マイクロブログを用いた鉄道の運行トラブル発生期間および付帯情報の抽出

土屋　圭[†]　豊田　正史[††]　喜連川　優[††,†††]

† 東京大学大学院情報理工学系研究科　〒 113–8656 東京都文京区本郷 7-3-1
†† 東京大学生産技術研究所　〒 153–0041 東京都目黒区駒場 4-6-1
††† 国立情報学研究所　〒 101–8430 東京都千代田区一ツ橋 2-1-2
E-mail: †{tsuchiya,toyoda,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし　近年，スマートフォンやソーシャルネットワークサービスの普及によって，リアルタイム性のある，詳細な情報がウェブにアップロードされるようになった．その結果，ソーシャルメディアには他のメディアよりも実世界の動きが早期に反映される．また，ソーシャルメディアには他メディアの報道内容を補完するような，より詳細な情報が含まれることもある．実世界のイベントについて，リアルタイムかつ詳細に把握することは意思決定を下す際に非常に重要である．そこで，ソーシャルメディアから実世界のイベントを抽出するという機運が高まっている．本論文では，鉄道運行トラブル発生時の意思決定支援を目的に，Twitter の投稿を解析することによってトラブルの検出および影響の予測を行う手法を提案する．トラブル検出ではバースト検出手法を用いた実験により，トラブルを早期に検出できることを示す．影響の予測ではトラブルの継続時間および他路線への影響の予測を行い，実際の運行データと比較して提案手法の有効性を示す．

キーワード　データマイニング，ウェブマイニング，マイクロブログ，イベント抽出

1. はじめに

近年，スマートフォンやソーシャルネットワークサービスの普及によって，リアルタイム性のある，詳細な情報がウェブにアップロードされるようになった．スマートフォンの普及率は世界中で増加の一途を辿っており，特に日本においては 2016 年までに 70%を超えると推定されている．世界中で広く利用されているソーシャルネットワークサービスの 1 つである Twitter では，2012 年 6 月に 1 日の投稿件数が 4 億を超えた．また，Twitter のアクティブユーザの 60%がモバイル端末から利用しているということも分かっている．人々は Twitter のようなリアルタイムに情報を共有するウェブサービスを，スマートフォンによっていつでも，どこでも利用することができる．その結果，実世界で発生したイベントは即座にウェブ上のコンテンツに反映される．

我々は実世界のイベントの情報をテレビや新聞，ニュースサイト，ソーシャルメディアなど様々なメディアを通じて知ることができる．特にソーシャルメディアには，他のメディアよりも実世界の動きが即座に反映されることや，他メディアの報道内容を補完するような，より詳細な情報を含む場合があるという特徴がある．例えば，地震や台風などの自然災害，交通渋滞，鉄道の運行トラブルなどが発生すると，ソーシャルメディアには現場の状況が即座に反映される．実世界のイベントについて，リアルタイムかつ詳細に把握することは意思決定を下す際に非常に重要である．そこで，ソーシャルメディアから実世界のイベントの情報を抽出するという機運が高まっている．

本論文では，鉄道運行トラブルを対象に Twitter の投稿を解析することによって情報抽出を行う．鉄道運行トラブルは，数千数万人規模の人が分単位の意思決定に迫られるようなイベントである．例えば，トラブルに巻き込まれた際，次の予定を中止にすべきか，今駅に向かうべきか，このまま車内で待機すべきか，他路線に乗り換えるべきかなど様々な意思決定を行う必要がある．そこで，本論文では鉄道運行トラブル発生時のリアルタイムな意思決定支援を目的に，トラブルの検出および影響の予測を行う手法を提案する．トラブル検出ではバースト検出手法を用いた実験により，トラブルを早期に検出できることを示す．影響の予測ではトラブルの継続時間および他路線への影響の予測を行い，実際の運行データと比較して提案手法の有効性を示す．

本論文の構成は以下の通りである．まず，2 章ではソーシャルメディアから実世界のイベントの抽出を行った関連研究について述べる．次に 3 章で本論文で行うタスクの全体像について説明する．4 章ではトラブル関連ツイートを抽出する方法について述べる．5 章ではバースト検出手法を用いた運行トラブルの発生および復旧検出を行う．そして，6 章では運行トラブル継続状況を予測する手法，7 章では運行トラブルの連鎖の予測手法について提案し，評価実験を行う．最後に 8 章で本論文をまとめる．

2. 関連研究

ソーシャルメディアを解析することによって，実世界のイベントを抽出するという研究は多岐に渡っている．先行研究は対象としているイベントの種類によって，人の行動や状態，社会の動き，経済の動き，自然現象の 4 つに大別することができる．

人の行動パターンや状態などを対象にした研究として，Adar ら [1] らや Sadilek ら [2] らの研究がある．Adar らはインターネットユーザの過去の検索エンジンのクエリを解析することによって，今後の振る舞いを予測するという研究を行った．Sadilek らは，Twitter の投稿内容，投稿位置，フォロー・フォロワー関係を解析することによって，人の健康状態を予測するという研究を行った．

次に，社会の動きを対象とした研究として，Radinsky ら [3] や Panagiotis ら [4] による研究がある．Radinsky らは 150 年分のニュース記事を解析することによって，あるイベントが発生した際に，そのイベントによって将来引き起こされるであろうイベントを予測する手法を提案した．Panagiotis らは Twitter を解析することによって，米国議会選挙の予測を行った．

