

修 士 論 文

マイクロブログを用いた鉄道運行トラブルの
検出と影響の予測

Detecting Train Troubles and Predicting
Their Impacts from Microblogs

指導教員

喜連川 優 教授



東京大学大学院情報理工学系研究科
電子情報学専攻

氏 名

48-126427 土屋 圭

提 出 日

平成26年2月6日

概要

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。さらに、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 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。

謝辞

はじめに，喜連川優教授に深く感謝致します．喜連川教授には，本研究に関して様々な助言を頂くとともに，情報理工学を学ぶ者にとってこの上ない研究環境を与えて下さいました．折に触れて喜連川教授がお話になることは，研究に関することのみならず，人生の指針となるようなことで大変貴重なものでした．

次に，豊田正史准教授に深く感謝致します．豊田准教授には本研究の方針から具体的な実験内容，さらには発表資料の作り方に至るまで非常に手厚くご指導して頂きました．平日の深夜や土日，年末年始など，どんな時でも常にご相談にのって下さいました．私の知識不足故の実験の不手際や，拙い考察に対しても決して怒ることなく，常に的確なアドバイスをして頂き，研究に対する意欲を失うことなく非常に有意義な研究生活を送ることができました．研究についてのみでなく，Linux マシンの基本的な知識や，プログラミング方法まで，豊田准教授から学んだことは数え切れません．豊田准教授に心から感謝申し上げます．本当にありがとうございました．

修士ミーティングやNLP ミーティングにおいて，研究に関する議論やより良い発表のため，時には深夜までご指導して下さいました，鍛冶伸裕特任准教授，吉永直樹特任准教授，中野美由紀特任准教授，伊藤正彦助教，横山大作助教に深く感謝致します．鍛冶特任准教授には，常に的確かつ鋭いアドバイスを頂きました．また，締切当日にもかかわらず論文の添削をして頂くなど大変お世話になりました．吉永特任准教授には，研究の議論や発表練習などはもちろん，大学周辺のおいしいお店に連れて行って頂くなど，研究以外の面でも非常にお世話になりました．中野特任准教授からは研究に対する姿勢について学ぶことが多かったです．中野特任准教授は非常に明るくて，ミーティングの雰囲気がとても良く，厳しいご指摘を頂きなが

らも、気分を落とすことなく議論することができました。伊藤正彦助教には本研究に関する様々なアドバイスを頂きました。また、研究会の発表先での観光案内や飲み会など、研究以外の面でも非常にお世話になりました。横山大作助教には本研究に関して大変有益なアドバイスを数多く頂きました。また、本研究を進める上で非常に重要なデータを提供して頂きました。全ての先生方が非常に熱心にご指導して下さい、たくさんのことを学ぶことができました。喜連川・豊田研究室で2年間を過ごすことができた私はとても幸せでした。先生方に心から感謝申し上げます。

一緒に研究生活を共にした同期の岡本君、栗原君、鈴木君、仁科君に感謝致します。同期の皆と他愛のない話をしたり、一緒にご飯を食べたりして過ごす時間はかけがえのないものでした。また、様々なアドバイスを下さった先輩方、打ち上げ企画など気を配ってくれた後輩たちに感謝致します。そして、快適な研究生活を支えて下さった秘書の皆様に深く感謝致します。

最後に、いつも山形から支えてくれた家族に感謝の意を捧げます。

2014年2月6日

目次

謝辞	i
第1章 はじめに	1
1.1 ウェブ解析による実世界のイベント抽出の重要性	1
1.2 本研究の目的と貢献	2
1.3 本論文の構成	3
第2章 関連研究	5
2.1 人の行動・状態を対象とした研究	5
2.2 社会・経済活動を対象とした研究	6
2.3 自然現象を対象とした研究	7
第3章 鉄道運行トラブル関連ツイートの抽出	8
3.1 路線名を含むツイートの取得	8
3.2 分類器の作成	8
3.2.1 分類器の性能評価	13
3.2.2 誤分類の考察	16
第4章 鉄道運行トラブルの検出	19
4.1 バースト検出手法	19
4.1.1 データ構造	20
4.1.2 バースト判定方法	21
4.2 パラメータの調整	22
4.3 評価実験	23

