cooc(w_i, w_j) = \frac{F(w_i, w_j)}{F(w_i) + F(w_j) - F(w_i, w_j)} \qquad (1)$$

ここでの，$F(w_i, w_j)$ は，w_i と w_j の両方を含むツイートの数，$F(w_i)$ は，w_i を含むツイートの数を指す．

3.2.2 時系列的な近さに基づく関連度

次に，2 つの単語，w_i と w_j の時系列的な近さに基づく関連度（以下，時間関連度とする）$Time_{rel}(w_i, w_j)$ を求める．単語 w_i を含むツイートの集合を，$d_{w_i} = \{d_0^i, \ldots, d_n^i\}$ とし，その中から，特定の発信者 u によって書かれたツイートの集合のみを取得する関数を $get(d_{w_i}, u)$，2 つのツイート，d_p と d_q の持つタイムスタンプの差を絶対値で取得する関数を $getDiff(d_p, d_q)$ と定義する．w_i と w_j の持つタイムスタンプの差の集合 $diff(w_i, w_j)$ を，**図 2** に示すアルゴリズムのように求める．求める例としては，**図 3** のように求める．ここでは，d_{w_i} と d_{w_j} を，発信者単位で比較し，そのタイムスタンプの差のマルチ集合を求め，その差は，「分」単位で取得するものとする．

一般に，タイムスタンプの近い記事間の関連度は高い可能性があると考えられる．戸田ら [36] は，記事の持つタイムスタンプに着目し，「文書間のタイムスタンプが一定の時間離れるごとに，一定の割合で類似度が減少する」という仮定に基づき，記事間の時間類似度を以下の式のように求めている．

$$TimeWeight(t) = T_0 \times \exp\left(-\frac{0.693}{t_{1/2}}t\right) \qquad (2)$$

ここでの t は，2 つの記事間のタイムスタンプの差を指す．$t_{1/2}$ は，時間類似度が 50%になるときのタイムスタンプの差（半減期）を指す．T_0 は，タイムスタンプの差が 0 の場合の重みで，戸田らは 1 とした．T_0 は，タイムスタンプの

[*6] 本論文では，本手法の検討段階において実装上および精度上最も扱いやすかったことから，この係数を用いた．以降で示す関連度との組合せに対して最適な単語共起関連度の計算方法の検討は，今後の課題である．

```
Require: user, d_{w_i}, d_{w_j}
Ensure: diff は可変長配列
  count ← 0
  for i = 0 to size(user) do
    u ← user[i]
    d1 ←get(d_{w_i}, u)
    d2 ←get(d_{w_j}, u)
    for j = 0 to size(d1) do
      for k = 0 to size(d2) do
        difference ← getDiff(d1[j], d2[k])
        if difference > 0 then
          diff[count] ← difference
          count ← count + 1
        end if
      end for
    end for
  end for
```

図 2　$diff(w_i, w_j)$ を求めるアルゴリズム
Fig. 2　Algorithm for calculating $diff(w_i, w_j)$.

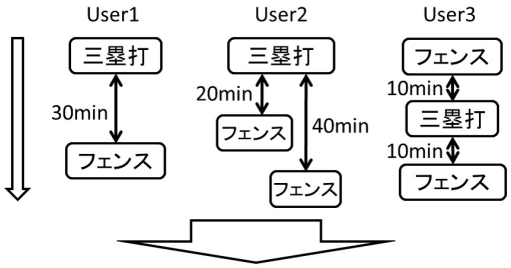


図 3　$diff(w_i, w_j)$ を求める例
Fig. 3　Example of calculating $diff(w_i, w_j)$.

差が 0 の場合に式 (2) がとる値を調整するための定数であり，本論文中では特に断りのない場合は $T_0 = 1$ とする．

我々が求めたいのは記事間の類似度ではなく，単語間の関連度であるため，上記の式を，単語間の時間関連度を求められるように，改変したものを用いる．式 (2) のように単語間の関連度を算出する場合，t を $diff(w_i, w_j)$ の平均値と定義すると，タイムスタンプ間の差の偏りが考慮されていないため，ここでは以下のように w_i と w_j の単語間の時間関連度 $Time_{rel}(w_i, w_j)$ を求める．

$$Time_{rel}(w_i, w_j) = \sum_{x=0} Time'_{rel}(C_x) \tag{3}$$

$Time'_{rel}(C_x)$ は以下のように定義する．

$$Time'_{rel}(C_x) = TimeWeight(t_x) \times SizeWeight(C_x) \tag{4}$$

ここで，C_x は，$diff(w_i, w_j)$ をある粒度にクラスタ化したものである．ここでは，図 4 のように，要素を昇順にソートした $diff(w_i, w_j)$ に含まれる，タイムスタンプの差を走査し，クラスタ化を行う．隣り合う要素の差が，閾値 k 以下の場合はそれらを同じクラスタに属させ，k より大きい場合は異なるクラスタに属させる．C_x に含まれる要素の数が 1 だった場合には，それを除外する．

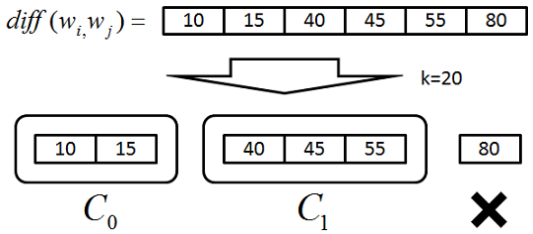


図 4　$diff(w_i, w_j)$ をクラスタ化する例
Fig. 4　Example of clustering $diff(w_i, w_j)$.

t_x は，C_x に含まれる要素の平均値を指す．$SizeWeight(C_x)$ は以下のように定義する．

$$SizeWeight(C_x) = \frac{|C_x|}{|diff(w_i, w_j)|} \tag{5}$$

ここで，$|C_x|$ は，C_x に含まれる要素の数を指し，$|diff(w_i, w_j)|$ は，$diff(w_i, w_j)$ の持つ要素の数を指す．

3.2.3　時系列上の出現間隔に基づく関連度

3.2.2 項で示した時間関連度は，単語が出現したときのタイムスタンプ間の差に着目し，その値が小さいほど，単語間の関連度が高いものとした．一方で，ある一定の間隔のものが大量にある場合，それに関しても，単語間の関連度が高いといえる可能性がある．そのため，$diff(w_i, w_j)$ を用いて，その標準偏差 $Std(w_i, w_j)$ を求める．その値を用いて，タイムスタンプ間の差の標準偏差に基づく関連度（以下，間隔関連度とする）$Time_{std}(w_i, w_j)$ を以下のように定義する．

$$Time_{std}(w_i, w_j) = \frac{1}{1 + \alpha Std(w_i, w_j)} \tag{6}$$

ここでの α は，0 より大きい任意の値である．前もって行った予備実験の結果から，期間の短いイベントの場合には値を大きくし，長いイベントの場合には小さくすることで，それぞれのイベントに即した間隔関連度を求められる[*7]．なお，ここでいうイベントとは，ツイートを収集した期間中に発生した具体的事象を意味しており，たとえば「特定の放送番組内で主人公が飛び上がった」などの事象に対応する．これらのイベントが発生したという事実は，本手法に対して事前に与えられているとは仮定しない．

3.3　出現順序を考慮した単語間の制約の利用

3.3.1　出現順序の方向性への考慮の必要性

ここまでの，時系列上に出現する単語間のタイムスタンプ差は，その絶対値を用いていた．単語 A →単語 B，単語 B →単語 A といった出現順序が異なる場合でも，それらのタイムスタンプ差の集合に属する値はすべて正となって

[*7] 予備実験から，α の値はおよそ 0.25 から 0.85 の間で設定する必要があることを確認している．

いる．

イベント内で，1 度しか起こらなかった話題に関する単語は，時系列的な近さや出現間隔に基づき単語間のつながりを抽出する際には，起点となりやすく，関連度として高い値を示す傾向が起こるのではないかと考えられる．本手法は，時間に依存する指標を用いているため，時間に依存する関連度が高い単語ペアの間には，出現順序や出現間隔の特徴を持つと考えられる．しかし，これまでのタイムスタンプの差異の絶対値を用いる方法では，その出現順序の方向性を考慮することができない．

本節では，特定の話題に関連する単語が，時間に依存する指標において，起点となる傾向があるという仮説に基づき，単語の出現順序を考慮した単語間の制約の導入を行う．

3.3.2　出現順序を考慮した単語間制約の利用方法

ここに 2 つの単語 w_i と w_j があった場合，単語の出現順序を考慮すると，以下の 3 つの特徴があると考える．

(1) 多くの発信者が，時系列上で $w_i \rightarrow w_j$ の順番にそれぞれを含むツイートを投稿している．
(2) 多くの発信者が，w_i と w_j の両方を含むツイートを投稿している．
(3) 多くの発信者が，時系列上で $w_j \rightarrow w_i$ の順番にそれぞれを含むツイートを投稿している．

そこで，w_i と w_j が，どの程度，出現順序に特徴を持っているかの指標として，以下の式のように，Tanimoto 係数を用いて T_{flow}，T_{cooc} を求める．

$$T_{flow}(w_i \rightarrow w_j) = \frac{U(w_i \rightarrow w_j)}{U(w_i) + U(w_j) - U(w_i \rightarrow w_j)} \tag{7}$$

$$T_{cooc}(w_i, w_j) = \frac{U(w_i, w_j)}{U(w_i) + U(w_j) - U(w_i, w_j)} \tag{8}$$

$U(w)$ は，w を含むツイートを投稿した発信者数を指し，$U(w_i, w_j)$ は，w_i と w_j が共起しているツイートを 1 度でも投稿した発信者数を指す．$U(w_i \rightarrow w_j)$ は，1 度でも $w_i \rightarrow w_j$ の順序で時系列上に出現させている発信者の数を指す．**図 5** に，実際に U および T_{flow}，T_{cooc} を求める例を示す．

2 つの単語 w_i，w_j において，ここで求めた $T_{flow}(w_i \rightarrow w_j)$，$T_{flow}(w_j \rightarrow w_i)$，および $T_{cooc}(w_i, w_j)$ の値が閾値 β 以上の場合，そこには，時系列的に出現順序に特徴があるとしてここでは制約として扱う．

ここで，w_i，w_j の間における $T_{flow}(w_i \rightarrow w_j)$ の値が閾値 β 以上の際に，その 2 つの単語間の共起関連度 $Cooc(w_i, w_j)$ を求めることは，制約の特徴に沿わない．同様に $T_{cooc}(w_i, w_j)$ の制約が得られた際に，時系列的な近さや出現間隔に基づいて時間関連度 $Time_{rel}(w_i, w_j)$ と，間隔関連度 $Time_{std}(w_i, w_j)$ を算出することも，制約の特徴に沿わない．また，これまでタイムスタンプの差異集合を指す $diff(w_i, w_j)$ は，差異の絶対値となっていた．しかし，$w_i \rightarrow w_j$ の出現順序に特徴を見出した場合，$w_j \rightarrow w_i$ の出現順序で登場している差異を用いるのは適さない．そこで，制約を用いる場合，$diff(w_i, w_j)$ は，$w_i \rightarrow w_j$ の出現順序で登場した際の差異集合とし，$diff(w_j, w_i)$ は，$w_j \rightarrow w_i$ の出現順序で登場した際の差異集合とする．

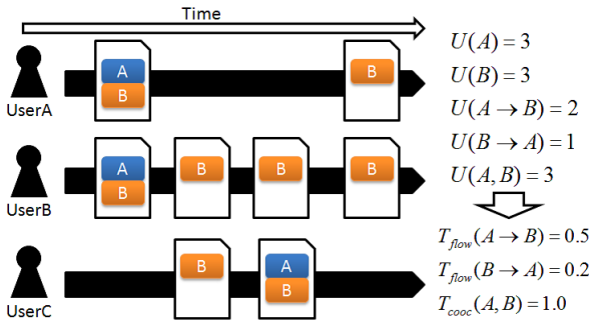


図 5　U および T_{flow}，T_{cooc} を求める例
Fig. 5　Example of calculating U, T_{flow} and T_{cooc}.

2 つの単語間関連度を求める際には，以下の式のように，まず制約に従い，$Rel_{flow}(w_i \rightarrow w_j)$，$Rel_{cooc}(w_i, w_j)$ を求める．

$$Rel_{flow}(w_i \rightarrow w_j) = \begin{cases} \max \begin{pmatrix} Time_{rel}(w_i, w_j), \\ Time_{std}(w_i, w_j) \end{pmatrix} & (T_{flow}(w_i \rightarrow w_j) \geq \beta) \\ 0 & (\text{上記以外}) \end{cases} \tag{9}$$

$$Rel_{cooc}(w_i, w_j) = \begin{cases} Cooc(w_i, w_j) & (T_{cooc}(w_i, w_j) \geq \beta) \\ 0 & (\text{上記以外}) \end{cases} \tag{10}$$

最終的に 2 つの単語間の関連度 $Rel(w_i, w_j)$ は，上で求めた 3 つの関連度を用いて，以下の式のように定める．

$$Rel(w_i, w_j) = \max \begin{pmatrix} Rel_{cooc}(w_i, w_j), \\ Rel_{flow}(w_i \rightarrow w_j), \\ Rel_{flow}(w_j \rightarrow w_i) \end{pmatrix} \tag{11}$$

3.4　単語クラスタに基づくツイートのまとまり抽出

本節では，前節までに求めた単語間の関連度に基づき，ツイートをまとめる手法を述べる．本論文では，求めた単語間関連度に基づき，イベントに関する単語が集められた単語クラスタを生成し，その単語クラスタに対して，ツイートを関連付けることで，ツイートのまとまりを構成する．なお，本論文では，ツイートのまとまり抽出の適用対象となるツイート群に対応して，野球の試合や TV 番組の放送などといった「大きな 1 つのイベント」が存在することを仮定する．これらの「大きなイベント」のことを，本論文中ではツイートを収集する際の検索キーワードになぞ

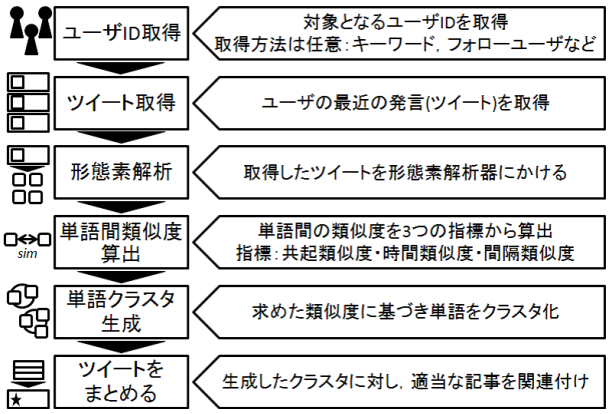


図 6　ツイートをまとめるまでの流れ
Fig. 6　Flow of our method.