また，経済活動を対象とした研究として，Gilbert ら [5] や Bollen ら [6]，Gruhl ら [7] の研究がある．Gilbert らと Bollen らはいずれも株価変動の予測を行った．Gruhl らはブログ上の不安表現と株価指数の変動に相関があることを発見し，この相関を用いることによって株価変動予測を行った．一方で，Bollen らは Twitter に投稿された感情表現に着目して株価変動予測を行った．Daniel らは，ある書籍のブログ上での言及度合いからその書籍の売上を予測する手法を提案した．

最後に，自然現象を対象とした研究として，Haipeng ら [8] や Sakaki ら [9] の研究がある．Haipeng らは写真共有サイトである Flickr(注1) を解析することによって，アメリカの積雪および植生の有無を観測する研究を行った．Sakaki らは Twitter の投稿内容と投稿位置を用いて，地震と台風それぞれの発生時間および発生位置を推定した．

以上で述べた先行研究は，ソーシャルメディアの様々な情報を用いて，対象イベントの基本的な情報を抽出している．しかし，意思決定支援を行えるような情報を抽出している研究は著者の知る限り，存在しない．対象とするイベントの基本情報だけでなく，そのイベントの付帯情報や影響など，具体的な意思決定を可能にできうる情報の抽出がこの分野の課題であると言える．

3. タスク設定

本章では，本論文で行うタスクの全体像について説明する．本論文では，大きく 3 つのタスクに取り組む．

まず，鉄道運行トラブル発生後および復旧後の早期対応を支援するため，トラブルの発生時間および復旧時間の検出を行う．その際，発生したトラブルの深刻さも同時に提示するため，トラブルを全線見合わせ，一部見合わせ，見合わせ (全線および一部見合わせを同一とみなしたトラブル) の 3 つに分けて検出を試みる．検出した時間と実際の時間の差を評価することによって，トラブルを早期に検出できたかどうかを確認する．

次に，トラブル発生後，次の予定を検討する際の意思決定支援のために，トラブルが一定時間以上長引くかどうかの予測を行う．提案手法の有効性は実際の継続時間と予測結果を比較することによって評価する．

最後に，トラブル発生時の他ルートを検討する際の意思決定支援のため，トラブルが他の路線に影響を及ぼすかどうかの予測も行う．提案手法の有効性は実際のトラブル連鎖状況と予測結果を比較することによって評価する．

4. トラブル関連ツイートの抽出

本章では，路線名を含むツイートからトラブル関連ツイートを抽出する手法について述べる．

4.1 トラブル関連ツイート分類器の作成

本論文では，路線名を含むツイートを解析対象とした．路線名を含むツイートには “今，千代田線が全線で運転見合わせている” というように，現在の運行トラブル状況について述べているものが含まれるため，トラブルの状況を把握する上で重要な手掛かりとなる．しかし，“これから千代田線に乗る” のように運行トラブルとは関係ないツイートも存在すため，ツイートが現在の運行トラブルについて述べているかどうかを判定する必要がある．

そこで，ツイートがトラブルに関連しているかどうかを分類する分類器を作成した．分類器には線形カーネルの SVM，SVM のライブラリには LIBSVM(注2) を用いた．特徴量には，トラブルの状況はツイートの内容から得ることができるため，ツイートに含まれる形態素を用いた．日本語の形態素解析器には MeCab(注3) を用いた．

4.2 ラベル付きツイートデータの作成

SVM の学習データを作成するために，約 100 万ユーザによって投稿された約 100 億ツイートの中から，東京メトロ 9 路線のいずれかの路線名を含むツイートを取得する．なお，ツイートの正規化のため，収集した全てのツイートに対して，i) 検索時に使用した路線名を “QUERYWORD” という文字に置換，ii) 投稿に含まれる URL を “URL” という文字に置換，iii) 投稿に含まれる@から始まるユーザ名を “MENTION” という文字に置換，という処理を行った．

次に，それらのツイートの中から 2011 年 9 月 21 日(注4)，2012 年 4 月 3 日(注5)，2012 年 9 月 30 日(注6) の 3 日間に投稿されたツイートをランダムに 5000 件サンプリングし，人手でラベル付けを行う．ラベルの種類は運行トラブルの発生状況，復旧状況の 2 つであり，それぞれ独立にラベル付けを行った．トラブル発生状況のラベルは，T_0：運行トラブル発生状況について言及なし，T_1：全線運転見合わせ，T_2：一部区間運転見合わせ，T_3：遅延や直通運転中止など運転見合わせ以外の異常の 4 種類，トラブル復旧状況のラベルは R_0: 復旧状況について言及なし，R_1: 復旧状況について言及ありの 2 種類である．なお，復旧状況については，実際にツイートを確認した結果，どの種類のトラブルから復旧したのかを判断することが困難であるため，ラ

(注1)：http://www.flickr.com/

(注2)：http://www.csie.ntu.edu.tw/ cjlin/libsvm/

(注3)：http://mecab.sourceforge.net/

(注4)：台風 15 号の影響で全路線で見合わせや遅延などが生じた．

(注5)：強風の影響で銀座線を除く全路線で遅延などが生じた．

(注6)：台風 17 号の影響で 6 路線で遅延などが生じた．

ベルは 2 種類のみとした．ラベル付けの結果を表 1 に示す．

	T_0	T_1	T_2	T_3	R_0	R_1
#tweet	3010	868	461	661	4745	255

表 1　ラベル付けの結果

4.3　分類器の性能評価

10 分割交差検定によって，分類器の性能評価を行った．表 2 に性能評価の結果を示す．単一のトラブルを対象とした場合は T_1 は F 値が 7 割弱であるのに対し，T_2，T_3 は 6 割前後と T_1 よりも性能が下がる結果となった．また，複数ラベルを同一のトラブルと見なし，トラブルの粒度を粗くすると，分類器の性能が上がることも確認できた．