4.3.1	実験設定	23
4.3.2	実験結果と考察	25
第 5 章	鉄道運行トラブルの継続時間予測	31
5.1	初期段階における継続時間予測	31
5.1.1	提案タスクと予測手法	31
5.1.2	評価実験	32
5.2	継続時間の逐次予測	35
5.2.1	提案タスクと予測手法	35
5.2.2	評価実験	36
第 6 章	鉄道運行トラブルの連鎖予測	39
6.1	実験データの作成	39
6.2	鉄道運行トラブルの連鎖予測手法	41
6.3	評価実験	44
6.3.1	実験設定	44
6.3.2	実験結果と考察	47
第 7 章	おわりに	51
	参考文献	53
	発表文献	55

目 次

4.1	Aggregation Pyramid の構造例	20
4.2	レベル 0 のセルの生成方法	21
4.3	レベル 1 以上のセルの生成方法	21
4.4	バースト検出アルゴリズム	22
4.5	トラブル発生検出における評価値の分布	26
4.6	トラブル復旧検出における評価値の分布	27
4.7	2012 年 4 月 3 日東西線の運転見合わせのヒストグラムおよび検出時間	28
4.8	2011 年 9 月 21 日東西線の全線見合わせのヒストグラムおよび検出時間	29
4.9	2011 年 7 月 6 日半蔵門線の一部見合わせのヒストグラムおよび検出 時間	30
5.1	継続時間の分布	32
6.1	路線間の連鎖関係を表す有向グラフ	42
6.2	連鎖関係を表す有向グラフの例	43
6.3	連鎖関係を表す有向グラフの例 (連鎖率あり)	44

第1章 はじめに

1.1 ウェブ解析による実世界のイベント抽出の重要性

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 [1]。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 [2]。また、Twitter の約1億4千万人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 [3]。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し得る、より詳細な情報を含むという特徴がある。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そのイベントが発生したことはもちろん、イベントの状況に関して、現場にいないと分からないような情報が投稿される。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。

ソーシャルメディアからのイベント抽出に関する先行研究は数多く存在するが、

1.2. 本研究の目的と貢献

意思決定を支援できる情報を抽出している研究はあまりなされていない。対象とするイベントの基本的な情報抽出のみでなく、具体的な意思決定に役立つ情報抽出はこの分野の課題であると言える。

1.2 本研究の目的と貢献

本論文では、ソーシャルメディアの1つである Twitter を解析することによって、**鉄道運行トラブル発生中のリアルタイムな意思決定支援**を行えるような情報の抽出を目指す。

鉄道運行トラブルが発生した際、数千数万人規模の人が分単位の意思決定に迫られる。例えば、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。このような意思決定を支援するために、本研究では大きく4つのタスクを提案する。

1つ目はトラブル関連ツイートの抽出である。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は運行トラブルについて述べているものを含んでいるため、運行トラブル状況を把握する上で重要な手掛かりとなる。一方で、路線名を含んでもトラブルとは関係のないツイートも多く存在する。したがって、路線名を含むツイートがトラブルに関連があるかどうかを分類する必要がある。そこで、トラブル関連ツイートを抽出するための分類器を作成し、性能評価を行った。分類器はトラブル発生状況、復旧状況、混雑状況に関連するツイートを抽出するものをそれぞれ作成した。