らえて，便宜的に「キーワード」という用語で表現することにする．

3.4.1　単語クラスタとツイート群の関連付け

単語のクラスタリング方法は，ここでは排他的階層型クラスタリング [35] を採用する．この際，複数のクラスタが得られる．これ以降の単語クラスタとは，複数得られたクラスタの中で最もキーワードと関連の強いクラスタのことを指す．単語クラスタは，全単語クラスタのうち，サイズが一番大きく，かつそのクラスタに属する単語数が 10 以上であることを条件とした．もし，上記の条件をどのクラスタも満たさなかった場合には，適したクラスタが存在しないとし，抽出しない．

その後，単語のつながりを考慮し算出された関連度を用いて生成された単語クラスタに対して，各ユーザのタイムラインに表示されるツイートを関連付ける．最終的な本手法の処理の流れを**図 6** に示す．本手法を適用したシステムを，スマートフォンなどの小型携帯端末に組み込む際には，サーバ側とデータのやりとりを行い，各々のユーザのタイムライン上のツイートをまとめる．そのため，処理に時間とリソースを要すると考えられる図 6 の上位 5 つのプロセスは，すべてサーバ側で行い，小型携帯端末側では，その結果と自身のタイムラインを用いてツイートをまとめる処理を行う．

単語クラスタに対してツイートを関連付ける方法として，ツイートが単語セットに属する単語を持つか持たないかで判定する方法，および，前もって単語に付与されたスコアの合計値で判定する方法の 2 通りが考えられる．

前者は，単語クラスタの持つ単語を最低 x 個以上持つツイートを，その単語クラスタに関連付ける方法である．この方法は，計算量も少なく，処理時間が短く済むという特徴がある．単語クラスタに属する単語群の品質が高くないと，x の値を小さくした場合にはノイズとなるツイートの多くが，その単語セットに属されてしまう可能性がある．また，x の値を大きくすると文字数制限のある Twitter の記事に対しては，とりこぼしが多くなってしまう危険性もある．

後者は，単語クラスタの持つ単語のスコアの合計値が y 以上のツイートを，その単語クラスタに関連付ける方法である．この方法は，前者の方法に比べると精度の向上が望め，前もってスコアリング計算をサーバ側で終えている場合には，処理時間は前者の方法とほぼ同等であると予想される．一方で，採用するスコアリング方法について考察する必要がある．

今回採用する単語クラスタにツイートを関連付ける方法は，単語クラスタに属する単語の影響が大きい．本手法では単語クラスタを生成する段階で，イベントに強く関係する単語だけでなく，一般的な単語も用いて単語間のつながりを発見しているため，単語クラスタの中には，イベントとの関連度が低い一般的な単語も含まれる．前者の方法のみでは，一般的な単語を持つイベントと関連しないツイートも，単語クラスタと関連付けられてしまう恐れがある．

そのため，本手法では，前もって単語にスコアを付与する後者の方法を用いる．このスコアは，イベントのキーワードに対して関連が強いほど高くなり，関連が弱く，キーワードと関連しない一般的な単語であるほど低く値を示す必要がある．

3.5　単語のスコアリング方法

イベントと関連が強い単語ほど，多くの発信者がそれに関して言及するという仮定に基づいて，発信者群 U のうち，単語 w_i を含むツイートを 1 度でも投稿した発信者の出現回数 $U(w_i)$ を用いて，発信者群 U 中における w_i の出現頻度を指す $UF(w_i)$ を求める．

$$UF(w_i) = \frac{U(w_i)}{|U|} \tag{12}$$

ここでの $|U|$ は，全発信者数である．ただし，この式は，多くの発信者によってつぶやかれた単語をスコアとして上位に配置させることは可能だが，一般語であるか否かが考慮されておらず，単語クラスタとツイートを関連付ける際には何らかの補助が必要であると考えられる．

そのため，本手法では，単語が一般語か否かを判定するために Yahoo!の提供する Web API [44] のテキスト解析の 1 つである，キーフレーズ抽出を用いて関連度を求める．Yahoo! Web API のキーフレーズ抽出とは，日本語文を解析し，特徴的な表現をキーフレーズとして抽出し，その結果をスコアとして返してくれるサービスである．結果のスコアは最大で 100，最小で 0 となる．単語 w_i のスコアの値を 100 で割り，正規化した値を $YS(w_i)$ とする．

最終的に単語クラスタ C 内における単語 w_i のスコア $score(C, w_i)$ を以下のように求める．

$$score(C, w_i) = \frac{UF'(w_i) + YS(w_i)}{2} \tag{13}$$

$UF'(w_i)$ は，求めた $UF(w_i)$ の中で最大値が 1.0 になるように正規化した値を指す．ツイートの単語と，単語クラスタに属する単語が同一であったら，そのツイートに対してスコアを加算していき，閾値 γ 以上なら関係があるとし，単語クラスタに対し，ツイートを関連付ける．これをタイムライン上のツイート群に適用することで，話題ごとにまとまったツイート群を抽出できると考える．

4. 評価

提案手法が前提とする制約条件は，2.3 節で述べたとおり，他の多くの手法が前提とするものとは大きく異なるため，提案手法を文献 [32] や [4] などの既存の研究と直接比較をした場合でも，それら既存の手法に対して適切な条件での比較とすることが困難である．そこで，本章では，特に複数種の関連度計算方式の組合せを導入したことの効果と，手法に与えるパラメータの影響の解析を中心とした評価を行う．

4.1 実験条件

本節では，本論文の評価実験における共通の実験条件について述べる．本評価実験では，2010 年 9 月 29 日 00:00〜23:59 の間に日本人のユーザによって Twitter に投稿された，合計 11,392,095 件のツイートを用いた．

このツイート集合に対して，以下の手順に従って，対象となるツイート群を選択した．

(1) 任意のキーワード kw を含むツイートを期間 T の間に投稿したユーザが対象
(2) 各々のユーザの，期間 T に投稿されたツイートを取得

ここでの，T の値は，本論文では，キーワードに関連するイベントの発生する時間帯が前もって判明しているものとし，その時間帯と，その前後 15 分間を T と設定した．

ツイート群を抽出する際のキーワードとして，「けいおん」と「阪神」を本実験では用いた．同日は，テレビアニメ「けいおん!!」の最終回放映日であり，1 時 25 分から 1 時 55 分の間に放映された．また，同日には，プロ野球の試合として，「阪神 対 巨人」がナイトゲームで行われた．「けいおん」は，約 30 分間のテレビアニメであるため，ツイートを取得する期間 T を，1 時 10 分から 2 時 10 分までとした．ツイートを取得した結果，ユーザ 959 名，合計 41,692 件が対象となった．「阪神」に関するプロ野球の試合は，18 時からおよそ 3 時間にわたり試合が行われたため，期間 T を 17 時 45 分から 21 時 15 分までと設定し，ユーザ 629 名，合計 26,973 件のツイートを取得した．

記事の形態素解析には MeCab [22] を用いた．URL とハッシュタグは形態素解析を行う前に各ツイートから取り除いた．MeCab によって各ツイートに対して形態素解析を行った結果から，名詞・動詞・形容詞を抽出し，単語の出現順序と出現回数をツイート情報として用いた．ここで，用いた単語には，非自立語，接尾語，代名詞を除くものとし，また「する」,「なる」といった文書中で頻出する単語や，記号のみで構成されている単語も除いた．また，ひらがな，またはカタカナ 1 字で抽出された単語や，発信者に言及された回数が全発信者数の 3%未満の単語も，今回は対象から除外した．

以降では，4.2 節で，単語間の関連度を実際に 3 つの指標から求め，それに制約を付与した場合としない場合の結果の考察を行う．次に，4.3 節で，その単語間の関連度を用いて単語クラスタリングを行い，キーワードに関連する単語群が集約されるか否かを確認する．

4.2 単語間の関連度算出手法の評価

提案手法の単語間関連度の算出方法を適用した結果を**表 1** と**表 2**，および**表 5** と**表 6** に示す．表 1 と表 2 は，出現順序を考慮せずに関連度を求めた結果，表 5 と表 6 は，出現順序を考慮した制約を付与し，関連度を求めた結果を示す．単語ペアの右の数値はその単語間の関連度を指す．表中の $Cooc$ は，共起関連度によって求めた結果，$Time_{rel}$ は，時間関連度によって求めた結果，$Time_{std}$ は，間隔関連度によって求めた結果であり，それぞれ上位 30 位まで示している．なお，式 (2) におけるパラメータを，$T_0 = 1$ と $t_{1/2} = 3$ とし，$diff(w_i, w_j)$ をクラスタ化する際の閾値 k は 3 とした．式 (6) における，係数 α は，「けいおん」の際は 0.85，「阪神」の際は 0.55 とした．それぞれの表において，特にキーワードと関連が強い単語は太字とした．関連の強い単語には，単一では意味の把握が困難な単語も含まれる．それらは，単語間の共起関係などを見て，熟語に該当するものや，2 つの単語を見ることでキーワードと関連していると判断できるものは，関連が強い単語としている．

また，**表 3** と**表 4**，および**表 7** と**表 8** は，共起関連度のみ，共起関連度＋時間関連度，共起関連度+時間関連度＋間隔関連度の各組合せによる単語間の関連度を求めた結果であり，それぞれ上位 30 位まで示している．各指標を組合せた場合の単語間の関連度は，その中で最大となる関連度の値とした．

4.2.1 出現順序による制約を考慮しない場合

共起関連度は，「けいおん」の場合は，キーワードと関連する，ほぼ熟語に近い単語ペアが上位にきている．時間関連度，および間隔関連度を見ると，ほぼすべてがキーワード「けいおん」に関連したものがきている．特に「衣装」という単語が，時間関連度，間隔関連度において多くが上位にきているため，この単語が，話題の中での局所的な話題の変化を見る際には，重要な単語であることが予測できる．つまり，イベントの発生している期間に「単語 A」→「衣装」→「単語 B」という流れがあったことが，時間関連度および間隔関連度の面から想像できる．実際にアニメの内容と比較すると，アニメの開始から約 12 分後に，衣

表 1　「けいおん」で求めた単語間の関連度の各上位
Table 1　Top 30 rank of relationship between words in "K-ON".

Rank	*Cooc*		$Time_{rel}$		$Time_{std}$	
1	京-アニ	0.930	心霊-衣装	0.794	心霊-衣装	0.920
2	けい-おん	0.916	留年-げろ	0.785	衣装-勝つ	0.920
3	卒-アル	0.883	逃げる-留年	0.759	心霊-留年	0.920
4	黒い-下着	0.763	律-挟む	0.729	衣装-留年	0.920
5	**YOU-THANK**	0.745	黒い-デスデビル	0.713	逃げる-衣装	0.920
6	番外-編	0.732	パート-衣装	0.712	黒い-衣装	0.893
7	宮崎-あおい	0.615	和-みかん	0.707	留年-げろ	0.860
8	最終-回	0.595	大事-訪問	0.705	逃げる-勝つ	0.860
9	映画-化	0.595	ばあちゃん-はっさく	0.704	光る-留年	0.850
10	腐る-男子	0.537	和-光る	0.700	風邪-衣装	0.842
11	次回-予告	0.520	留年-デスデビル	0.698	逃げる-心霊	0.833
12	チェック-AMAZON	0.517	和-ばあちゃん	0.693	下着-光る	0.826
13	劇場-版	0.512	映画-トップページ	0.691	黒い-光る	0.823
14	衣装-昔	0.476	**ED**-遠慮	0.691	心霊-デスデビル	0.813
15	放課後-ティー	0.462	前髪-はっさく	0.680	黒い-勝つ	0.809
16	ニコ-視聴	0.458	和-同級生	0.670	逃げる-下着	0.803
17	かな-恵	0.456	訪問-浮く	0.666	黒い-間違う	0.785
18	**NO-THANK**	0.455	映画-実写	0.650	犬-下着	0.774
19	クラスタ-特撮	0.453	黒い-音部	0.649	昔-勝つ	0.762
20	今期-最強	0.447	黒い-料理	0.648	間違う-下着	0.762
21	前売り-券	0.429	心霊-昔	0.648	心霊-キャラソン	0.751
22	**YOU-NO**	0.420	浮く-前髪	0.647	心霊-ばあちゃん	0.744
23	タイム-ティー	0.410	映画-前売り	0.646	黒い-浮く	0.744
24	リブ-欄	0.391	下着-料理	0.646	心霊-100	0.742
25	飛ぶ-遅れる	0.388	映画-フィルム	0.640	黒い-ふんする	0.741
26	入部-員	0.380	のる-思い出	0.639	黒い-逃げる	0.740
27	ニコ-生	0.371	大事-前髪	0.639	下着-浮く	0.740
28	さわ-ちゃん	0.361	みかん-前髪	0.638	和-ムツゴロウ	0.734
29	限定-初回	0.340	のる-遠慮	0.637	ばあちゃん-留年	0.732
30	生徒-会	0.308	和-前髪	0.635	逃げる-みかん	0.729

表 2　「阪神」で求めた単語間の関連度の各上位
Table 2　Top 30 rank of relationship between words in "Hanshin".