Target Label	Precision	Recall	F-value
T_1	0.695	0.666	0.680
T_2	0.607	0.573	0.589
T_3	0.583	0.534	0.557
$T_1 + T_2$	0.715	0.711	0.713
$T_1 + T_2 + T_3$	0.759	0.746	0.752
R_1	0.710	0.671	0.690

表 2　10 分割交差検定の結果

5.　鉄道運行トラブルの検出

本章では，鉄道運行トラブルの発生時間および復旧時間を検出する方法について述べ，検出した時間と実際の時間の差を評価することによって，提案手法の有効性を確認する．なお，トラブルは全線見合わせ，一部見合わせ，見合わせの 3 つに分けて検出を試みる．

5.1　提 案 手 法

本論文ではバースト検出手法を用いて，トラブルの発生時間および復旧時間の検出を行う．データストリーム中の異常箇所はバーストと呼ばれ，バーストを検出する様々な研究 [10], [11], [12], [13], [14] が行われている．運行トラブルが発生した時間帯にはトラブル関連ツイートが数多く投稿される．トラブル関連ツイートの出現時系列をデータストリームとみなすと，トラブル関連ツイートが集中している箇所はバーストとみなせる．

そこで，4.1 節で作成した分類器によって抽出したトラブル関連ツイートの出現時系列に対して蝦名ら [14] のリアルタイムバースト検出手法を適用する．バースト検出によって，バーストした期間が得られるため，本論文では，その期間の終了時間をトラブルの検出時間とする．蝦名らの手法は Zhang ら [11] の手法をリアルタイムにバースト検知ができるように改良したものである．Zhang らの手法は一定期間ごとにバースト解析を行う．監視する間隔を短くすることでリアルタイムに近いバースト検出が可能になるが，イベントが発生していない期間に無駄な計算が行われる．したがって，監視対象とするイベントの数に比例して計算時間が増加するため，リアルタイムなバースト検出手法には不向きである．一方で，蝦名らの手法はイベントが発生するごとにバースト解析を行うため，イベントが発生していない期間の無駄な計算を削減することができる．また，一定期間内に複数イベントが発生した場合に，それらを 1 つのデータに圧縮してデータを保持することによって，イベントが集中して発生した際の膨大な計算を避けることができる．したがって，蝦名らの手法はリアルタイムなバースト検出を行うことが可能になり，鉄道運行トラブルという，リアルタイム性が重要なイベントの検出に適していると言える．

蝦名らの手法では，いくつかのパラメータを与える必要がある．パラメータはバーストの判定基準や，計算をどの程度を省くか，解析時間の最小単位などを決定する．本論文では，ツイートの投稿時間が秒単位で得られることから，解析時間の最小単位は 1 秒に設定した．残りのパラメータの調整を行うために，あるトラブル n に対してパラメータの組$\boldsymbol{\theta}$を与えたときの，バースト検出の結果を式 (1) によって評価する．

$$score_n(\boldsymbol{\theta}) = |T_{actual}(n) - T_n(\boldsymbol{\theta})| \tag{1}$$

ただし，$T_{actual}(n)$ は実際の運行トラブルの発生時間，$T_n(\boldsymbol{\theta})$ はパラメータ$\boldsymbol{\theta}$ が与えられたときのバースト検出時間である．ここで，時間は全て秒単位である．トラブルの総数を K とするとき，あるトラブル n に適用するパラメータは式 (2) のように決定する．

$$\arg\min_{\theta} \sum_{i=1, i\neq n}^{K} score_i(\boldsymbol{\theta}) \tag{2}$$

パラメータの探索はグリッドサーチによって行った．

5.2　評 価 実 験

2011 年 3 月から 2013 年 10 月までに東京メトロにおいて発生した運行トラブルを対象に評価実験を行った．検出結果は式 (1) によって評価する．表 3 に対象としたトラブルの数を示す．正解データには goo 路線運行情報(注7) を用いた．トラブル関連

	T_1	T_2	$T_1 + T_2$
Occurrence	52	39	88
Recovery	36	26	54

表 3　運行トラブルのケース数

ツイートの抽出には，対象とするトラブルに応じた分類器を用いた．ただし，復旧検出においては，いずれのトラブルに対しても R_1 を正例に学習させた分類器を用いた．

蝦名らの手法はバースト検出を行う前にデータ構造内に一定のデータを蓄積する必要がある．その際に，バースト検出対象の日に投稿されたツイートを用いると，その日の早い時間帯で実際のバーストがあった際にバースト検出ができなくなる．そこで，東京メトロが営業を行わない午前 1 時から午前 5 時までの約 4 時間を除いた 20 時間における，トラブル関連ツイートの平均到着間隔で予めデータを蓄積した状態で評価実験を行った．トラブル関連ツイートの平均到着間隔は式 (3) によって求める．

(注7)：http://transit.goo.ne.jp/unkou/

$$\frac{20 \times 3600}{AVG(m) \times FPR} \quad (3)$$

ここで，$AVG(m)$ は東京メトロの路線名 m を含むツイート数の平均値，FPR は分類器の False Positive Rate である．

比較手法には，トラブル関連ツイートの 5 分間隔のヒストグラムに対して閾値を設定し，その閾値を超えた区間の終了時間をバーストとする手法を用いた．閾値は，蝦名らの手法のパラメータと同様にして求めた．