2つ目は、トラブル発生後あるいは復旧後の早期対応を支援するためのトラブルの発生および復旧の検出である。トラブル関連ツイート数の時系列変化に対してバースト検出手法を適用することによって、トラブルの発生時間および復旧時間の検出を行った。評価実験では東京メトロ9路線で発生した全線見合わせ、一部見合わせ、全線および一部見合わせを同一とみなしたトラブル(以下、見合わせと呼ぶ)の3種類を対象に、実際の時間と検出した時間にどれくらいの誤差があるのかを確認した。また、バースト検出手法を用いることの有効性を確認するため、トラブル関連ツイート数がある閾値を超えた時点をトラブル発生時間とする手法との比較も行った。

1.3. 本論文の構成

3つ目はトラブル継続時間の予測である。トラブルがどの程度長引くかを知ることができれば、「少し時間が経ってから駅に向かう」、「今後の予定を取りやめる」などの具体的な意思決定が可能になる。継続時間の予測は、トラブル発生後初期段階で予測を行う手法と、トラブル発生後数分間隔で逐次予測を行う手法の2つを提案する。評価実験では東京メトロ9路線で発生した運転見合わせを対象に、実際の継続時間と予測結果の比較を行った。

4つ目はトラブルの連鎖予測である。発生しているトラブルがどの路線に影響を及ぼすかを知ることができれば、他ルートを検討する際に役立つ。トラブルの連鎖予測では、トラブル発生後一定時間内に投稿されたトラブル関連ツイートを解析することによって、ある路線で発生したトラブルが他の路線に影響を及ぼすかどうかの予測を行う。評価実験では首都圏の鉄道路線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トラブルとみなしたもの)を対象に、実際の連鎖状況と予測結果の比較を行った。

1.3 本論文の構成

本論文の構成は次のとおりである。

第2章 ウェブ解析によって実世界のイベント抽出を行った最新の研究動向について説明し、本論文との違いについて述べる。

第3章 鉄道の路線名を含むツイートから運行トラブルに関連するツイートを抽出する方法について述べ、評価実験を行う。

第4章 バースト検出手法を用いて運行トラブルの発生および復旧の検出を行い、評価実験によって早期に検出できるか確認する。

第5章 運行トラブルの継続時間を予測する手法について提案し、評価実験を行う。

1.3. 本論文の構成

第6章 運行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し，評価実験を行う．

第7章 全体のまとめと今後の課題について述べる．

第2章 関連研究

本章では，ウェブを解析して実世界のイベント抽出を行った先行研究を，対象としているイベントの種類によって，人の行動・状態，社会・経済活動，自然現象の3つに分けて述べる．また，それぞれの先行研究がどのウェブコンテンツを解析し，どのような情報を抽出しているかについて説明し，本論文との違いについて言及する．

2.1 人の行動・状態を対象とした研究

人の行動パターンや状態などを対象にした研究として，Sadilek ら [4] らの研究がある．Sadilek らは，Twitter 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．病気が伝播する最も重要な要因の一つは人同士の接触である [5] ことに着目し，Twitter の投稿内容，投稿位置およびフォロー・フォロワー関係を観測情報とした確率モデルを用いている．評価実験では一週間以上先の未来における人の健康状態を予測し，有効性を示した．Sadilek らの研究は，健康状態の予測を個人単位という非常に細かい粒度で行っているが，病状といった病気の状態や程度については言及しておらず，あらゆる病気を1つに丸めて扱っているという点で本研究とは異なる．また，実世界のイベント抽出という観点からは外れるが，Adar ら [6] によるインターネットユーザの行動を解析した研究がある．Adar らはインターネットユーザの過去の行動履歴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．検索エンジンのクエリを解析対象としており，テレビ番組情報の出現などの時系列との相関の解析を行い，今後の動向の予測を行った．また，複数時系列の違いなどを見やすくするような可視化インターフェースの作成も行った．