Rank	*Cooc*		$Time_{rel}$		$Time_{std}$	
1	由-伸	0.892	登場-神	0.651	盗塁-代打	0.910
2	けい-おん	0.846	フォロー-話す	0.560	外れる-おかしい	0.773
3	ビデオ-判定	0.641	ラッキー-行き	0.556	期待-不調	0.702
4	ブラウン-解任	0.500	始まる-平野	0.550	三塁打-代走	0.680
5	可能-性	0.442	楽しい-大学	0.534	成功-大道	0.676
6	犠牲-フライ	0.394	写真-ほう	0.514	代走-大道	0.653
7	実況-神	0.349	でる-AKB	0.504	待機-俊介	0.638
8	速報-緊急	0.333	うち-家族	0.501	線-乗る	0.626
9	速報-地震	0.328	成功-俊介	0.479	三塁打-俊介	0.616
10	AKB-48	0.325	ORZ-ラミレス	0.474	盗塁-大道	0.604
11	メンバー-頃	0.319	ベース-当たる	0.471	発-俊介	0.598
12	勝-敗	0.295	三塁打-フェンス	0.467	代走-山口	0.596
13	ブラウン-監督	0.292	山口-桧山	0.466	犠牲-俊介	0.572
14	15-メンバー	0.292	判定-フェンス	0.465	代走-俊介	0.568
15	実況-球	0.287	球児-伸	0.462	犠牲-大道	0.552
16	監督-解任	0.280	球児-由	0.453	犠牲-由	0.535
17	番-実況	0.277	登場-桧山	0.450	粘る-由	0.527
18	地震-緊急	0.275	最後-14	0.449	粘る-伸	0.527
19	盗塁-成功	0.271	楽しい-ぶり	0.445	高橋-山口	0.519
20	表-死	0.263	手-松本	0.445	山口-俊介	0.508
21	風邪-ひく	0.261	ブラウン-アレ	0.441	俊介-桧山	0.508
22	鳴る-緊急	0.258	うい-名前	0.441	怖い-ウチ	0.503
23	携帯-1	0.257	下さる-勝	0.441	伸-桧山	0.502
24	楽天-解任	0.254	ファン-人気	0.441	由-大道	0.489
25	くい-ぼる	0.252	見せる-嬉しい	0.441	伸-大道	0.489
26	楽天-ブラウン	0.252	聞く-反応	0.431	城島-発	0.470
27	番-神	0.250	監督-姿	0.430	由-桧山	0.467
28	めし-えり	0.250	ビデオ-判定	0.429	知る-出場	0.463
29	神-球	0.248	真弓-位	0.428	高橋-桧山	0.454
30	15-頃	0.239	ベース-フェンス	0.427	監督-姿	0.453

装に関連する話題について言及されている．その 1 分後に「心霊写真」に関する話題が発生し，6 分後に「留年」に関する話題が発生している．また，それぞれの話題がアニメ中で触れられたのはここだけである．これらのことから，

表 3　「けいおん」で求めた各関連度の組み合わせによる単語間の関連度の各上位
Table 3　Top 30 rank of relationship combination in "K-ON".

Rank	*Cooc*		$Cooc + Time_{rel}$		$Cooc + Time_{rel} + Time_{std}$	
1	京-アニ	0.930	京-アニ	0.930	京-アニ	0.930
2	けい-おん	0.916	けい-おん	0.916	心霊-衣装	0.920
3	卒-アル	0.883	卒-アル	0.883	衣装-勝つ	0.920
4	黒い-下着	0.763	心霊-衣装	0.794	心霊-留年	0.920
5	**YOU-THANK**	0.745	留年-げろ	0.785	衣装-留年	0.920
6	番外-編	0.732	黒い-下着	0.763	逃げる-衣装	0.920
7	宮崎-あおい	0.615	逃げる-留年	0.759	けい-おん	0.916
8	最終-回	0.595	**YOU-THANK**	0.745	黒い-衣装	0.893
9	映画-化	0.595	番外-編	0.732	卒-アル	0.883
10	腐る-男子	0.537	律-挟む	0.729	留年-げろ	0.860
11	次回-予告	0.520	黒い-デスデビル	0.713	逃げる-勝つ	0.860
12	チェック-AMAZON	0.517	パート-衣装	0.712	光る-留年	0.850
13	劇場-版	0.512	和-みかん	0.707	風邪-衣装	0.842
14	衣装-昔	0.476	大事-訪問	0.705	逃げる-心霊	0.833
15	放課後-ティー	0.462	ばあちゃん-はっさく	0.704	下着-光る	0.826
16	ニコ-視聴	0.458	和-光る	0.700	黒い-光る	0.823
17	かな-恵	0.456	留年-デスデビル	0.698	心霊-デスデビル	0.813
18	**NO-THANK**	0.455	和-ばあちゃん	0.693	黒い-勝つ	0.809
19	クラスタ-特撮	0.453	映画-トップページ	0.691	逃げる-下着	0.803
20	今期-最強	0.447	**ED**-遠慮	0.691	心霊-衣装	0.794
21	前売り-券	0.429	前髪-はっさく	0.680	黒い-間違う	0.785
22	**YOU-NO**	0.420	和-同級生	0.670	留年-げろ	0.785
23	タイム-ティー	0.410	訪問-浮く	0.666	犬-下着	0.774
24	リブ-欄	0.391	映画-実写	0.650	黒い-下着	0.763
25	飛ぶ-遅れる	0.388	黒い-音部	0.649	昔-勝つ	0.762
26	入部-員	0.380	黒い-料理	0.648	間違う-下着	0.762
27	ニコ-生	0.371	心霊-昔	0.648	逃げる-留年	0.759
28	さわ-ちゃん	0.361	浮く-前髪	0.647	心霊-キャラソン	0.751
29	限定-初回	0.340	映画-前売り	0.646	**YOU-THANK**	0.745
30	生徒-会	0.308	下着-料理	0.646	心霊-ばあちゃん	0.744

表 4　「阪神」で求めた各関連度の組み合わせによる単語間の関連度の各上位
Table 4　Top 30 rank of relationship combination in "Hanshin".

Rank	*Cooc*		$Cooc + Time_{rel}$		$Cooc + Time_{rel} + Time_{std}$	
1	由-伸	0.892	由-伸	0.892	盗塁-代打	0.910
2	けい-おん	0.846	けい-おん	0.846	由-伸	0.892
3	ビデオ-判定	0.641	登場-神	0.651	けい-おん	0.846
4	ブラウン-解任	0.500	ビデオ-判定	0.641	外れる-おかしい	0.773
5	可能-性	0.442	フォロー-話す	0.560	期待-不調	0.702
6	犠牲-フライ	0.394	ラッキー-行き	0.556	三塁打-代走	0.680
7	実況-神	0.349	始まる-平野	0.550	成功-大道	0.676
8	速報-緊急	0.333	楽しい-大学	0.534	代走-大道	0.653
9	速報-地震	0.328	写真-ほう	0.514	登場-神	0.651
10	AKB-48	0.325	でる-AKB	0.504	ビデオ-判定	0.641
11	メンバー-頃	0.319	うち-家族	0.501	待機-俊介	0.638
12	勝-敗	0.295	ブラウン-解任	0.500	線-乗る	0.626
13	ブラウン-監督	0.292	成功-俊介	0.479	三塁打-俊介	0.616
14	15-メンバー	0.292	ORZ-ラミレス	0.474	盗塁-大道	0.604
15	実況-球	0.287	ベース-当たる	0.471	発-俊介	0.598
16	監督-解任	0.280	三塁打-フェンス	0.467	代走-山口	0.596
17	番-実況	0.277	山口-桧山	0.466	犠牲-俊介	0.572
18	地震-緊急	0.275	判定-フェンス	0.465	代走-俊介	0.568
19	盗塁-成功	0.271	球児-伸	0.462	フォロー-話す	0.560
20	表-死	0.263	球児-由	0.453	ラッキー-行き	0.556
21	風邪-ひく	0.261	登場-桧山	0.450	犠牲-大道	0.552
22	鳴る-緊急	0.258	最後-14	0.449	始まる-平野	0.550
23	携帯-1	0.257	楽しい-ぶり	0.445	犠牲-由	0.535
24	楽天-解任	0.254	手-松本	0.445	楽しい-大学	0.534
25	くい-ぼる	0.252	可能-性	0.442	粘る-由	0.527
26	楽天-ブラウン	0.252	ブラウン-アレ	0.441	粘る-伸	0.527
27	番-神	0.250	うい-名前	0.441	高橋-山口	0.519
28	めし-えり	0.250	下さる-勝	0.441	写真-ほう	0.514
29	神-球	0.248	ファン-人気	0.441	山口-俊介	0.508
30	15-頃	0.239	見せる-嬉しい	0.441	俊介-桧山	0.508

「衣装」と「心霊」の時間関連度，間隔関連度はともに非常に高くなっている．今回の予備実験での半減期は 3 分であるため，3 分以上離れて発生した話題に関連する単語間の時間関連度は低くなる．「衣装」と「留年」は，6 分間話題が離れているため時間関連度では上位にこず，間隔関連度では，高い位置にきている．表中には載っていないが，「衣装」と「留年」間の時間関連度は 0.25 となっていた．

表 5　「けいおん」で求めた単語間の関連度の各上位（制約適用後）
Table 5　Top 30 rank of relationship between words in “K-ON” (Using constraints).

Rank	*Cooc*		$Time_{dif}(w_i \to w_j)$		$Time_{std}(w_i \to w_j)$	
1	京-アニ	0.930	衣装→平沢	1.000	光る→衣装	0.960
2	けい-おん	0.916	大事→光る	0.972	光る→昔	0.958
3	卒-アル	0.883	心霊→犯人	0.968	衣装→遅れる	0.958
4	黒い-下着	0.763	浮く→光る	0.933	昔→遅れる	0.957
5	**YOU-THANK**	0.745	訪問→光る	0.923	衣装→飛ぶ	0.956
6	編-番外	0.732	衣装→子	0.904	昔→飛ぶ	0.956
7	最終-回	0.595	黒い→デスデビル	0.898	衣装→平沢	0.954
8	映画-化	0.595	パート→衣装	0.875	間違う→逃げる	0.951
9	次回-予告	0.520	心霊→オーラ	0.857	衣装→逃げる	0.949
10	劇場-版	0.512	パート→昔	0.841	衣装→合	0.949
11	衣装-昔	0.476	のる→遅れる	0.833	光る→間違う	0.948
12	**NO-THANK**	0.455	下着→デスデビル	0.822	光る→ばあちゃん	0.942
13	**YOU-NO**	0.420	衣装→心霊	0.821	衣装→勝つ	0.940
14	飛ぶ-遅れる	0.388	留年→げろ	0.813	光る→つむ	0.940
15	さわ-ちゃん	0.361	おでこ→ばあちゃん	0.794	ばあちゃん→逃げる	0.935
16	生徒-会	0.308	光る→ばあちゃん	0.794	風邪→衣装	0.934
17	写真-心霊	0.306	はっさく→ばあちゃん	0.794	衣装→遠慮	0.934
18	番外-訪問	0.284	光る→ 100	0.794	留年→げろ	0.931
19	タイム-放課後	0.279	はっさく→前髪	0.794	おでこ→ばあちゃん	0.929
20	回収-伏線	0.257	大事→子供	0.794	大事→留年	0.925
21	おでこ-光る	0.240	遅れる→遠慮	0.794	遅れる→遠慮	0.923
22	絵-ジャケ	0.238	みかん→光る	0.792	おでこ→衣装	0.923
23	編-訪問	0.232	みかん→訪問	0.791	光る→心霊	0.922
24	卒業-アルバム	0.219	光る→変	0.771	光る→ムツゴロウ	0.920
25	分-あと	0.211	黒い→留年	0.769	ばあちゃん→留年	0.917
26	変-前髪	0.203	訪問→前髪	0.768	衣装→留年	0.916
27	年-100	0.201	浮く→前髪	0.761	下着→遅れる	0.913
28	写-ジャケ	0.188	昔→心霊	0.760	黒い→遅れる	0.910
29	先生-花田	0.185	心霊→勝つ	0.760	ムツゴロウ→留年	0.910
30	純-憂	0.182	黒い→音部	0.758	犬→勝つ	0.909

表 6　「阪神」で求めた単語間の関連度の各上位（制約適用後）
Table 6　Top 30 rank of relationship between words in “Hanshin” (Using constraints).

Rank	*Cooc*		$Time_{dif}(w_i \to w_j)$		$Time_{std}(w_i \to w_j)$	
1	由-伸	0.893	桧山→山口	0.614	エース→スタンド	0.992
2	けい-おん	0.846	犠牲→死球	0.614	エース→春	0.991
3	ビデオ-判定	0.642	発売→開始	0.545	粘る→弾	0.991
4	ブラウン-解任	0.500	ビデオ→三塁打	0.545	六→スタンド	0.990
5	可能-性	0.443	判定→三塁打	0.541	代走→盗塁	0.910
6	犠牲-フライ	0.394	フェンス→三塁打	0.538	弾→杯	0.789
7	緊急-速報	0.333	盗塁→俊介	0.537	同点→犠牲	0.702
8	地震-速報	0.329	犠牲→平野	0.509	三塁打→代走	0.679
9	48-AKB	0.326	判定→フェンス	0.505	代走→バント	0.678
10	勝-敗	0.295	同点→犠牲	0.481	三塁打→大道	0.604
11	ブラウン-監督	0.293	ビデオ→塁打	0.471	フェンス→三塁打	0.600
12	監督-解任	0.281	マートン→犠牲	0.471	盗塁→俊介	0.598
13	緊急-地震	0.276	風邪→ひく	0.469	代走→山口	0.596
14	盗塁-成功	0.271	ビデオ→判定	0.458	犠牲→山口	0.583
15	風邪-ひく	0.262	ホームラン→三塁打	0.449	ラッキー→俊介	0.576
16	緊急-鳴る	0.258	三塁打→犠牲	0.444	犠牲→俊介	0.572
17	楽天-解任	0.255	判定→塁打	0.440	代走→俊介	0.568
18	くい-ぼる	0.253	逆転→判定	0.439	犠牲→死球	0.561
19	楽天-ブラウン	0.252	逆転→ビデオ	0.438	犠牲→大道	0.552
20	めし-えり	0.250	ホームラン→判定	0.430	由→粘る	0.527
21	とら-ほる	0.236	ただ→いま	0.416	伸→粘る	0.527
22	映画-化	0.233	判定→ビデオ	0.409	山口→脇谷	0.525
23	地震-鳴る	0.219	少ない→ 40	0.397	桧山→山口	0.525
24	古城-死球	0.213	発売→展開	0.382	山口→高橋	0.519
25	待機-とら	0.212	山口→桧山	0.378	盗塁→山口	0.516
26	巨人-阪神	0.209	性→歳	0.366	大道→由	0.489
27	母-家族	0.205	ガッツ→ラミ	0.360	大道→伸	0.489
28	楽天-監督	0.204	こっち→個人	0.354	桧山→高橋	0.483
29	たん-ぼる	0.197	由→伸	0.347	判定→ビデオ	0.473
30	10-月	0.197	代走→バント	0.337	桧山→由	0.467

「阪神」に関する結果では，共起関連度では，キーワードと関係のあまりない単語ペアが上位にきている．これは，9 月 29 日起きたニュースに関連するものが多く登場しており，「阪神」と関連する野球に関するニュース記事を投稿した発信者の影響を強く受けているため起きたものであると考えられる．

表 7　「けいおん」で求めた各関連度の組み合わせによる単語間の関連度の各上位（制約適用後）
Table 7　Top 30 rank of relationship conbination in “K-ON” (Using constraints).