実験結果を表 4 に示す．トラブルの発生検出では，T_1，T_1+T_2 においてバースト検出手法の評価値が閾値による手法に比べて 2 分から 3 分程度良く，バースト検出手法の有効性が確認できた．T_2 においては，バースト検出手法，閾値による手法のいずれも実際のトラブル発生時間との誤差が 20 分以上となっており，実用的でないことが分かった．復旧検出においては，バースト検出手法によって T_1 では 6 分以内，T_1+T_2 で 8 分以内に検出できた．しかし，いずれの場合も閾値による検出手法の方が早期に検出できており，バースト検出手法を用いることの有効性は確認できなかった．また，T_2 の検出においては，トラブルの発生検出の場合と同様に，バースト検出手法，閾値による検出手法のいずれも実用的でないことが分かった．

	Method	T_1	T_2	T_1+T_2
Occurrence	Burst Detection	**286.0**	1404.5	**254.0**
	Threshold	420.0	**1260.0**	420.0
Recovery	Burst Detection	313.0	**1903.0**	452.0
	Threshold	**241.0**	3539.0	**301.0**

表 4　バースト検出の評価値の中央値

一部見合わせの検出においてバースト検出手法が有効に機能しなかった原因について考察を行う．評価値が 1800 秒を超えた 14 件のケースについて解析したところ，主な原因は正解データの不正確さであった．ツイートの内容を確認したところ，正解データでは一部見合わせになっているが，実際には一旦全線で運転を見合わせた後に一部運転見合わせに変わっているというケースが 5 件見られた．この場合は，運行状況が全線見合わせから一部見合わせに切り替わる時間を正解時間にしない限り，一部見合わせの検出の評価を正しく行うことはできない．正確な正解データを準備して評価を行う必要がある．また，Twitter ではトラブルの発生についての投稿がなされているにもかかわらず，正解データではその時間帯は異常なしとされていたケースが 3 件あり，より詳細な運行情報に基づいて評価する必要があると考えられる．

次に，バースト検出手法が有効に機能した場合についての考察を行う．図 1 に 2012 年 4 月 3 日の東西線の運転見合わせの発生検出を行った際の見合わせ関連ツイートのヒストグラムおよび検出時間を示す．実際のトラブル発生時間付近にトラブル関連ツイートが最も集中して出現していることが分かる．このようなケースではバースト検出手法が有効であることが分かった．このケースでは実際のトラブル発生時間とバースト検出による検出時間の差は 85 秒である．一方で，閾値による検出手法では，実際のトラブル発生時間よりも早期にトラブル関連ツイートが集中した箇所を検出時間としており，誤差が 32280 秒と非常に大きい．

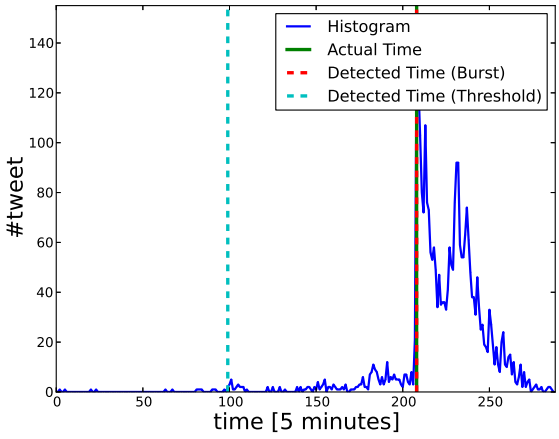

図 1　2012 年 4 月 3 日の東西線の運転見合わせのヒストグラムおよび検出時間

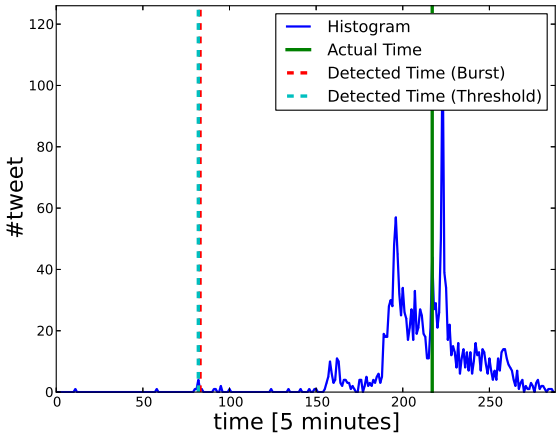

図 2　2011 年 9 月 21 日の東西線の全線見合わせのヒストグラムおよび検出時間

バースト検出手法が有効に機能しなかったケースとしては，2011 年 9 月 21 日に東西線で発生した全線見合わせが挙げられる．図 2 にこのケースの全線見合わせ関連ツイートのヒストグラムおよび検出時間を示す．このケースではバースト検出手法，閾値による検出手法いずれの場合も実際のトラブル発生時間よりも非常に早い段階で発生したバーストを検出している．このバーストは分類器の誤分類によって生じたものである．また，このケースはバーストしている箇所が複数見られた．正解時間の前のバーストは，一部で運転を見合わせたことに対して，単に「止まった」としか述べていないツイートが多く投稿されたことによって生じたものである．また，正解時間の後のバーストは，全線見合わせの情報が遅れて拡散されたことによって生じたものである．このように，複数のバーストが存在する場合は正解時間を検出するのは困難であることが確認できた．誤分類によるバーストは分類器の性能を良くすることによって解決できるが，他のバーストを検出しないようにすることは困難で