2.2 社会・経済活動を対象とした研究

社会の動きを対象とした研究として、Radinsky ら [7] や Panagiotis ら [8] 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。Radinsky らは LinkedData[9] オントロジーを用いてイベント間の類似度を定義し、イベントが与えられたときに、そのイベントに最も類似するようなイベントを求め、求めたイベントから引き起こされたイベントを予測結果としている。評価実験によって、提案手法が人と同程度の精度で予測できることが示された。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。ツイート数やツイートの感情分析を行い、ランダムに予測した場合に比べ、良い精度で予測ができることが確認できた。これらの研究はいずれも未来に起こるイベントを予測したという点において、本論文における鉄道運行トラブルの継続時間や連鎖の予測に類似する。一方で、本論文ではリアルタイムに発生しているイベントを対象にしており、この点においてこれらの研究とは異なる。

次に、経済活動を対象とした研究として、Gilbert ら [10] や Bollen ら [11], Daniel ら [12] 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。Gilbert 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。6 種類の感情のうち、平穏の感情変動とダウ平均株価に高い相関が見られ、この相関を用いた手法を提案している。Daniel らは、ある書籍のブログ上での言及回数と、Amazon¹におけるその書籍の売り上げに相関があることに着目し、ある書籍について言及しているブログの記事数からその書籍の売上を予測する手法を提案した。これらの研究もやはり将来のイベントの予測に焦点を当てており、リアルタイムに発生しているイベントを対象にしていない点において本論文とは異なる。

¹<http://www.amazon.co.jp/>

2.3 自然現象を対象とした研究

自然現象を対象とした研究として、Haipeng ら [13] や Sakaki ら [14] の研究がある。Haipeng らは写真共有サイトである Flickr 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。Haipeng らは雪というものが写真に頻繁に映りこむということに着目し、写真に付与されたタグのみでなく、画像自体の特徴量も解析対象としている。実験結果では、都市部など写真を投稿するユーザーが多い地域においてある程度正確に現象の観測ができることが確認できた。また、衛星写真などでは雲などの影響で測定できなかった地域についても、Flicker のデータを用いて積雪の有無の観測ができた。Haipeng らの研究は現在発生しているイベントを対象にしている。さらに、積雪の有無だけではなく積雪量についての言及もなされており、本研究に類似していると言える。しかし、積雪は比較的長時間同じ状態が継続するのに対して、本論文で扱う鉄道運行トラブルは分単位で状況が変わるため、リアルタイムな解析が必要となる。この点において Haipeng らの研究は本論文とは異なる。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。対象とするイベントをリアルタイムに特定するために、Twitter のリアルタイム性を利用している。位置の推定にはカルマンフィルタおよび粒子フィルタを用いた。Sakaki らは地震を検知するアプリケーションも作成しており、地震発生後 20 秒以内にユーザに地震発生の通知を送信することに成功している。この研究は対象とするイベントをリアルタイムに検出している点において本論文と類似している。一方で、地震の震度や台風の規模、あるいはそれらによって生じた被害状況についての言及はなされておらず、イベント発生時の意思決定を支援することはできない。この点で本論文とは異なる。

第3章 鉄道運行トラブル関連ツイートの抽出

3.1 路線名を含むツイートの取得

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は運行トラブルについて述べているものを含んでいるため、運行トラブル状況を把握する上で重要な手掛かりとなる。そこで、Twitter API¹ を用いて、2011 年 3 月から 2013 年 12 月までの間に、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、首都圏の路線名を含むツイートを抽出した。表 3.1 に、ツイート数の日平均が約 100 を超えている路線のツイートの統計を示す。

3.2 分類器の作成

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について述べているものが含まれている。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のないツイートや、“今夜は千代田線遅れるだろう” のように現在の運行トラブルについて述べていないツイートも存在するため、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分類する必要がある。そこでツイートがトラブルに関連しているかどうかを判定する分類器を作成した。分類器には線形カーネルの SVM、また、SVM のライブラリには LIBSVM²を用いた。特徴量には、トラブルの状況がツイートの内容に表れるこ