Rank	*Cooc*		$Cooc + Time_{dif}(w_i \to w_j)$		$Cooc + Time_{dif}(w_i \to w_j)$ $+Time_{std}(w_i \to w_j)$	
1	京-アニ	0.930	衣装→平沢	1.000	衣装→平沢	1.000
2	けい-おん	0.916	大事→光る	0.972	大事→光る	0.972
3	卒-アル	0.883	心霊→犯人	0.968	心霊→犯人	0.968
4	黒い-下着	0.763	浮く→光る	0.933	光る→衣装	0.960
5	**YOU-THANK**	0.745	京-アニ	0.930	光る→昔	0.958
6	編-番外	0.732	訪問→光る	0.923	衣装→遅れる	0.958
7	最終-回	0.595	けい-おん	0.916	昔→遅れる	0.957
8	映画-化	0.595	衣装→子	0.904	衣装→飛ぶ	0.956
9	次回-予告	0.520	黒い→デスデビル	0.898	昔→飛ぶ	0.956
10	劇場-版	0.512	卒-アル	0.883	衣装→平沢	0.954
11	衣装-昔	0.476	パート→衣装	0.875	間違う→逃げる	0.951
12	**NO-THANK**	0.455	心霊→オーラ	0.857	衣装→逃げる	0.949
13	**YOU-NO**	0.420	パート→昔	0.841	衣装→合	0.949
14	飛ぶ-遅れる	0.388	のる→遅れる	0.833	光る→間違う	0.948
15	さわ-ちゃん	0.361	下着→デスデビル	0.822	光る→ばあちゃん	0.942
16	生徒-会	0.308	衣装→心霊	0.821	衣装→勝つ	0.940
17	写真-心霊	0.306	留年→げろ	0.813	光る→つむ	0.940
18	番外-訪問	0.284	おでこ→ばあちゃん	0.794	ばあちゃん→逃げる	0.935
19	タイム-放課後	0.279	光る→ばあちゃん	0.794	風邪→衣装	0.934
20	回収-伏線	0.257	はっさく→ばあちゃん	0.794	衣装→遠慮	0.934
21	おでこ-光る	0.240	光る→ 100	0.794	浮く→光る	0.933
22	絵-ジャケ	0.238	はっさく→前髪	0.794	留年→げろ	0.931
23	編-訪問	0.232	大事→子供	0.794	京-アニ	0.930
24	卒業-アルバム	0.219	遅れる→遠慮	0.794	おでこ→ばあちゃん	0.929
25	分-あと	0.211	みかん→光る	0.792	大事→留年	0.925
26	変-前髪	0.203	みかん→訪問	0.791	遅れる→遠慮	0.923
27	年-100	0.201	光る→変	0.771	おでこ→衣装	0.923
28	写-ジャケ	0.188	黒い→留年	0.769	訪問→光る	0.923
29	先生-花田	0.185	訪問→前髪	0.768	光る→心霊	0.922
30	純-憂	0.182	黒い-下着	0.763	光る→ムツゴロウ	0.920

表 8　「阪神」で求めた各関連度の組み合わせによる単語間の関連度の各上位（制約適用後）
Table 8　Top 30 rank of relationship conbination in “Hanshin” (Using constraints).

Rank	*Cooc*		$Cooc + Time_{dif}(w_i \to w_j)$		$Cooc + Time_{dif}(w_i \to w_j)$ $+Time_{std}(w_i \to w_j)$	
1	由-伸	0.893	由-伸	0.893	エース→スタンド	0.992
2	けい-おん	0.846	けい-おん	0.846	エース→春	0.991
3	ビデオ-判定	0.642	ビデオ-判定	0.642	粘る→弾	0.991
4	ブラウン-解任	0.500	桧山→山口	0.614	六→スタンド	0.990
5	可能-性	0.443	犠牲→死球	0.614	代走→盗塁	0.910
6	犠牲-フライ	0.394	発売→開始	0.545	由-伸	0.893
7	緊急-速報	0.333	ビデオ→三塁打	0.545	けい-おん	0.846
8	地震-速報	0.329	判定→三塁打	0.541	弾→杯	0.789
9	48-AKB	0.326	フェンス→三塁打	0.538	同点→犠牲	0.702
10	勝-敗	0.295	盗塁→俊介	0.537	三塁打→代走	0.679
11	ブラウン-監督	0.293	犠牲→平野	0.509	代走→バント	0.678
12	監督-解任	0.281	判定→フェンス	0.505	ビデオ-判定	0.642
13	緊急-地震	0.276	ブラウン-解任	0.500	桧山→山口	0.614
14	盗塁-成功	0.271	同点→犠牲	0.481	犠牲→死球	0.614
15	風邪-ひく	0.262	ビデオ→塁打	0.471	三塁打→大道	0.604
16	緊急-鳴る	0.258	マートン→犠牲	0.471	フェンス→三塁打	0.600
17	楽天-解任	0.255	風邪→ひく	0.469	盗塁→俊介	0.598
18	くい-ぼる	0.253	ビデオ→判定	0.458	代走→山口	0.596
19	楽天-ブラウン	0.252	ホームラン→三塁打	0.449	犠牲→山口	0.583
20	めし-えり	0.250	三塁打→犠牲	0.444	ラッキー→俊介	0.576
21	とら-ほる	0.236	可能-性	0.443	犠牲→俊介	0.572
22	映画-化	0.233	判定→塁打	0.440	代走→俊介	0.568
23	地震-鳴る	0.219	逆転→判定	0.439	犠牲→死球	0.561
24	古城-死球	0.213	逆転→ビデオ	0.438	犠牲→大道	0.552
25	待機-とら	0.212	ホームラン→判定	0.430	発売→開始	0.545
26	巨人-阪神	0.209	ただ→いま	0.416	ビデオ→三塁打	0.545
27	母-家族	0.205	判定→ビデオ	0.409	判定→三塁打	0.541
28	楽天-監督	0.204	少ない→ 40	0.397	フェンス→三塁打	0.538
29	たん-ぼる	0.197	犠牲-フライ	0.394	盗塁→俊介	0.537
30	10-月	0.197	発売→展開	0.382	由→粘る	0.527

時間関連度では，「三塁打」，「フェンス」，「判定」の 3 つの単語が時系列的に近い関係にあることが分かる．これは，「甲子園球場で行われた阪神対巨人戦において，坂選手の打球が一時本塁打と判定されたが，ビデオ判定を行った結果，フェンスに当たっていたとし，三塁打となった」という，短期間ではあるが時間の経過に依存した事実に関連す

る単語間の関連度が高くなったことによると考えられる．ビデオ判定は，結果が出るまで時間を必要とし，単一のツイートに書くのではなく，「ビデオ判定をすることになった」ということと，「判定の結果三塁打になった」という2つのことに関して投稿している発信者が多い．また，間隔関連度では，選手の名前，特に阪神対巨人戦における登板した継投や代打などで途中交代した選手の名前が多く見受けられる．投手の交代には，時系列的には距離が離れているが，その単語の持つタイムスタンプ間の差は一定のものが多いため，全体的に上位にきている傾向があった．三塁打は，この試合においては，18時50分頃のホームラン判定が覆った際にしか発生していない．また，代走による選手交代は19時55分，20時30分に起こっており，その2つの単語間に時間的関連性は見られないが，間隔に基づく関連性を見つけることができた．試合の内容を見ると，「ビデオ判定」のように1試合に1度起こるか起こらないかのような話題が発生している．その話題が一定期間発生したため，ここではそれに関連する単語の多くが，時間関連度が高くなっていることは上でも触れた．

この「ビデオ判定」は，期間中，1度しか発生しない話題に関するものであると考えられる．この「ビデオ判定」の話題に関連する単語である「ビデオ」と「判定」が間隔関連度でも頻出するのは，この「期間中1度しか発生しない話題」であることに強く関係していると考える．それらの単語が，間隔関連度を求める際に起点のように扱われ，その前後に発生した話題に関連する単語との出現間隔が比較的類似するようになり，間隔関連度が高くなっている．逆に，期間中に何度も登場するような単語は，出現間隔がばらけてしまい，それらの単語間の時系列的な類似性が一部見られたとしても，結果として算出方法によっては，関連度が低くなる傾向がある．

表3および表4は，キーワード「けいおん」および「阪神」に対して，単語間の関連度の計算として，単語共起のみに基づく場合，単語共起と時間関連度の2つの関連度の最大値をとった場合，式(11)に基づく関連度を用いた場合（単語共起，時間関連度，および間隔関連度の3つのうちの最大値を用いた場合）での，関連度上位30件をそれぞれ列挙したものである．表3を見ると，単語共起のみを用いた場合では，「けい-おん」や「京-アニ」のような，「けいおん」というキーワードそのものとの関連が強いが，必ずしもその放送回の内容を示したものではない単語が上位にきている．これらの単語は，時間関連度の追加，さらに間隔関連度の追加によって低い順位に移動しており，3つすべての関連度を用いた場合では，その番組放送回内の内容を示す単語ペアが，より上位に現れるようになったと考えられる．表4では，キーワード「阪神」と直接関連のない「けい-おん」が，3つの関連度を組み合わせた場合でも3位となってしまっているが，「ブラウン-解任」や「AKB-48」といった「阪神」の試合内容そのものと関連のないものが，低い順位に移動する傾向が見られた．以上から，3つの関連度を組み合わせることで，実際に生じた出来事（イベント）と関連の高い単語間の関連度を高くすることができていると考えられる．

4.2.2　出現順序を考慮した制約を用いた場合

表5および表6は，表1および表2に対して，それぞれ出現順序を考慮するよう制約を付与した場合の結果である．キーワード「けいおん」においては，共起関連度では，制約を付与しない結果と比較すると，キーワードと関連のある単語ペアが上位にくる傾向が見られた．時間関連度では，1つの単語に対して，出現順序が大きく偏る結果が見られる．「おでこ→おばあちゃん」，「光る→おばあちゃん」「はっさく→おばあちゃん」の組合せがそれにあたる．この組合せは時間関連度も同じ値を示しているため，「おでこ」，「光る」，「はっさく」に関連のある話題から，「おばあちゃん」に関連のある話題の変遷が読み取れる．間隔関連度では，単語「光る」が頻出している．単語間のつながりの起点となる単語が「光る」であり，そこからの話題の展開が読み取れる．

キーワード「阪神」において，共起関連度は，制約を用いない場合と比較するとほとんど違いはない．しかし，ある2つの単語を共起させたツイートを投稿した発信者数が少ないためか，制約が生成されない組合せがあり，少しだけ結果が異なる．時間関連度に関しては，出現順序を考慮したことで，単語のつながりから，話題の変遷を明確に追えることが可能になったといえる．「ビデオ→三塁打」「判定→三塁打」「ホームラン→三塁打」「判定→塁打」は特に前項で述べた話題の流れと一致する．「マートン→犠牲」といった単語ペアも，共起関連度の「犠牲-フライ」というつながりを考慮すると，マートン選手が犠牲フライを打ったことが推測可能である．間隔関連度では，0.9以上の組合せは，今回のプロ野球の試合と関係のないようなつながりが多く見られる．時間関連度よりも，選手名が多く見られる．出現順序を考慮したことで，選手の登場順序もつながりから把握可能となった．組合せの上位を見ると，「→山口」が多く見られ，「山口→」が少ないことを見ると，山口選手は試合の終盤に登場したことが推測可能である．この試合において山口選手は，ジャイアンツの最後の投手として登板している．

表7および表8は，表3および表4に対して，それぞれ出現順序を考慮するよう制約を付与した場合の結果である．4.2.1項の場合と同様に，3つの関連度を組み合わせることで，イベントと関連の高い単語間の関連度を高くすることができていると考えられる．

以上のことから，出現順序を考慮し，制約を付与することで，良好な結果が得られたと考えられる．

表 9　「けいおん」で求めた単語クラスタ（計 2 クラスタ）
Table 9　Word clusters in “K-ON”.

Size	Words
63	心霊，衣装，留年，げろ，勝つ，黒い，下着，ばあちゃん，はっさく，和，みかん，ふんする，浮く，前髪，逃げる，昔，**ED**，遠慮，ムツゴロウ，変，料理，間違う，同級生，ご飯，のる，思い出，**ED**，デスデビル，訪問，光る，編，番外，風邪，オーラ，パート，大事，律，挟む，純，おでこ，手，わかる，写真，キロ，音部，100，犬，おわる，**NO**，**THANK**，キャラソン，お前，ムギ，亀，まつげ，なくなる，家，年，部屋，はじ，合，おおう，たんこぶ
9	映画，トップページ，化，実写，前売り，フィルム，公式，画，**HP**

表 10　「阪神」で求めた単語クラスタ（計 70 クラスタ）の一部
Table 10　Word clusters in “Hanshin”.

Size	Words
35	待機，俊介，三塁打，代走，山口，高橋，桧山，希望，登場，神，盗塁，大道，四，ラン，ビデオ，**HR**，審判，マートン，判定，ホームラン，犠牲，由，球児，粘る，伸，ベース，当たる，フライ，始まる，平野，フェンス，うまい，発，城島，成功
5	楽天，うれしい，ブラウン，あれ，解任
4	楽しい，大学，ぶり，2010
4	フォロワー，物，ぼる，くい
3	緊急，鳴る，速報
3	今度，ラッキー，行く

4.3　抽出した単語クラスタの評価

本手法で求めた単語間の関連度は，2 つの単語間の関連度を，共起関係や時系列的な類似性から求めたものである．今回実験で選択したキーワードは，一定期間発生したイベントに関連するものであり，単一の単語ペアのみによって，そのイベントを表すことは困難である．一定期間発生するイベントの場合には，それに関連する単語は，「単語 A」→「単語 B」→「単語 C」のように連続して単語間がつながり，1 つのイベントを表すことが考えられる．

そのため，本節では，その単語間の関連度を用いて単語クラスタリングを行い，キーワードに関連する単語群が集約するか確認する．

4.3.1　出現順序による制約を考慮しない場合

出現順序による制約を考慮せず，階層型クラスタリングでクラスタ化した結果を**表 9** と**表 10** に示す．「けいおん」に関するクラスタリング結果については，計 2 のクラスタが，「阪神」に関するクラスタリング結果としては，計 70 のクラスタが形成された．太字の単語は，キーワードと関連が強いと考えられる単語である．

ここでは，階層型クラスタリングを用いて，関連の強い単語セットごとにクラスタ化し，キーワードと関連の強い単語セットのみを抽出することを試みた．単語間の関連度は，それぞれの指標によって求められているため，それらを統合する必要がある．ここでは，3 つの指標によって求めた関連度の平均値を，その単語間の関連度と設定した．階層型クラスタリングは，単語とクラスタ間の関連度が，閾値 l を下回った場合に，クラスタリングを止める．キーワードが「けいおん」の場合，全体的に関連度が高いために，閾値 $l = 0.35$ と設定し，「阪神」の場合には，閾値 $l = 0.15$ と設定した．それぞれのクラスタリング結果を，表 9 と表 10 に示す．

キーワード「けいおん」の結果では，1 つの巨大なクラスタが形成されているが，それに含まれる単語の多くがキーワードと強く関連したものであることを確認できる．「阪神」の結果では，プロ野球の阪神対巨人戦に関するクラスタ以外にもいくつかニュースに関したクラスタが形成されている．ブラウン監督の解任や，地震速報に関連するクラスタが見られる．ニュースとして Twitter に投稿されるツイート[*8]は，基本的に単一のツイートで完結しているため，それらに含まれる単語間の共起関連度は高くなる傾向がある．そのため，ニュースに関連したクラスタの多くは，そのニュースに強く関連する単語の組合せによって形成されている．

表 11　出現順序を考慮し「けいおん」で求めた単語クラスタ（合計 1 クラスタ）
Table 11　Word clusters in “K-ON” (Using constraints).