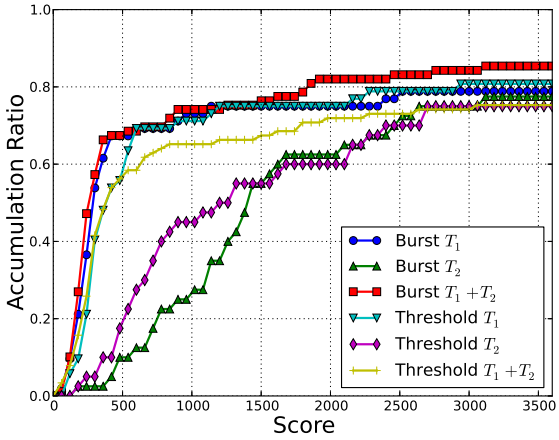

図 3　評価値の 1 時間以内の分布

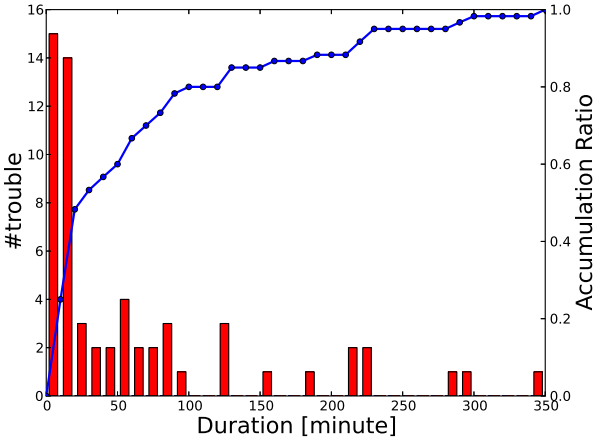

図 4　継続時間の分布

あると考えられる．

最後に，トラブル発生検出の評価値の累積分布を図 3 に示す．バースト検出によって，T_1 は，5 分以内に全体の 5 割強，T_1+T_2 は 5 分以内に全体の 6 割弱のトラブルを検出できたことが確認できた．一方で，T_2 ではバースト検出手法，閾値による手法いずれも 5 分以内に検出できたケースが全体の 1 割にも満たなかった．また，累積分布が 8 割程度のところで平坦になっていることから，バースト検出が有効に機能するケースは実験データ全体の 8 割程度であり，残りの 2 割は図 2 で示した例のように検出が困難であると考えられる．

6. 鉄道運行トラブルの継続時間の予測

本章では，運行トラブル発生後にそのトラブルが収束するまでの時間を予測する手法について提案し，見合わせを対象にした実験によって評価を行う．

6.1 提案手法

トラブル発生後 5 分の間に投稿されたトラブル関連ツイートから，そのトラブルが t 分以上継続するかどうかを判断する．本論文ではトラブル発生後，継続時間を予測する問題を，継続時間がある一定時間以上かどうかでトラブルを分類する問題として捉える．まず，トラブルが発生してから 5 分間に投稿された路線名を含むツイートから，4.1 節で作成した分類器によってトラブル関連ツイートを取得する．次に，取得したトラブル関連ツイートをすべて合わせて 1 つのドキュメントとみなし，ドキュメントごとに特徴量を求める．特徴量には，ツイートにトラブルの継続時間を示唆するような内容が含まれると仮定し，ドキュメントの中に含まれる形態素を用いる手法 (以下，WA と呼ぶ) と，ドキュメントの中の形態素の出現回数を用いる手法 (以下，WC と呼ぶ) の 2 通りを実験する．トラブルの継続時間がある時間 t よりも長い場合を正例，短い場合を負例とした学習データによって，分類器を学習する．分類器には線形カーネルの SVM を用いた．

6.2 評価実験

2011 年 3 月から 2013 年 10 月までの間に東京メトロ 9 路線において発生した見合わせ 60 件を対象に評価実験を行った．正解データには goo 路線運行情報を用いた．これらのトラブルの継続時間についての統計を図 4 に示す．トラブルの継続時間に対して設定する閾値 t は 15 分，30 分，45 分，60 分の 4 通りとした．評価は leave-one-out 交差検定によって行う．ベースラインには，全て正解クラスが多いクラスと判定する手法を用いた．

Method	15 分	30 分	45 分	60 分
WA	0.633	0.650	**0.717**	**0.817**
WC	0.600	**0.667**	**0.717**	0.783
Baseline	**0.650**	0.533	0.583	0.650

表 5　継続時間の分類精度

Positive Feature(30 分未満)			Negative Feature(30 分以上)		
品詞	特徴量	重み	品詞	特徴量	重み
名詞	故障	0.132	名詞	人身	-0.131
名詞	一時	0.092	名詞	ユーザ名	-0.109
名詞	点検	0.082	名詞	停電	-0.079
名詞	ドア	0.077	名詞	勘弁	-0.076
名詞	ホーム	0.073	動詞	見合わせ	-0.074

表 6　$t=30$ 分における WA を用いた SVM の特徴量の重みの例

Positive Feature(15 分未満)			Negative Feature(15 分以上)		
品詞	特徴量	重み	品詞	特徴量	重み
名詞	銀座	0.119	名詞	車輌	-0.113
名詞	ホーム	0.118	名詞	停止	-0.094
名詞	点検	0.114	動詞	見合わせ	-0.085
名詞	ドア	0.113	名詞	町	-0.085
名詞	車両	0.107	名詞	勘弁	-0.074