¹<https://dev.twitter.com/docs/api/1.1>

²<http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

3.2. 分類器の作成

路線名	合計	日平均	標準偏差
山手線	1874576	1825.29	1739.53
中央線	963693	938.36	1365.35
総武線	535786	521.70	819.95
京浜東北線	504963	491.69	865.87
常磐線	391994	381.69	576.12
東海道線	419342	408.32	663.77
埼京線	466892	454.62	639.38
京葉線	318462	310.09	707.92
横須賀線	252541	245.90	533.79
湘南新宿ライン	261240	254.37	365.13
南武線	199161	193.93	345.21
武蔵野線	307607	299.52	724.95
横浜線	208959	203.47	328.81
高崎線	184257	179.41	417.93
小田急線	339171	330.25	607.64
東急東横線	598514	582.78	1415.78
東急田園都市線	287005	279.46	431.69
東京メトロ千代田線	193013	187.94	262.09
東京メトロ副都心線	231844	225.75	570.98
東京メトロ銀座線	266823	259.81	1099.56
東京メトロ半蔵門線	157115	152.98	737.62
東京メトロ日比谷線	197375	192.19	293.58
東京メトロ丸ノ内線	186695	181.79	477.88
東京メトロ南北線	107595	104.77	289.41
東京メトロ東西線	333411	324.65	516.74
東京メトロ有楽町線	177064	172.41	306.09
都営大江戸線	252459	245.82	1002.39
都営新宿線	285591	278.08	632.80
都営浅草線	117413	114.33	539.06
都営三田線	98302	95.72	510.65
京王線	481509	468.85	919.96
京王井の頭線	203242	197.90	349.01
東武東上線	242509	236.13	313.15
東武宇都宮線	159492	155.30	372.41
西武池袋線	157136	153.00	296.65
ゆりかもめ	201373	196.08	393.65
りんかい線	224254	218.36	523.26

表 3.1: 路線名を含むツイートの統計

3.2. 分類器の作成

とから、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 MeCab³, MeCab の辞書には IPA 辞書⁴ を用いた。

SVM の学習データを作成するために、5000 件のツイートに対して人手でラベル付けを行った。これらのツイートは 2011 年 9 月 21 日⁵, 2012 年 4 月 3 日⁶, 2012 年 9 月 30 日⁷ の 3 日間に投稿された、東京メトロ 9 路線のいずれかの路線名を含むツイートの中からランダムにサンプリングしたものである。ラベルは運行トラブルの発生状況、復旧状況、混雑状況の 3 つに対して、それぞれ独立に付与する。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 : 復旧について言及なし, R_1 : 復旧について言及ありの 2 種類, 混雑状況のラベルは C_0 : 混雑状況について言及なし, C_1 : 混んでいる, C_2 : 空いているの 3 種類である。ラベル付けの結果を表 3.2 に示す。また、ラベル付けを行ったツイートの例を各ラベルごとに 3 つずつ示す。

	T_0	T_1	T_2	T_3	R_0	R_1	C_0	C_1	C_2
#tweet	3010	868	461	661	4745	255	4488	328	184

表 3.2: 人手によるラベル付けの結果

- T_0 , R_0 , C_0 : 言及なし
 - 飯田橋の**有楽町線**と**東西線**乗り換えが今日一番疲れた原因だと思う
 - **有楽町線**の座席固めでいいね
 - 結果論として、現状のうちにとってはあのまま新幹線で東京出てそこから**丸ノ内線**で荻窪か、さっきのルート of 渋谷から井の頭で吉祥寺、ってのが正解に近いルートだったかな。

³<http://mecab.sourceforge.net/>

⁴<http://code.google.com/p/mecab/downloads/detail?name=mecab-ipadic-2.7.0-20070801.tar.gz>