Size	Words
82	彼氏，心霊，音部，憂，黒い，時間，おわる，部，留年，**THANK**，**NO**，**YOU**，のる，100，飛ぶ，逃げる，変，つむ，遠慮，遅れる，げろ，キタ，決定，浮く，はじまる，オーラ，残る，前髪，ムギ，光る，花田，ごはん，ふんする，はっさく，ムツゴロウ，なくなる，律，下着，映画，化，わかる，回収，さわ，写真，デスデビル，衣装，手，勝つ，静か，来年，ぽい，失敗，平沢，子，キャラソン，部屋，撮る，犬，大事，おでこ，アル，卒，子供，風邪，合，家，訪問，思い出，昔，和，**ED**，はじ，待機，みかん，いむ，間違う，ばあちゃん，まつげ，ジャケ，犯人，最強，パート

表 12　出現順序を考慮し「阪神」で求めた単語クラスタ（合計 46 クラスタ）
Table 12　Word clusters in “Hanshin” (Using constraints).

Size	Words
66	金本，外す，スタメン，**HR**，結果，アニキ，相性，打線，凡，内野，代走，盗塁，俊介，讀賣，ラッキー，振る，ボール，関本，古城，守備，山口，藤川，脇谷，松本，高橋，投手，代打，球児，ひる，桧山，先制，由，粘る，弾，杯，打，虎，伸，抑える，犠牲，鳥谷，死球，三塁，先頭，安打，マートン，逆転，ホームラン，ビデオ，フライ，同点，塁打，三塁打，セッキー，坂，判定，フェンス，平野，レフト，勝負，大道，バント，ラジオ，鳥，打てる，タイガース
24	任天堂，ベスト，歳，初，高，夢，ぶり，イチロー，サッカー，リーグ，チーム，決まる，限り，AKB，限定，フル，東，48，宮崎，公開，22，六，偉い，スタンド，スーパー，エース，春，やばい，場，動画，笑う，立つ，千，大学，親，性，展開，発売，開始，ゲーム
4	帰宅，おく，いま，ただ
4	お金，幸せ，全部，物
2	矢野，小笠原
2	ラミレス，阿部
2	采配，打球
2	ガッツ，ラミ

4.3.2　出現順序を考慮した制約を用いた場合

表 11，**表 12** に，出現順序を考慮した制約を付与し，単語クラスタリングを行った結果を示す．制約を与える際の閾値 γ は，0.085 とした．

制約を付与しない場合と比べた場合，キーワード「けいおん」の結果では，クラスタが大きくなっているが，その中に含まれる単語を見ると，アニメ「けいおん」と関連しない単語も多い．付与しないときには別のクラスタになっていた，「けいおん」と関連の強い単語である，「映画」と

*8　本提案手法のツイートの収集方法では，キーワードと関連しないツイートも多く対象となるため，このようなケースも起こりうる．

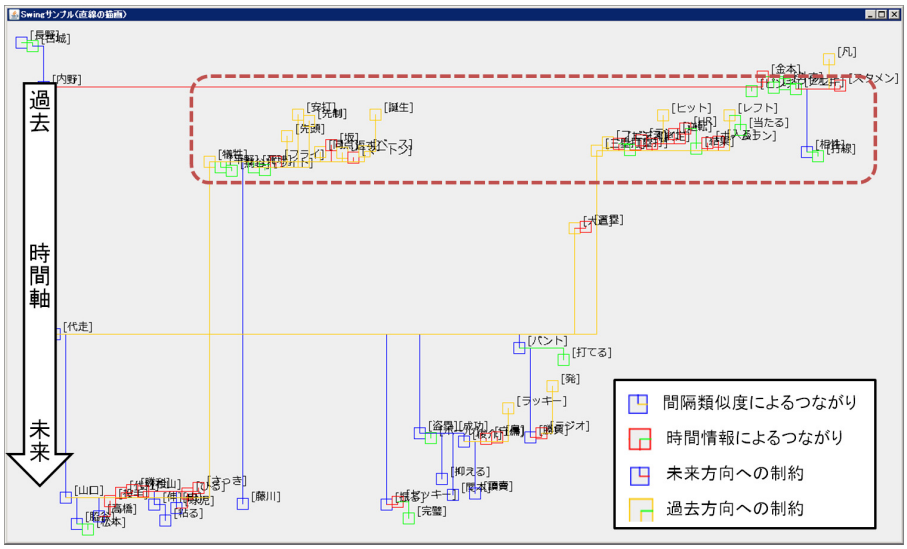


図 7　阪神に関連する単語クラスタをモデル化した例（全体）
Fig. 7　Model of word clusters in (“Hanshin”).

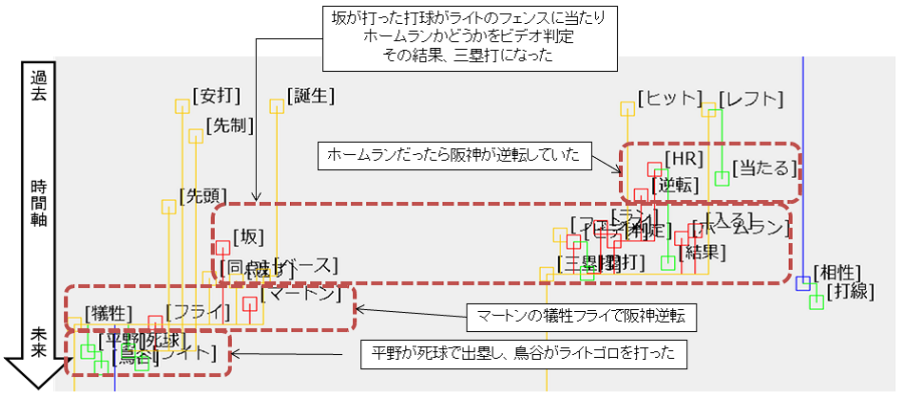


図 8　阪神に関連する単語クラスタをモデル化した例（部分拡大）
Fig. 8　Detailed model of word clusters in (“Hanshin”).

「化」が，1 つにまとめられている．関連の強い単語の個数は，制約を付与しない場合 63 個中 45 個，付与する場合 82 個中 52 個で，その割合はそれぞれ 0.71，0.63 となり，付与しない場合のほうがキーワード「けいおん」においては良い．この原因として，対象となったアニメ「けいおん」は，SF ではなく日常生活に近い内容がテーマのアニメであるため，一般的な語が増えてしまったことが原因だと考えられる．キーワードと強く関連したものであるかどうかについては，大学院修士課程の学生数名のなかから，本論文で扱った話題に関するそれぞれの事実関係を最も把握していた人物にインタビューを行い，そこで出現する単語と実際のイベントとの事実的対応関係がおおむね適切であることを確認した．

キーワード「阪神」の結果を見ると，「阪神」と関連するイベント，阪神対巨人のプロ野球の試合に関連する単語が多く集まっている．出現順序を考慮することで，それまで両方向に対してつながりの抽出を行おうとした場合と比べると，高い値として求められなかった組合せが抽出できている．出現順序を考慮せずに，両方向に対してつながりを見つけようとした場合には，出現間隔が一定になることが少なくなり，時系列的な近さや，同じ出現間隔に基づく関連度では抽出できない組合せがあった．一方で，今回のプロ野球の試合に関連する単語が別の小さなクラスタとして集まっているものも多い．

単語の出現順序を考慮し，制約を用いた場合，単語間の関係に方向性が与えられているため，クラスタリング結果からモデルを構築することが可能である．**図 7** は，制約を用いて生成した単語クラスタ群の中で，最良のクラスタと考えられるものをモデル化したものである．**図 8** は，図 7 中の赤い点線で囲まれた箇所を拡大したものである．縦軸が時間軸になっており，上から下への時間の流れがある．リンクの色は，青・黄が，間隔関連度によるつながりを示し，赤・緑が時間情報によるつながりを示している．青・赤は未来方向への制約で，黄・緑は過去方向への制約を示す．リンクの長さは，ここではノード間の関連度ではなく，タイムスタンプ差の平均値である．ノードと絶対時間は関

連付けられていないため，図中のノードはすべて相対的に配置されている．図 7 を見ると，時間関連度と間隔関連度がともにバランス良く用いられて，単語間のつながりを示していることが分かる．また，図 8 を見ると，単語どうしの話題としてのつながりも見られ，ビデオ判定の話題が発生した前後では，それに関連のある単語が多く見られる．

4.4　イベントに関連するツイート群抽出の評価

制約を用いて単語クラスタを形成させることで，イベントに関連する単語の多くを，単一の単語クラスタとして抽出した際に，その単語クラスタに属する単語にスコアを与え，イベントに関連するツイートを判定できるかどうかの評価を行う．

ここでは，キーワード「阪神」を対象として，実験を行う．前もってツイートに対し，今回対象となるイベント，阪神対巨人に関するツイートか否かを人手で答えとして付与する．合計 3 名の協力者によって，合計 100 ユーザの，全ツイート数 9,312 件に対して答えを付与した．協力者らには，今回の試合のハイライトの動画を見てもらい，試合に参加した選手や戦績の分かる試合詳細の書かれた資料を配布した．その後，ユーザごとにツイートを表示し，各ツイートごとに正解か否かを与えてもらった．そのときの判断基準は，そのツイート単体でも，今回の試合に関する内容であるものかが判読できるかどうかとした．ツイートに付与されていたハッシュタグは，協力者らのツイートに答えを付与する際の判断基準が一定とならない可能性があるため，非表示とした．最終的に，3,074 件の正解ツイートと，6,238 件の不正解ツイートに分けられた．

提案手法での設定すべきパラメータは，間隔関連度を求める際の α，制約を付与する際の閾値 β，単語クラスタリングを停止する閾値 l，および単語クラスタにツイートを関連付ける際のスコアの閾値 γ，の 4 つである．α は単語間のタイムスタンプの差異集合のぶれ幅への寛容性を示す．α が 1 に近い値になるほど間隔関連度の値が全体として小さくなり，0 に近い値にすると間隔関連度が全体的に大きな値をとるようになる．β は，2 つの単語間に出現順序，および共起に関して何かしら特徴を持つか否かを判定するための閾値である．1 に近づけると，制約がほとんど抽出できず，0 に近づけると多くの制約が抽出される．l は，単語間の関連度を用いてクラスタリングを行う際に，そのクラスタリングを止めるための閾値である．その値を 1 に近づけると，高い関連度を持つ単語どうしでしかクラスタ化されず小さなクラスタが多く得られ，0 に近づけると巨大なクラスタが少数得られる．γ は，単語クラスタにツイートを関連付ける際に算出されるクラスタとツイート間のスコアと比較し，そのスコアが γ よりも大きい場合，クラスタとツイートが関連付けられる．

ここでは，α は，予備実験などから 0.55 に固定する．β と l をそれぞれ変動させ，それぞれにおいて求められたスコアリング済みの単一の単語クラスタに対し，γ を変動させながら，精度を比較する．適合率（Precision）と再現率（Recall），および F 値（F-measure）を求め，その結果から適したパラメータを推測する．ただし，本手法をシステムに適用した際に，あるツイートに関連のある話題が単語クラスタとの関連からはうまく見つけられなかった場合よりも，ある単語クラスタとまったく関連を持たないツイートがその単語クラスタと関連付けられる場合のほうが，そのツイートを見逃す可能性があるという観点で影響が大きいと考えられるため，ここでは特に，単語クラスタにどれだけ正しくツイートが割り当てられたかについて，適合率に着目して評価を行う．

表 13　パラメータの組合せ（F 値の高い順）
Table 13　Top 10 parameter sets (F-measure).

β	l	γ	Size	Precision	Recall	F-measure
0.080	0.07	0.3	2349	0.832	0.636	**0.721**
0.080	0.07	0.2	2806	0.754	0.688	0.720
0.085	0.06	0.3	2345	0.820	0.626	0.710
0.085	0.06	0.2	2649	0.760	0.655	0.703
0.085	0.05	0.3	2719	0.747	0.661	0.701
0.090	0.05	0.3	2439	0.792	0.629	0.701
0.080	0.07	0.1	3177	0.688	**0.711**	0.700
0.095	0.05	0.2	2369	0.799	0.616	0.696
0.095	0.05	0.3	2191	**0.833**	0.594	0.693
0.095	0.05	0.1	2530	0.767	0.631	0.692

4.4.1　実験結果の考察

α を 0.55 で固定し，β を 0.025 から 0.175 まで 0.005 刻みで変動させ，l をそれぞれの β の値に対して，0.05 から 0.85 まで 0.01 刻みで変動させ，それぞれにおいて，単語クラスタを 1 つ自動で抽出した．抽出される単語クラスタは，全単語クラスタのうち，サイズが一番大きく，かつそのクラスタに属する単語数が 10 以上であることを条件とした．その単語クラスタに対して，スコアの閾値 γ を 0.1 から 0.9 まで 0.1 刻みで変動させ，対象となる 9,312 件のツイートを関連付ける．その際の適合率，再現率，F 値を求め，考察を行う．

F 値の最も高いパラメータの組合せは，$\beta = 0.08$，$l = 0.07$，$\gamma = 0.3$ であった．その際に，単語クラスタに対して，2,349 件のツイートが関連付けられた．F 値の高い上位 10 位までのパラメータの組合せ（β，l，γ）と，その際に単語クラスタに関連付けられたツイート数（Size）を，**表 13** に示す．上位のパラメータの組合せを見ると，β は 0.08〜0.095，l は 0.05〜0.07，γ は 0.1〜0.3 の範囲で設定すると，今回の実験対象の場合には良い結果を得られることが分かる．

次にパラメータを $\beta = 0.08$ と $l = 0.07$ に固定し，γ を変動させた際の結果についてまず考察する．単語クラスタとツイートの関連付けの際の閾値 γ を，0.1 から 0.9 まで変動させた際の，適合率（Precision），再現率（Recall），F 値（F-measure），関連付けられたツイート数（Size）の変

表 14　最良パラメータでの各関連度の組合せによる評価値
Table 14　Performance using best parameter (Precison, Recall, and F-measure).