表 7　$t=15$ 分における WA を用いた SVM の特徴量の重みの例

表 5 に分類精度を示す．$t=15$ 分においては，ベースラインが最も高い精度を示したが，その他においては提案手法の有効性が確認できた．また，用いる特徴量として WA を用いた方が WC よりも良い分類性能を示す傾向があることが分かった．

次に，表 6 に特徴量に WA を用いて 30 分以上長引くかどうかを判定する分類器の重みの例を示す．この分類器では，ホームドアの点検や車両故障による見合わせの場合は早期に復旧する傾向にあることや，人身事故や停電による見合わせは長引く

傾向にあるというような，一般的な知識を学習していることが重みから判断することができる．その結果，予測精度が良かったと考えられる．

一方で，提案手法が有効に機能しなかった，特徴量に WA を用いて 15 分以上長引くかどうかを判定する分類器の重みを表 7 に示す．15 分未満を判断する場合においては，30 分の場合と同様に有効な特徴量が確認できた．しかし，15 分以上の場合を見てみると，トラブルが長引くことを示唆するような特徴量は見られない．また，それぞれのクラスについて F 値は 15 分未満については F 値が 0.667 であるのに対し，15 分以上では F 値が 0.593 という結果であった．したがって，15 分以上のトラブル分類時の性能が悪く，提案手法が有効に機能しなかったと考えられる．

7. 鉄道運行トラブルの連鎖予測

本章では，鉄道運行トラブルが発生した際に，そのトラブルが他の路線に連鎖するかどうかを予測する手法について提案する．首都圏で発生した運行トラブル (見合わせ，遅延，直通運転中止などを全て同一とみなしたトラブル) を対象にした評価実験により，提案手法の有効性を示す．

7.1 実験データの作成

運行トラブルが他の路線に影響を与えたかどうかの正解データを作成するために，2011 年 6 月から 2013 年 10 月までに発生した運行トラブルの情報を goo 路線運行情報から取得した．

まず，取得した路線運行情報の中から他路線の影響を受けたことによって，運行トラブルが発生したケースを抽出する．他路線からの影響を受けたことが明確に記述されている例を以下に示す．

- 山手線は 22:38 頃、京浜東北根岸線内で発生した人身事故の影響で、現在も一部列車に遅れが出ています。
- 湘南新宿ラインは東海道本線内で発生した人身事故の影響で、一部列車に遅れが出ていましたが、12:30 現在、ほぼ平常通り運転しています。

運行情報は自然言語で書かれているが，文章が定型的である．そこで，パターンマッチングによって他の路線に影響を与えた運行トラブルの情報を抽出した．抽出する情報は，連鎖元の路線 x，連鎖先の路線 y，連鎖元のトラブル発生時間 t の 3 つである．適用したパターンは以下の 2 つである．1 つ目は，先に述べた例の 1 つ目に相当するもので，路線 y について「t 頃、x 線内で」という記述を含むパターンである．この場合，連鎖元となったトラブルの発生時間も分単位で取得することができる．2 つ目は，先に述べた例の 2 つ目に相当するもので，路線 y について,「x 線内で」という記述を含むパターンである．この場合，連鎖元となったトラブルの発生時間は取得することができない．そこで，連鎖元のトラブル発生時間を取得するために，この情報が掲載された日と同じ日に，この情報が掲載されるよりも早く掲載された路線 x のトラブル情報が存在するかを調べる．もしそのような路線 x のトラブル情報が存在し，そのトラブルの詳細に「t 頃」という記述がなされていた場合，この t を連鎖元のトラブル発生時間とする．

連鎖ありのデータ数	717
連鎖なしのデータ数	436
連鎖元の路線数	28
連鎖先の路線数	69

表 8　連鎖予測で用いるデータの統計

次に，他路線に影響を与えなかった運行トラブルを抽出する．まず，トラブルの詳細に「t 頃」という記述が存在し，かつ,「線内」という記述がないものを取得する．このような例を次に示す．

- 京浜東北根岸線は 22:19 頃、東京駅で発生した人身事故の影響で、現在も列車に遅れが出ています。

その後，先に述べた方法で連鎖ありと判定されたトラブルと重複しないかを確認し，連鎖ありのトラブルに含まれなかったものを，連鎖なしの正解データとする．

以上で述べた方法によって，連鎖あり，なしそれぞれの運行トラブルの正解データを作成した．連鎖元の路線として対象としたのは，首都圏の鉄道のうち，路線名を含むツイートを十分取得できる路線である．また，連鎖なしの正解データにしか含まれない路線は除外した．表 8 に作成した正解データの統計を示す．

7.2 提 案 手 法

本論文では，路線 x_i で発生したトラブルが路線 y_j へ連鎖するかどうかという予測を分類問題として捉える．まず，路線 x_i のトラブル発生後 5 分間に投稿された路線名を含むツイートから 4.1 節で作成した分類器によってトラブル関連ツイートを抽出する．抽出したトラブル関連ツイートを 1 つのドキュメントとみなし，特徴量を求める．特徴量にはツイートの中にトラブルが連鎖するかどうか示唆する内容が含まれると仮定して，ドキュメントの中に形態素が出現したかどうか (以下，WA と呼ぶ)，ドキュメントの中の形態素の出現回数 (以下，WC と呼ぶ) を用いた．さらに，ダイヤが密になる通勤時間帯などにおいては，他路線からの影響を受けやすいと仮定し，トラブルの発生時間 (以下 T と呼ぶ) も特徴量として用いる．次に，連鎖先の路線ごとに分類器を作成する．ここで，連鎖元と連鎖先のペアごとに分類器を作成しない理由は，学習データを十分に確保できないためである．分類器の学習方法の違いによって，2 種類の手法を提案する．