⁵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。

⁶強風の影響で銀座線を除く全路線で遅延などが生じた。

⁷台風 17 号の影響で 6 路線で遅延などが生じた。

3.2. 分類器の作成

- T_1 : 全線運転見合わせ
 - － 【列車運行情報】 メトロ**有楽町線**、**副都心線**、全線で運転見合わせ。
#NHK #twiphoon #台風 17:32 更新
 - － 中目黒駅の混雑で**日比谷線**運転見合わせ。どんだけカオス
 - － **東西線**完全終了のおしらせ…木場で身動き取れん www
- T_2 : 一部区間運転見合わせ
 - － まさかの**千代田線**表参道止まり wwwwww どうやって帰るんよ…
 - － 東武線が人身事故で**半蔵門線**渋谷～押上のみで運転って w
 - － **東西線**が中野～東陽町間折り返し運転、東陽町～西船橋間は運転中止。
まあ明日朝の出勤には十分復旧するでしょう。
- T_3 : 遅延や直通運転中止など運転見合わせ以外の異常
 - － **有楽町線**すいてるぜひやっほーい!! と思ったら、1 駅行ったところで、運転間隔調整 w
 - － 東京メトロ**南北線**【運転状況】台風の影響で、列車に遅れが出ています。
なお、東急目黒線との直通運転を中止しています。(09/30 20:10)
 - － **銀座線**、動いてましたが、のろのろ運転。でも、乗ってれば着くもんね
♪ (´ ▽ `) 止まってるよりマシです。
- R_1 : 復旧について言及あり
 - － **副都心線**再開！新宿脱出！
 - － **丸ノ内線**は動き出しました！赤坂見附まで出られる！
 - － 行徳に到着。そして**東西線**復旧。なら、苦労せずとも東陽町で待ってりや良かったじゃんという突っ込みもありかもしれませんが、そこはそれ。非日常を楽しむくらいの心の余裕も欲しいよねってことで。まあ、疲れたのは事実ですが。

3.2. 分類器の作成

- C_1 : 混んでいる
 - 東西線座れんかった(´Д`)
 - とりあえず千代田線、総武線はすげー混んでるよ
 - 丸ノ内線の荻窪行き、中央線がアウトだからすげー混んでた(´Д`)ぎゅうぎゅう。支線は運良く座れたけど、神田川の水位が心配だわ…
- C_1 : 空いている
 - 丸ノ内線ガラガラ～
 - 千代田線すげー空いてる。湯島から下りで始めて座れた。
 - 銀座線なんでこんな空いてるのお(°▽°)雨降ってないし、ラッキー♪

次に、運転トラブル発生状況についてのみ、運行トラブルが発生した時間帯が既知の場合に、その時間帯に投稿されたツイート全てを正例とした学習データを作成し、これを用いた半自動分類もあわせて試みる。例えば、千代田線が全線見合わせを行った時間帯を取得できた場合は、その時間帯に投稿された「千代田線」を含むツイート全てに対して T_1 のラベルを付与する。半自動の結果の有効性が確認できれば、過去の運行状況だけを用いてツイートの分類ができ、人手でラベル付けを行う手間が大幅に削減される。半自動のラベル付けは、人手でラベルを付けた場合と同様に、2011年9月21日、2012年4月3日、2012年9月30日に投稿された東京メトロ9路線のいずれかの路線名を含むツイートに対して行った。半自動のラベル付けの結果を表3.3に示す。

	T_0	T_1	T_2	T_3
#tweet	5986	5270	8970	12308

表 3.3: 半自動のラベル付けの結果

3.2. 分類器の作成

3.2.1 分類器の性能評価

ラベル付けを行ったツイートを学習データとして分類器の性能評価を行った。人手でラベル付けを行ったツイートを学習データとした分類器は、10 分割の交差検定によって評価を行った。半自動でラベル付けを行ったツイートを学習データとした分類器は、人手でラベル付けを行ったツイートをテストデータとして評価を行った。ベースラインには全てのツイートを正例とする手法を用いた。なお、ツイートの正規化のため、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。