各関連度の組合せ	Precision	Recall	F-measure
$Cooc$	0.822	0.521	0.638
$Cooc + Time_{dif}(w_i \rightarrow w_j)$	0.861	0.363	0.511
$Cooc + Time_{dif}(w_i \rightarrow w_j) + Time_{std}(w_i \rightarrow w_j)$	0.832	0.636	0.721

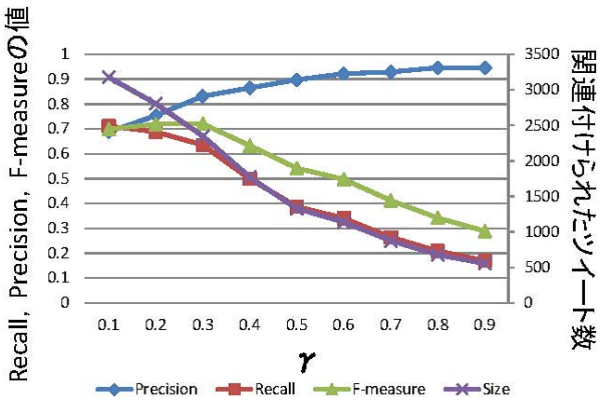


図 9　γ の値を変遷による精度の変化 ($\alpha = 0.55, \beta = 0.08, l = 0.07$)
Fig. 9　Performance differences on parameter γ ($\alpha = 0.55$, $\beta = 0.08$, $l = 0.07$).

表 15　γ の 0.3 から 0.4 への変動時に関連付けられなくなったツイート（一部）
Table 15　Example of removed tweets ($\gamma = 0.3 \rightarrow \gamma = 0.4$).

Answer	Score	Text
True	0.392	讀賣優勝 100%消滅ざまー！ T 3 - 1 G #hanshin #tigers
True	0.365	止めを差すんだ，城島！
True	0.317	チャンスを潰すことに長けた打線だな (´・ω・`)#kyojin #giants
False	0.392	横浜 4 点とってる!!　と思ったら 10 点取られてた…… これがベイスボールの真骨頂か……
False	0.364	【経済】(省略) ニトリホールディングス（HD）が 29 日発表 した 10 年 8 月期中間連結決算は，売上高が前年同期 (省略)
False	0.306	成田・中川，地元登板飾れず＝千葉国体（時事通信）

遷を図 9 に示す．

提案手法において，γ は，単語クラスタの持つ一般語に影響されて，誤ったツイートが関連付けられることを防ぐ目的で採用している．そのため，γ を低い値とすると，単語クラスタの持つ一般語に近い単語を持つ関係のないツイートが関連付けられることになり，適合率が低下することが予想される．図 9 を見ると，γ の値を高くしていくことで，適合率が上がっていくことが分かる．ただし，一般語を持ちながら，正解データとされたツイートが単語クラスタと関連付けられなくなるため，同時に再現率も低下する．**表 15** に，γ の値を 0.3 から 0.4 に変動させた際に，単語クラスタに対して関連付けられなかったツイートの一部を示す．太字は，単語クラスタ内に属する単語である．このとき，420 件の正解ツイートと，156 件の不正解ツイートが関連付けられなくなった．関連付けられなくなったツイートの内容を見ると，「坂本」，「大道」，「城島」といった選手名の単語のスコアが 0.4 未満となっており，それらについて言及された正解ツイートの多くが関連付けられなくなっていた．また，スコアの算出方法が単純な加算であるため，どうしても長文のニュース記事などは誤った関連付けが行われてしまう傾向も見られた．γ が 0.3 以下まで関連付けが行われていた表 15 の下 2 件のツイートがそれにあたる．

Yahoo! Web API と閾値 γ を用いて，一般語が区別されることで，不正解データが単語クラスタと関連付けられることを防ぐことができるが，一方で正解データも関連付けられにくくなり，再現率の低下につながる．本研究では，手法をシステムに適用した際に，誤ったツイートが単語クラスタに関連付けられることがなるべく少なくなることが望ましい．ここでの実験結果より，γ の値は 0.3 が適切であると判断する．

表 13 において F 値が最大となったときの α，β，および γ の値を用い，単語間関連度の計算方法として，単語共起のみ，単語共起および時間関連度，単語共起および時間関連度および間隔関連度の 3 つの場合を比較した際での，Precision，Recall，および F-measure の値を，**表 14** に示す．表 14 に示すとおり，F-measure の値については 3 つの単語間関連度を組み合わせた場合が最も高くなっている．また，単語共起および時間関連度の 2 つのみを用いた場合には，Precision の値は 3 つの場合で最も高くなっているが，Recall の値が大きく下がっていることから，多くの場面では 3 つの単語間関連度を組み合わせたほうが，総合的に見て良好な結果が得られると考えられる．

4.4.2　時間帯ごとの精度の変化

最後に，まとめられたツイート群を時系列で見た際に，高い精度でまとめられている時間帯と，あまり結果の良くない時間帯といった，時間帯ごとの分布の差がないかを調べる．ここでは，先の実験結果より，最も良い F 値を示し適合率も十分であったパラメータの組合せを用いる．**図 10** のグラフは，$\alpha = 0.55$，$\beta = 0.08$，$l = 0.07$，$\gamma = 0.3$ として求められた結果である．時間帯を 5 分ごとに区切り，その時間帯に属する正解ツイート数（緑線グラフ）と，本手法によってまとめられたツイート群内における TP（True Positive：青棒グラフ）と FP（False Positive：赤棒グラフ）を示している．グラフを見ると全時間帯において良好な結果が得られている．19 時 15 分から 19 時 25 分までの間のツイートは，誤って関連付けられたツイートが，他の時間帯と比較すると多い．**表 16** にその一部を載せる．これら

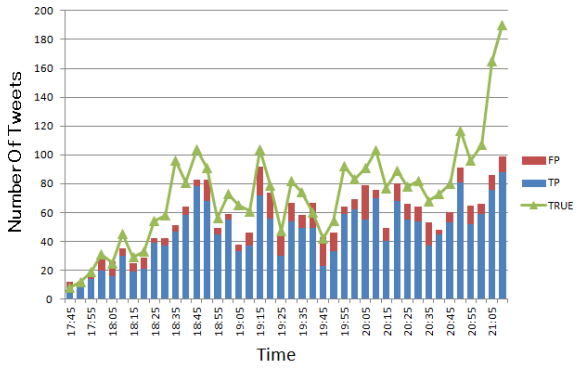


図 10　それぞれの時間帯におけるまとめられたツイート群の正解数・不正解数（$\alpha = 0.55$，$\beta = 0.08$，$l = 0.07$，$\gamma = 0.3$）

Fig. 10　Number of TP/FP tweets ($\alpha = 0.55$, $\beta = 0.08$, $l = 0.07$, $\gamma = 0.3$).

表 16　19 時 15 分から 19 時 25 分の間に投稿され誤ってまとめられたツイート（一部）

Table 16　FP tweets between 19:15 and 19:25.

Time	Text
19:16:02	状況が全くつかめない。とりあえず、ピンチなのは分かった。
19:17:20	腹は必死采配やな！鬼のようだ。
19:18:12	@xxx はいっ!!絶対勝ぁぁぁぁぁつ (●｀ョ´●)ﾉ
19:19:41	【社会】5 号機停止、さらに延長＝中部電力浜岡原発（時事通信）：中部電力は 29 日、これまで 9 月末ごろとしていた浜岡原発 5 号機（静岡県御前崎市）の停止期限をさらに延ばすと発表した。停止延長は 6 回目で、運転再開時期は未定。浜岡... [URL]
19:20:00	大道の結果予想 [URL] #nanjnews
19:20:00	大道打撃予想 [URL] #nanjnews
19:24:57	巨人が完全に脇谷を捨て田中賢介に乗り換えてる件 [URL] #nanjnews
19:24:58	高橋建の引退試合に無様な試合する野村カープ [URL] #nanjnews

を見ると，2ch [46] のスレッド名を発言する Bot のツイートと，ニュース記事の内容がまとめられている傾向があるように見えた．単語クラスタに属する単語群の影響を強く受ける本手法では，このようにスコアの高い単語が Bot のツイートやニュース記事で使われた単語とマッチしてしまうと，関連付けが行われてしまうことから，この傾向が見られたと考えられる．

また，時間帯によっては，ほとんどまとめることができていないツイートがあることが観測された．正解データ数を示す緑線のグラフと，青色の棒グラフの差が大きいほど，その時間帯において，再現率が悪いことを示している．18 時 35 分前後，20 時 55 分以降が顕著である．これは，この時間帯に起きた話題に関する単語が，単語クラスタ内にないことが主な原因である．

18 時 35 分から 18 時 40 分までの時間帯に属する正解データのツイートを**表 17** に一部載せる．単語クラスタと関連付けが行われたものには○，関連付けが行われなかったものには×が付いている．この時間帯は，×が多く，正解データのとりこぼしが多い．内容を見ると，金本選手の守るレフト方向への打球があって，それに関するツイートが多く見られる．単語「レフト」のスコアが低いために，ここでは「レフト」しか含まれないようなツイートの関連付けに失敗している．一方で，「レフト」「打つ」と 2 つの単語を持つツイートはスコアの合計値が γ の 0.3 を超えて

表 17　18 時 35 分から 18 時 40 分の間に投稿された正解ツイート（一部）

Table 17　TP tweets between 18:35 and 18:40.

Time	関連付け	Score	Text
18:38	○	0.682	しっかりせんかぁ～！能見～！！
18:38	○	1.101	巨人先制したな
18:37	×	0.072	レフトざまぁ！とか思うの新鮮
18:37	×	0.000	ヤニキ wwww #giants
18:37	×	0.085	投げられないなら取ってみろよ #hanshin #tigers
18:37	○	0.321	レフトに打つなんてずるい #hanshin #tigers
18:37	○	0.458	1 点で抑えよう！　#hanshin
18:37	×	0.000	ヤニキ wwww #giants
18:37	×	0.000	か www ね www も www と www #hanshin
18:37	×	0.000	あさむー偉いぞ！かわゆいぞ！
18:37	×	0.000	@xxx ブレッシャーに監督自らが打ち勝てるかですな
18:37	○	0.316	工藤松本はちっちゃいのによく当たるイメージ
18:37	×	0.072	おいおい，レフト誰も守ってねーじゃねーか！どうなってんだよ！ #hanshin

表 18　21 時 10 分から 21 時 15 分の間に投稿された正解ツイート（一部）

Table 18　TP tweets between 21:10 and 21:15.

Time	関連付け	Score	Text
21:12	○	0.611	とらほー久保田ほー！ RT @tt: とらほー久保田ほー！
21:12	×	0	うぉーうぉーうぉうぉーはーんしーんたいがーすふれぇふれっふれっふれっ！
21:12	×	0	とらほー！ RT @kk とらほー！ RT @aa: とらほー！ RT @hh: とらほー！
21:12	×	0	涙ぐんではりますなーイワックマー
21:12	×	0	とらほー RT @AAAA: とらほー
21:12	×	0	ん？ピッチャーだれだって？モミアゲ気になる・・・.
21:12	×	0	とらほー！ RT @mmmm: とらほー！
21:12	×	0	おめほー！ @dd とらほー！ QT @gg: とらほー！
21:12	×	0	せきほー！　#hanshin
21:12	×	0	マジック 7
21:12	×	0	う～ん有難う…（泣）・・・RT @rr: @jj 阪神かった？（＾∇＾）
21:12	○	1.403	お，巨人阪神 9 回までいったか。あとは藤川がおさえるだけだな
21:12	×	0	@iiii あ そか w カーブはいつとばすの？

いるため関連付けられ，本手法がうまく適用できた例といえる．

同様に，再現率の低い 21 時 10 分から 21 時 15 分までの時間帯に属する正解データのツイートを**表 18** に一部載せる．ちょうど試合が終わった直後で，この日の阪神対巨人戦は，阪神タイガースの勝利で終わった．その際に，「とらほー[*9]」と阪神ファンから多くツイートされたが，単語クラスタにそれに関連する単語がなかったため，関連付けることができなかった．こういった状況には，ハッシュタグなどの補助が必要だと考えられる．

5.　つながりに基づくツイート提示システム

ここでは，タイムライン上でのツイート提示の改善に本提案手法を適用した，試作システムについて述べる．

タイムライン上の記事の表示に対して，限られた画面領域内で「まとめ表示」を適用した場合の概観を**図 11** に示す．それまで自分のフォローする発信者が同一の話題に対して，ほぼ同じ内容のツイートを投稿していた場合，図 11 の左のように，小型携帯端末の表示画面の多くがそれらのツイートで埋め尽くされてしまう．

本システムでは，図 11 の右のように，意図的に共通話題

[*9]　「阪神わんだほー」の略

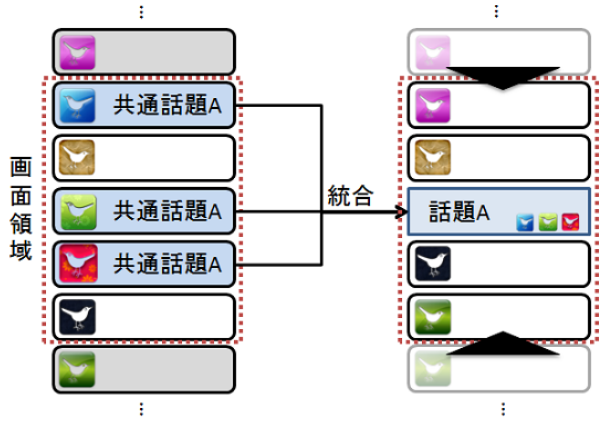


図 11　タイムラインへの「まとめ表示」適用の概観
Fig. 11　Overview of clustered timeline.