1 つ目の手法は，連鎖先の路線 y_j の分類器の学習に，全トラブルを用いる手法である．例として，連鎖元の路線として x_1，x_2，x_3，連鎖先の路線として y_1，y_2，y_3 が存在する場合を考える．図 5 のように，連鎖元から連鎖先に，連鎖率を重みとしたエッジを生成する．ここで，連鎖が発生しない場合は連鎖率は 0 としてエッジを生成する．このとき，y_1 の分類器の学習には連鎖元の路線 x_1，x_2，x_3 で発生したトラブル全てを用いる．正例 (連鎖あり) には，x_1 および x_2 で発生したトラブルのうち，y_1 に連鎖したトラブルを用いる．負例 (連鎖なし) には，x_1，x_2，x_3 で発生したトラブルのうち，y_1 に連鎖しなかったトラブルを用いる．ここで，負例には y_1 以外の路線に連鎖したトラブルも含まれる．例えば，x_1 から y_2 には連鎖したが，y_1

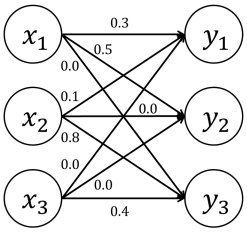

図 5　連鎖関係を表す有向グラフの例 (連鎖率あり)

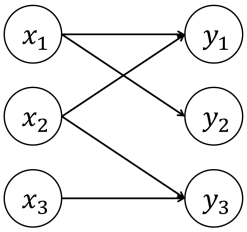

図 6　連鎖関係を表す有向グラフの例 (連鎖率なし)

には連鎖しなかったようなトラブルは，y_1 の分類器の学習では負例として扱う．予測を行う際は，全ての分類器を用いる．すなわち，連鎖先の路線すべてに対して，連鎖するかどうかの予測結果が得られる．この手法は全ての連鎖パターンの予測を行うことができる一方で，負例の数が多くなりすぎるという欠点がある．

2 つ目の手法は，連鎖先の路線 y_j の分類器の学習に，y_j の連鎖元になった路線のトラブルのみを用いる手法である．例として，1 つ目の例と同じ状況を考える．図 6 のように，連鎖が発生した場合のみ連鎖元から連鎖先にエッジを生成する．このとき，y_1 の分類器の学習には x_1，x_2 で発生したトラブルのみを用いる．正例には x_1 および x_2 で発生したトラブルのうち，y_1 に連鎖したトラブルを用いる．負例に用いるデータは x_1 および x_2 で発生したトラブルのうち，y_1 に連鎖しなかったトラブルである．予測を行う際は，学習時に連鎖先となったことがある路線の分類器のみを用いる．例えば，x_1 で発生したトラブルの連鎖予測を行う場合は y_1 および y_2 の分類器，x_2 で発生したトラブルの連鎖予測を行う場合は y_1 および y_3 の分類器を用いる．この手法は，負例の数を少なくすることができるが，学習時に発生しなかった連鎖パターンを予測することはできない．例えば，テストデータに $x_1 \rightarrow y_3$ という連鎖が存在する場合は，予測を行うことができない．

7.3　評 価 実 験

評価実験では作成した正解データのうち，2011 年から 2012 年に発生した運行トラブルを学習データ，2013 年に発生した運行トラブルをテストデータとして用いる．評価は連鎖元から連鎖先のエッジ単位で行う．ここで，分類器を作成する際に，学習データの正例，負例の数がそれぞれ 5 件以上ずつ存在しない路線は評価の対象から除外した．トラブルを限定して学習する手法では，学習データに含まれない連鎖パターンに対して予測を行うことができない．そのため，学習データに含まれる連鎖元の路線から新たな路線への連鎖は False Negative として扱う．また，学習データに含まれない路線が連鎖元となった場合は，その路線から連鎖するトラブルは False Negative，連鎖しないトラブルは True Negative として評価する．

Method	Precision	Recall	F-value
WA	0.587	0.456	0.513
WA+T	0.539	0.516	0.528
WC	**0.629**	0.544	**0.583**
WC+T	0.607	**0.554**	0.579

表 9　全トラブルを学習する手法の連鎖ありの分類結果

Method	Precision	Recall	F-value
WA	0.733	0.469	0.572
WA+T	0.736	0.496	0.593
WC	**0.749**	0.479	0.584
WC+T	0.738	**0.516**	**0.608**

表 10　トラブルを限定して学習する手法の連鎖ありの分類結果

Line	Method	Precision	Recall	F-value
都営浅草線	WC	1.000	1.000	1.000
京王線	WA	0.958	1.000	0.979
東京メトロ東西線	WC+T	1.000	0.682	0.811
東急田園都市線	WA+T	1.000	0.667	0.800
東京メトロ副都心線	WC+T	0.750	0.750	0.750

表 11　分類結果が良かった路線

Line	Method	Precision	Recall	F-value
京浜東北線	WA+T	1.000	0.118	0.211
山手線	WC	0.500	0.273	0.353
東急東横線	WA	1.000	0.231	0.375
横須賀線	WA	1.000	0.263	0.417
東海道線	WA+T	0.643	0.310	0.419