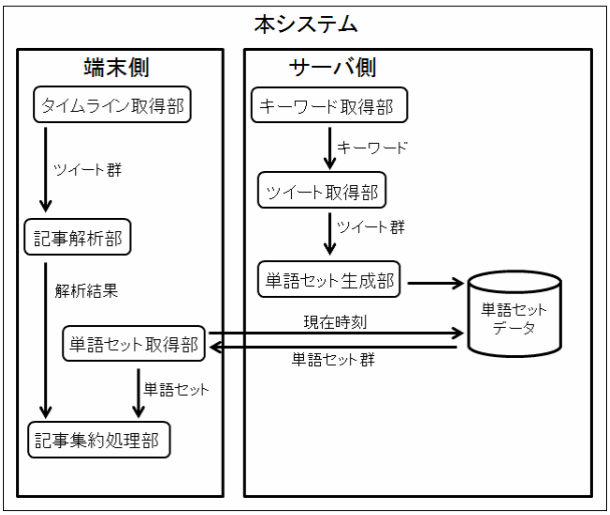


図 12　本システムの構成
Fig. 12　System architecture.

に関するツイートを 1 つのツイートのようにまとめて提示することを目的とする．ユーザが話題に関して興味を持った場合には，1 つのツイートのように集約された話題ごとのツイート群をクリックすることで，その話題に関してのツイートだけを閲覧することができる．

本システムの構成図を図 12 に示す．ユーザがアプリケーションとして本システムの端末側アプリケーションを起動した後，その端末側アプリケーションは通常の Twitter クライアントのようにユーザ自身のタイムラインを取得する．ここで，タイムラインとして取得したツイートは，必要に応じて形態素解析などが行われる．一方で，サーバ側ではつねにツイートをまとめる処理に必要な単語セットを更新しており，端末側アプリケーションで定期的にサーバ側の単語セットデータを保存しているデータベースに単語セットを読み込みに行き，解析結果と単語セットを用いてツイートを関連する話題ごとにまとめ，ユーザに提示する．ここでの単語セットとは，本論文で提案した手法を適用し，各キーワードごとに求めた，最もサイズの大きい単語クラスタに属する単語集合のことを指す．

本システムは，AndroidOS（1.6 以上）を搭載した端末上で動作するシステムを Java で実装を行っている．ウェブサーバとして Apache2.2，データベースマネジメントシステムとして MySQL を用いる．また，ツイートを形態素解析する際に，サーバ側では MeCab を用いる．AndroidOS を搭載した小型携帯端末に MeCab を搭載することも考えられるが，現状では 1 つのアプリケーションにつき最大 32 MB までしかメモリを確保できないことと，つねに最新の辞書を端末内に保持することの非効率性を考慮して，本提案システムでは代替の形態素解析器として，Yahoo! JAPAN が提供するテキスト解析 Web API の 1 つである「日本語形態素解析」を利用する．ただし，MeCab では問題となったメモリ使用量の問題は Web API では生じにくいが，Web API を利用する場合はネットワークを介したデータのやりとりが発生するため，応答速度上の課題が生じる．そのため，提案システムは現在，WiFi などを用いて小型携帯端末からでも高速に Web API を利用できる環境内での利用を想定している．

5.1　端末側での処理

端末側では，アプリケーションとして搭載されたシステムを起動することで処理が始まる．Twitter からタイムラインを取得する処理と，サーバ側から単語セットを取得する処理の関係で，ユーザは本システムをネットワークにつながる環境で利用していることが前提である．

Web API を用いて形態素解析する処理時間を考慮して，まとめ処理を行ったタイムラインを生成する処理は，別スレッドとしてバックグランドで実行し，その結果が返ってくるまでは通常のタイムラインを表示する方法を，本システムでは採用している．

5.1.1　タイムライン取得部

タイムライン取得部では，ユーザのタイムラインを Twitter から取得し，取得したツイート群を記事解析部に渡す．Twitter の提供する API を利用する場合，自身のタイムラインを取得するためにはユーザの OAuth[*10]認証が必要である．そのため，初めて本システムを起動した際には，OAuth 認証を通すための処理がタイムライン取得部の前に実行される．本システムでは，ユーザのタイムラインを最初に 200 件取得する．新しく投稿されたツイートを読み込む場合や，すでに読み込み済みの記事より古いツイートを読み込む場合には，それらも最大 200 件取得する．

5.1.2　記事解析部

記事解析部では，タイムライン取得部で取得したツイート群を形態素解析器にかけ，ツイートを単語単位に分ける．

[*10] 特定のアプリケーションが特定のサービスに対してユーザに代わって何らかの行動をすることを許可するためのプロトコル．詳細は oauth.net を参照されたい．

この形態素解析結果は，単語セットと記事の関連付けを行う際に用いられる．

5.1.3　単語セット取得部

単語セット取得部では，サーバ側のデータベースに保存されている単語セットを取得する．後述するが，イベントに関連する単語セットに属する各単語には，それぞれスコアが前もって求められている．そのスコアの値もあわせてここでは取得し，記事集約処理部で，単語セットに記事を関連付ける際と，アプリケーションを通してまとめた記事群を提示する際にそれを用いる．

サーバ側では，定期的に単語セットを生成する処理が行われるため，データベースには起動時から取得する時刻までに生成された単語セット群が保存されている．そのため，サーバ側に保存されている単語セットを取得する際には，現在時刻に基づいて，最近生成された単語セット群のみを取得する．本システムでは，現在時刻から過去 T 分以内に生成された単語セット群を取得するようにし，初期値では $T = 30$ となっている．この T の値を，システム側の設定から変更可能な値にしておくことで，各ユーザの意向に沿った結果を得ることが可能になると考える．

5.1.4　記事集約処理部

記事集約処理部では，記事解析部によるユーザのタイムラインに表示されるツイート群の解析結果と，単語セット取得部による現在時刻周辺に起きたとされるイベントに関連する単語セット群を用いる．記事解析部と単語セット取得部の両方の結果を受け取った後に，各単語セット群に対して，ツイートを関連付ける．閾値 γ を 0.3 とし，ここでは非排他的に単語クラスタに対しツイートを関連付ける．

5.2　サーバ側での処理

サーバ側では，定期的にキーワード取得部から処理を開始し，端末側からの要求がなくても，データベースに単語セットを記憶しておく．本システムでは一定時間（今回の設定では 10 分）おきにこの処理を繰り返し，端末側からの要求があった場合には，現在時刻周辺に関連するキーワードによって求められた単語セット群を返す．

5.2.1　キーワード取得部

キーワード取得部では，次のツイート取得部で用いる検索キーワードを取得する．期間情報を持つイベントが発生した際に，タイムライン上の多くにその記事が発生し，それらをまとめることを本システムは主に想定しているため，キーワード取得部では，できるだけ直近でバーストしているイベントに関連のある単語を検索キーワードとして取得したい．

本システムでは，直近で Twitter 内でバーストしている単語を取得するために，buzztter [1] を利用する．buzztter は，直近の Twitter ユーザの多くが言及している単語をリストとして提示しているサイトである．キーワードに言及している発信者の数，合計発言者数を用いて，その頻度を求める．詳細な背景技術については不明だが，2007 年 11 月 6 日時点 [6] では，直近 60 分間と 60 分前から 2 日前までの頻度の差分をとり，それをスコアとして上位を表示しているようである．

buzztter は，普段より多く言及されている単語ないし語句を提示しているため，本研究におけるイベントの検知に利用可能である．そのため，キーワード取得部では，定期的に buzztter のクロールによるデータ取得を行い，直近で多くの発信者がつぶやいているバズワード群を取得する．定期的にサーバ側で処理を行う際にはこのバズワード群に基づいて処理を開始する．バズワード群に変化がない場合には，その上位から順番にキーワードとして設定し，下流の処理にその検索キーワードを渡す．バズワード群に変化がある場合には，1 度バズワード群を最新のものに更新し，次回の更新があるまで最上位から順番に検索キーワードとしていく．

ここでは，外部のサービスに依存し，イベントの検知を行っているが，将来的には，サーバ側にバースト法を適用した手法を適用し，単独でもイベントの検知を可能にしたいと考える．Twitter などのマイクロブログサービスの中からイベントを検知する手法としては，蝦名らのリアルタイムバースト検出法 [9] が有効であると考える．従来のバースト検出手法とは異なり，一定時間ごとにバーストを検出するのではなく，記事の発生ごとに行う．また，短時間に大量の記事が発生した際でも，高速性を保つことが可能なアルゴリズムを採用しているため，Twitter のようなマイクロブログサービスのイベント検知には有効であると考えられる．

5.2.2　ツイート取得部

ツイート取得部では，キーワード取得部から検索キーワードを受け取り，そのキーワードを基に解析対象となるツイート群を取得する．本論文での提案手法は，時系列的な記事の連続性に着目し，ツイート群をまとめているため，検索キーワードの検索結果のみでは，それらは解析対象として十分なツイート群を含むとは限らない．

解析対象として十分なツイート群を収集するために，4 章の実験を行う際の記事抽出方法をもとにしたツイート収集機構を実装している，実験時には，前もって収集済みの大規模な記事群を対象とし，イベントのキーワードと発生期間の情報を与え，特定の発信者らのツイートのみを抽出し，それを実験対象とした．本システムを実際に運用する場面では，イベントのキーワードはキーワード取得部で取得可能であるが，実験時のように事前にイベントの発生期間の情報を得ることは困難である．また，Twitter API の使用限度の都合上，常時大量の記事群を用意しておくことも難しい．そこで，本システムの運用時では，代替方法として以下の手順により解析対象となるツイート群を取得する．

(1) 単一の検索キーワード kw をキーワード取得部より受け取る．
(2) 最新の kw を含むツイート群 d を取得する．ここでは最大 200 件．
(3) d の中で，ツイートの持つタイムスタンプが現時刻より 1 時間以内のツイート群を d' とする．
(4) ツイート群 d' を投稿した発信者 ID を取得し，それらの発信者 ID 群を U とする．
(5) それぞれの発信者の直近 30 分以内の発言を取得し，それらの集合をツイート群 D とする．ただし，30 分以内の合計発言数が閾値 k を下回る発信者のツイートは D に含めない．

この方法で取得した記事群 D を，単語セット生成部に渡す．

ツイート取得部は，Twitter の提供する API [41] を利用して記事を収集する．実験結果より，解析対象として十分な記事数がある場合には，本手法は一定の有効性を得ることが可能である．そのため，ツイート取得部でも，API を利用して多くのツイートを取得し，次の単語セット生成部にそれらを渡したい．しかし，2012 年 1 月の段階では Twitter の提供する API には，1 時間あたりの使用制限が設けられており，十分なツイート群を取得することは困難である．現在，API は OAuth 認証済みの Token 1 つにつき 1 時間あたり約 300 回ほどである．ただし，この回数は Twitter の運用状況によってはこの値よりも低い値が上限となる場合もある．上記の手順を定期的に行う場合，最大発信者数 200 件をもとに，手順 3 のツイート群 d を取得する際にその使用回数の多くを割いてしまう．Twitter を利用する特定の発信者の発言を API を通して取得する方法は以下の 2 通りが考えられる．

(1) http://api.twitter.com/1/statuses/user_timeline.format を利用し，特定の発信者の投稿したツイートを取得．
(2) http://search.twitter.com/search.format を利用し，検索条件に「from:UserName」を指定し，その発信者の投稿したツイートを取得．

(1) の方法は，API に制限が設けられていない場合は有効である．ただし，API 制限が設けられている現在，本システムのように短時間で 200 名の発信者のツイートを取得したい場合には不適切であると考える．そのため，ここでは 2 の方法を採用し，特定の発信者のツイートを取得する．http://search.twitter.com/search.format を利用する場合，引数に「q = 検索語句 or 条件」を指定することで，その検索語句または条件を満たす検索結果を取得することができる．この検索条件に，「from:UserName」を指定することで，ユーザ名 UserName の発信者の投稿したツイート群を結果として受け取ることができる．この条件には「+OR+」とすることで OR 検索を適用することが可能である．本システムでは，(2) の方法を用いて API の使用回数を削減する．発信者 $user1$，$user2$，$user3$ の投稿したツイートを取得する際に，方法 (1) では，API を 3 回利用する必要があるのに対し，方法 (2) では，「q=from:user1+OR+from:user2+OR+from:user3」とすることで，1 度に 3 名の発信者のツイート群を取得することが可能である．ただし，方法 (2) の場合，1 度に取得できる件数は 100 件が限度であるため，この例の場合，仮に $user1$ が短時間に 100 件以上投稿していた場合には，検索結果に，1 人の発信者によるツイートが多く含まれた状態となり，ツイート群が大きく偏る可能性がある．しかし，このように偏る例は比較的稀であることが予備実験で確認されている．

そのため，本システムでは，発信者の発言を取得する際には，API 制限に対応するために 1 度に最大 3 名までを「from:UserName」として指定し，http://search.twitter.com/search.format を利用し，検索結果を受け取る方法を採用する．

将来的に，API 制限が解消されることが最良であるが，制限が設けられている場合には，発信者の過去の発言数などを考慮し，流動的な組合せを考え，効率的にツイートを取得する仕組みを搭載したい．現在 Twitter の提供する検索に該当する API では，過去 1 週間以内に投稿されたツイートのみが検索結果として返されるが，これは直近のツイートを主に対象とする本システムにおいては問題とはならない．発信者であるユーザの持つ ID は Twitter 内では厳密には数値であり，ユーザ名を示す UserName は，ScreenID と呼ばれ常時変更が可能である．そのため，過去に取得したツイート群の投稿者の ScreenID を用いて方法 (2) を使用する場合には，その ScreenID がすでに変更済みの可能性がある点に注意する必要がある．また，予備実験より，特定の発信者の発言を高速に取得した場合，(1) の方法は，30 名の発信者の発言をそれぞれ最大 200 件取得する際に合計約 100 秒，(2) の方法は，30 名の発信者の発言をそれぞれ「from:UserName」として指定し（「q=from:UserName」とし OR 検索は用いない），最大 100 件取得する際に合計約 15 秒かかることに留意されたい．特定の発信者の最近（1 週間以内）のツイートを対象とし，多くの記事を必要としない場合には，(2) の方法の採用が適していると考える．

5.2.3　単語セット生成部

ここでは，まずツイート取得部で取得したツイートを形態素解析する．次に提案手法を適用し，出現順序を考慮した制約を用いて単語間の関連度を算出する．そして，求めた単語間の関連度を用いて単語クラスタリングを適用し，その結果から単語クラスタを 1 つ選択する．ここでのパラメータは，実験結果より，適合率と F 値の結果が良かったパラメータ，$\alpha = 0.55$，$\beta = 0.1$，$l = 0.07$ としている．ここで求めた単語クラスタをデータベースに格納しておき，

図 13　本システムの動作例
Fig. 13　Prototype system using our method.

端末から要求があった際に，時刻に適した単語クラスタ群を返す．

5.3　実行例および考察

本システムのユーザインタフェース部分は，AndroidOS を搭載したタッチデバイス上で動作する Java アプリケーションとして試作した．システムの動作例を，**図 13** に示す．ここでは，プライバシ面での配慮から記事などの内容を若干改変して表示している．

図 13 (a) は，あるタイムライン上での，サッカーの試合に関する話題に対する言及記事を 1 つにまとめた様子を示している．今回のシステムでは，3 章で示した手法によって得られた，単語間の関係に基づいてクラスタ化した単語間のつながりを用い，記事での単語の出現の有無により，それらを 1 つにまとめて表示した例を示している．図 13 (b) は，ある話題についての記事群としてまとめられた中から，直近に投稿された記事を抽出して提示している．図 13 (c) は，共通の話題の時系列的な変化の追跡を目的とし，表示粒度を変更し，それぞれの特徴語などを提示している．ここでの粒度とは，たとえば時間間隔に基づいて記事集合を区切る際の窓の大きさである．

効果的なタイムラインの提示方法としては，同様の記事が多くタイムライン上に出現することを防ぐ目的で，同様の記事を複数まとめて 1 つの記事のように扱って提示する方法が考えられる．さらに，1 つにまとめられた話題の詳細について詳しく記事を読めるようにする方法，あるいは全体での話題の流れを俯瞰的に把握できるようにする方法などが考えられる．

本試作システムでは，ある話題について詳しく提示する方法として，たとえば，その話題に強く関連する直近の記事から特徴的なもののみを選び出してその数件を提示できるようにしている．同時に，ある話題について触れられた記事群を，異なる時間粒度の面から，特徴語などにより話題の変遷を追跡できるようにしている．本試作システムでは，これらの方法をそれぞれ実装するとともに，それらの具体的な使用例を示した．

6.　おわりに

本論文では，特定の話題に関するツイートをまとめるために，単語の時系列的な出現間隔に着目し，単語間の関連度を算出する手法を提案した．このために，単語の共起と時系列的な近さに基づく指標と，出現間隔の同一性に着目した間隔に基づく指標の 3 つの指標を定義した．実際のイベントを対象とし，本手法で提案した 3 つの指標を用いることで，それぞれの指標の特徴を持つ単語ペアを得ることができることを，予備実験により確認した．また，その指標によって求めた単語間の関連度を用いることで，イベントに関する単語集合を得ることができることを確認した．

指標に関する実験結果から，単語の出現間隔に指向性を持たせ，単語間に前もって制約を生成する手法を提案した．指向性による制約と 3 つの指標を用いて，単語集合を得ることで，つながりを持つ単語どうしをまとめられることを示した．制約と指標を単語間のつながりにより求め，単語集合に対しツイートを割り当てることで，同一の話題に対して言及しているツイート群が得られることを示した．

AndroidOS を搭載する端末上で動作可能な，本手法に基づく Twitter 閲覧支援システムを実装した．タイムライン上から，特定のイベントに関するツイートをまとめた形で提示することで，閲覧性の向上を目指した．

本論文で提案した手法について，Twitter に特化した手法ではなく，汎用の文書群からのトピック抽出手法としてみた場合における，その性質の解析および関連手法との比較評価は今後の課題である．また，本論文で示したシステムでの，実際のツイート閲覧の支援への貢献度に対する定量的な評価も，今後の課題である．

謝辞　担当編集委員および査読者らの的確で丁寧な指摘

に心より感謝する．本研究の一部は，科研費 No.23650046，22700142，22700094 の支援による．

参考文献

[1] buzztter - Twitter のイマを切り取ったー☆，⟨http://buzztter.com/⟩.

[2] Cha, M., Haddadi, H., Benevenuto, F. and Gummad, K.P.: Measuring User Influence on Twitter: The Million Follower Fallacy, *Proc. 4th Int'l AAAI Conference on Weblogs and Social Media*, Washington, DC (2010).

[3] Chapelle, O., Schlkopf, B. and Zien, A.: Semisupervised Learning, *Adaptive Computation and Machine Learning*, Cambridge, Mass., USA, MIT Press (2006).

[4] Chen, K.Y., Luesukprasert, L. and Chou, S.T.: Hot Topic Extraction Based on Timeline Analysis and Multidimensional Sentence Modeling, *IEEE Trans. Knowledge and Data Eng.*, Vol.19, No.8, pp.1016–1025 (2007).

[5] Cortes, C. and Vapnik, V.: Support Vector Networks, *Machine Learning*, Vol.20, pp.273–297 (1995).

[6] dara 日記—buzztter の裏側とその周辺技術，⟨http://d.hatena.ne.jp/darashi/20071106⟩.

[7] Dong, A., Zhamg, R., Kolari, P., Bai, J., Diaz, F., Chang, Y., Zheng, Z. and Zha, H.: Time is of the Essence: Improving Recency Ranking Using Twitter Data, *Proc. 19th International Conference on World Wide Web* (*WWW2010*), pp.331–340 (2010).

[8] Duan, Y., Jiang, L., Qin, T., Zhou, M. and Shum, H.-Y.: An Empirical Study on Learning to Rank of Tweets, *Proc. 23rd International Conference on Computational Linguistics* (*Coling 2010*), Beijing, pp.295–303 (2010).

[9] 蝦名亮平，中村健二，小柳　滋：リアルタイムバースト検出手法の提案，日本データベース学会論文誌，Vol.9, No.2, pp.1–6 (2010).

[10] Go, A., Bhayani, R. and Huang, L.: Twitter Sentiment Classification Using Distant Supervision, CS224N Project Report, Stanford (2009).

[11] Hannon, J., Bennett, M. and Smyth, B.: Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches, *Proc. 4th ACM Conference on Recommender Systems* (*RecSys'10*), New York, NY, USA, pp.199–206 (2010).

[12] Kwak, H., Lee, C., Park, H. and Moon, S.: What is Twitter, a Social Network or a News Media?, *Proc. 19th International Conference on World Wide Web* (*WWW2010*), pp.591–600 (2010).

[13] Loeb, S. and Terry, D.: Information Filtering, *Comm. ACM*, Vol.35, No.12, pp.26–81 (1992).

[14] Manevitz, L.M. and Yousef, M.: One-class SVMs for Document Classification, *Journal of Machine Learning Research*, Vol.2, pp.139–154 (2002).

[15] Hu, X., Sun, N., Zhang, C. and Chua, T.-S.: Exploiting Internal and External Semantics for the Clustering of Short Texts Using World Knowledge, *Proc. ACM International Conference on Information and Knowledge Management* (*CIKM2009*), Hong Kong, China, pp.919–928 (2009).

[16] 石川佳治，北川博之：忘却の概念に基づくクラスタリング手法の改良方式，日本データベース学会 Letters，Vol.2, No.3, pp.53–56 (2003).

[17] Jain, K.A.: Data Clustering: 50 Years Beyond K-means, *Pattern Recognition Letters*, Vol.31, No.8, pp.651–666 (2010).

[18] Java, A., Song, X., Finin, T. and Tseng, B.: Why We Twitter: Understanding Microblogging Usage and Communities, *Proc. 9th WebKDD and 1st SNA-KDD2007 Workshop on Web Mining and Social Network Analysis*, San Jose, CA, pp.56–65 (2007).

[19] Joachims, T.: A Probablistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization, *Computer Science Tech. Report*, CMU-CS-96-118 (1996).

[20] 鎌田基之，福田直樹，横山昌平，石川　博：ブログの相互関係性を考慮したブログ記事分類手法の検討，情報処理学会論文誌：データベース，Vol.2, No.2, pp.56–70 (2009).

[21] Kleinberg, J.: Bursty and Hierarchical Structure in Streams, *Proc. 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD2002*) (2002).

[22] MeCab, available at ⟨http://mecab.sourceforge.net/⟩.

[23] Miller, G.A.: WordNet: A Lexical Database for English, *Comm. ACM*, Vol.38, No.11, pp.39–41 (1995).

[24] 村松亮介，福田直樹，横山昌平，石川　博：SearchLife：単語の特徴量を考慮した多視点クラスタリング検索エンジン，情報処理学会論文誌：データベース，Vol.3, No.2, pp.123–137 (2010).

[25] 奥村　学：マイクロブログマイニングの現在，電子情報通信学会技術研究報告，Vol.111, No.427, NLC2011-59, pp.19–24 (2012).

[26] Phan, X., Nguyen, L. and Horiguchi, S.: Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections, *Proc. 17th International Conference on World Wide Web* (*WWW'08*), New York, NY, USA, ACM, pp.91–100 (2008).

[27] 坂本　翼，横山昌平，福田直樹，石川　博：マイクロブログを対象としたリアルタイムな要約生成システムの試作，第 3 回データ工学と情報マネジメントに関するフォーラム（DEIM フォーラム 2011）(2011).

[28] Semiocast　Brazil becomes 2nd country on Twitter, Japan 3rd - Netherlands most active country, available at ⟨http://semiocast.com/publications/2012_01_31_Brazil_becomes_2nd_country_on_Twitter_superseds_Japan⟩.

[29] State of the blogosphere 2009 introduction - technorati blogging, available at ⟨http://technorati.com/blogging/article/state-of-the-blogosphere-2009-introduction/⟩.

[30] Sriram, B., Fuhry, D., Demir, E. and Ferhatosmanoglu, H.: Short Text Classification in Twitter to Improve Information Filtering, *Proc. 33rd ACM International Conference on Information Retrieval* (*SIGIR'10*), pp.841–842 (2010).

[31] 角谷和俊，松本好市，高橋美乃梨，上原邦昭：マルチチャネル型ニュース配信システムのための時系列クラスタリング，情報処理学会論文誌：データベース，Vol.43, No.SIG 5(TOD14), pp.87–97 (2002).

[32] Swan, R. and Allan, J.: Automatic Generation of Overview Timelines, *Proc. 23rd ACM International Conference on Information Retrieval* (*SIGIR2000*), pp.49–56 (2000).

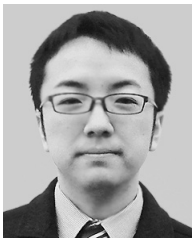
[33] 高村大地，横野　光，奥村　学：Summarizing Microblog Stream，人工知能学会第 22 回セマンティックウェブとオントロジー研究会，SIG-SWO-A 1001-03 (2010).

[34] 竹中姫子，古宮嘉那子，小谷善行：ベイジアンフィルタを用いた Twitter におけるツイートのハッシュタグ分類，情報処理学会研究報告，情報学基礎研究会報告，2011-102(1), pp.1–6 (2011-03-21).

[35] Tan, P.M., Steinbach, M. and Kumar, V.: *Introduction to Data Mining*, Addison-Wesley (2006).

[36] 戸田浩之，北川博之，藤村　考，片岡良治：時間的近さを考慮した話題構造マイニング，電子情報通信学会第 18 回

データ工学ワークショップ（DEWS2007）(2007).
[37] 戸田智子，横山昌平，福田直樹，石川　博：局所性を用いた多様性を考慮したブログからのトピック抽出手法について，第 1 回データ工学と情報マネジメントに関するフォーラム（DEIM フォーラム 2009）(2009).
[38] 徳永健伸，岩山　真：重み付き IDF を用いた文書の自動分類について，自然言語処理研究会報告 94，pp.33–40 (1994).
[39] Twitter, available at ⟨http://twitter.com/⟩.
[40] Twitter Blog, 200 million Tweets per day, available at ⟨http://blog.twitter.com/2011/06/200-million-tweets-per-day.html⟩.
[41] Twitter for Websites — Twitter Developers, available at ⟨https://dev.twitter.com/⟩.
[42] Twitter 利用実態調査の結果，⟨http://asciimw.jp/info/release/pdf/20091228.pdf⟩.
[43] Wagstaff, K., Cardie, C., Rogers, S. and Schroedl, S.: Constrained K-means Clustering with Back-ground Knowledge, *Proc. 18th International Conference on Machine Learning* (*ML2001*), San Francisco, CA, pp.577–584, Morgan Kaufmann (2001).
[44] Yahoo!デベロッパーネットワーク—検索，⟨http://developer.yahoo.co.jp/webapi/search/⟩.
[45] Wikipedia—The Free Encyclopedia, available at ⟨http://www.wikipedia.org/⟩.
[46] 2 ちゃんねる，⟨http://www.2ch.net/⟩.



青島 傳隼

2010 年静岡大学情報学部情報科学科卒業．同年同大学大学院情報学研究科修士課程入学．2012 年同大学院情報学研究科修士課程修了．同年株式会社ドワンゴ入社．現在，株式会社キテラス勤務．修士（情報学）．在学中は，Web マイニング，マイクロブログマイニングに興味を持つ．日本データベース学会会員．

坂本 翼

2011 年静岡大学情報学部情報科学科卒業．同年同大学大学院情報学研究科修士課程入学．現在に至る．Web マイニング，マイクロブログマイニングに興味を持つ．日本データベース学会会員．

横山 昌平　（正会員）

静岡大学情報学部講師．産業技術総合研究所，静岡大学情報学部助教を経て 2012 年より現職．2006 年東京都立大学大学院工学研究科修了，博士（工学）．データベース技術の研究開発に従事．電子情報通信学会，日本データベース学会各正会員．会誌「情報処理」BWG 幹事．

福田 直樹　（正会員）

1997 年名古屋工業大学工学部知能情報システム学科卒業．1999 年同大学大学院工学研究科電気情報工学専攻博士前期課程修了．2002 年同大学院博士後期課程修了．同年静岡大学情報学部情報科学科助手．2007 年より同助教．2010 年より同講師．現在に至る．博士（工学）．2012 年山下記念研究賞．IEEE-CS，ACM，人工知能学会，ソフトウェア科学会，情報システム学会各会員．2011 年より論文誌ジャーナル・JIP 編集委員会知能グループ副査．

石川 博　（フェロー）

静岡大学情報学部情報科学科教授．東京大学理学部情報科学科卒業．東京都立大学を経て 2006 年より現職．東京大学博士（理学）．著書に『データマイニングと集合知—基礎から Web，ソーシャルメディアまで—』（共立出版），『集合知の作り方・活かし方—多様性とソーシャルメディアの視点から』（共立出版），『データベース（情報工学レクチャーシリーズ）』（森北出版）等．国際論文誌 ACM TODS，IEEE TKDE，国際学会 VLDB，IEEE ICDE 等学術論文多数，1994 情報処理学会坂井記念特別賞，1997 科学技術庁長官賞（研究功績者）受賞．情報処理学会データベースシステム研究会主査，情報処理学会（データベース）共同編集委員長，International Journal Very Large Data Bases Editorial Board，日本データベース学会理事歴任．電子情報通信学会フェロー．ACM，IEEE 各会員．

（担当編集委員　河野 浩之）