表 12　分類結果が悪かった路線

テストデータ全てに対する実験結果を表 9，10 に示す．特徴量は形態素の出現回数を考慮した方が有効であることが分かった．また，トラブルを限定して学習する手法では，特徴量にトラブル発生時間を加えることが分類精度向上に寄与した．さらに，トラブルを限定して学習する手法では，未知の連鎖パターンの予測を行えないため，全トラブルを学習する手法に比べて再現率は低いが，適合率と F 値は高いことが確認できた．トラブルを限定して学習する手法は，学習データを増やして，対応できる連鎖パターンを増やすことで，再現率の向上を期待することができる．

次に，トラブルを限定して学習する手法によって，連鎖元の路線ごとに予測を行った際の連鎖ありの分類結果の一部を表 11 および表 12 に示す．路線ごとに予測結果に大きなばらつきが見られた．連鎖ありの分類結果について見てみると，京王線や東西線の F 値が高いことが確認できる．これらの路線は共通して，連鎖パターンが単純であることが確認できた．例えば，京王線は 3 種類の路線に影響を与えるが，この 3 路線がいずれも京王線でトラブルが発生した場合，ほぼ全てのケースで影響を受けている．このように連鎖パターンが単純な路線に対する予測結果は良い傾向にあった．一方で，京浜東北線や山手線の連

鎖ありの F 値は低いことが分かる．京浜東北線や山手線はいずれも 10 路線以上に影響を与え，かつ，連鎖先の路線も多くの路線から影響を受ける．このように連鎖パターンが複雑な路線に対する予測においては，再現率が低いために F 値が下がる傾向にあった．複雑な連鎖パターンの予測を行うには，連鎖元と連鎖先のペアごとに分類器を作成するなど，他の手法を検討する必要がある．

8. まとめと今後の課題

本論文では，鉄道運行トラブル発生時の意思決定支援を目的に，Twitter を解析することによって首都圏の鉄道運行トラブルの検出および継続時間と連鎖の予測を行った．運行トラブルの検出では，蝦名らが提案したリアルタイムバースト検出手法を用いることで，運行トラブルの発生時間および復旧時間の検出を試みた．評価実験の結果，全線見合わせおよび見合わせの発生検出において，半数以上のトラブルを 5 分以内の誤差で検出できることが確認できた．トラブルの継続時間の予測においては，トラブル発生後 5 分間のトラブル関連ツイートを解析することによって，そのトラブルが一定時間以上長引くかどうかを判定する手法について提案し，評価実験を行った．実験結果から，30 分，45 分，60 分以上長引くかどうかを判定する場合に提案手法の有効性が確認できた．トラブルの連鎖予測においては，トラブル発生後 5 分間に投稿されたトラブル関連ツイートを解析し，そのトラブルが他路線に影響を及ぼすかどうかを予測する手法について提案した．評価実験の結果，路線によっては実用レベルで連鎖の予測が可能であることが確認できた．

今後の課題として，まずトラブル関連ツイートを抽出する分類器の性能向上が挙げられる．本論文では，ツイート中に出現する形態素のみを特徴量として用いたが，性能向上のために文脈を考慮できる特徴量を加えたり，路線ごとに分類器を作成するなど様々な方法を検討する必要がある．また，本論文で扱わなかった鉄道運行トラブルに関する情報抽出も今後の課題である．本論文では路線名を含むツイートのみを解析対象としたが，例えば駅名を含むツイートも解析対象に加えることで，運行トラブルの発生場所を抽出できる可能性もある．本論文で提案した手法の改良に加え，鉄道運行トラブル状況に関する様々な情報抽出の可能性を模索する必要がある．

文　献

[1] Eytan Adar, Daniel S. Weld, Brian N. Bershad and Steven D. Gribble. Why We Search: Visualizing and Predicting User Behavior. *WWW*, pages 161–170, 2007.

[2] Adam Sadilek, Henry Kautz and Vincent Silenzio. Predicting Disease Transmission from Geo-Tagged Micro-Blog Data. *AAAI*, 2012.

[3] Kira Radinsky, Sagie Davidovich and Shaul Markovitch. Learning Causality for News Events Prediction. *WWW*, pages 909–918, 2012.

[4] Panagiotis T. Metaxas and Eni Mustafaraj. How (Not) to Predict Elections. *IEEE third international conference on social computing*, 2011.

[5] Eric Gilbert and Karrie Karahalios. Widespread Worry and the Stock Market. *AAAI*, 2010.

[6] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science 2(1)*, pages 1–8, 2010.

[7] Daniel Gruhl, R. Guha, and Ravi Kumar. The predictive power of online chatter. *KDD*, pages 78–87, 2005.

[8] Haipeng Zhang, Mohammed Korayem and David J. Crandall. Mining Photo-sharing Websites to Study Ecological Phenomena. *WWW*, 2012.

[9] Takeshi Sakaki, Makoto Okazaki and Yutaka Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sonsors. *WWW*, pages 851–860, 2010.

[10] Jon Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery, 7(4)*, pages 373–397, 2003.

[11] Xin Zhang and Dennis Shasha. Better Burst Detection. *ICDE*, pages 146–149 2006.

[12] Yunyue Zhu and Dennis Shasha. Efficient elastic burst detection in data streams. *KDD*, pages 336–345, 2003.

[13] Dennis Shasha and Yunyue Zhu. High performance discovery in time series. *Techniques And Case Studies (Monographs in Computer Science)*, 2004.

[14] 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, 2010.