人手でラベルを付けたツイートで学習した SVM の特徴量と重みの例を表 3.4, 3.5, 3.6, 3.7, 3.8, 3.9 に示す。正例とした運行トラブルの発生を示唆する形態素の重みが大きくなっていることが確認できた。一方で、表 3.5 の Positive Features に見られるように、特定の駅名や路線名に対する重みが大きくなっていることも確認された。このことから、ラベル付けを行ったツイートに記述されている特定のトラブルに特化した学習が行われたと考えられ、分類器の汎化能力の低下が懸念される。

Positive Features			Negative Features		
品詞	特徴量	重み	品詞	特徴量	重み
動詞	まっ	2.028	名詞	一部	-1.576
動詞	止まっ	1.956	助詞	たり	-1.162
名詞	見合わせ	1.852	助動詞	なかっ	-1.152
動詞	とまっ	1.782	副詞	ほとんど	-1.135
動詞	見合せ	1.752	名詞	有楽町線	-1.100
動詞	とまり	1.631	名詞	東陽	-1.051
名詞	ストップ	1.540	連体詞	こういう	-1.049
動詞	見合わせ	1.406	助詞	けども	-1.041
名詞	運休	1.260	名詞	再送	-1.021
副詞	なんで	1.246	副詞	よく	-1.002

表 3.4: T_1 を正例としたときの SVM の特徴量の重み

3.2. 分類器の作成

Positive Features			Negative Features		
品詞	特徴量	重み	品詞	特徴量	重み
名詞	一部	2.025	名詞	日比谷線	-1.497
動詞	折り返し	1.420	名詞	南北	-1.305
名詞	折り返し	1.367	名詞	東西線	-1.163
名詞	東陽町	1.283	名詞	模様	-1.146
名詞	どまり	1.226	動詞	下さい	-1.004
名詞	途中	1.217	名詞	ふしぎ	-1.000
名詞	おれ	1.140	動詞	止ま	-0.943
名詞	有楽町線	1.095	動詞	言っ	-0.931
名詞	、	1.084	名詞	0	-0.930
名詞	東陽	1.030	名詞	メーン	-0.889

表 3.5: T_2 を正例としたときの SVM の特徴量の重み

Positive Features			Negative Features		
品詞	特徴量	重み	品詞	特徴量	重み
名詞	調整	2.099	名詞	QT	-1.950
名詞	死亡	1.851	感動詞	あれ	-1.430
名詞	速度	1.820	名詞	半蔵門	-1.220
動詞	遅れ	1.580	動詞	まっ	-1.114
動詞	乱れ	1.571	名詞	票	-1.111
動詞	死ん	1.511	動詞	折り返し	-1.075
名詞	しん	1.449	名詞	風力	-1.018
動詞	やら	1.424	名詞	じき	-1.001
名詞	ダメ	1.413	感動詞	おやすみなさい	-1.000
名詞	お知らせ	1.391	動詞	なくなっ	-1.000

表 3.6: T_3 を正例としたときの SVM の特徴量の重み

表 3.10 に分類器の性能評価の結果を示す. 運行トラブル状況では, 全ての場合において人手の分類器が最も高い F 値を示した. 適合率については, 半自動が人手よりも高い値を示す場合があった. 特に, $T_1 + T_2$, $T_1 + T_2 + T_3$ を正例とした場合は半自動によって 8 割以上の適合率が得られた. また, 半自動とベースラインの F 値を比較すると, 全ての場合において半自動の方が高い結果であったが, 手動と同程

3.2. 分類器の作成

Positive Features			Negative Features		
品詞	特徴量	重み	品詞	特徴量	重み
名詞	再開	2.228	動詞	潰す	-1.000
動詞	動き出し	1.832	動詞	頼む	-1.000
名詞	復活	1.764	助動詞	ない	-0.992
名詞	復旧	1.666	助詞	なあ	-0.823
名詞	回復	1.647	助動詞	だっ	-0.811
名詞	開通	1.227	名詞	小田急	-0.806
名詞	開始	1.078	名詞	orz	-0.765
副詞	やっと	1.054	動詞	がんばれ	-0.756
動詞	動き	1.017	動詞	する	-0.733
名詞	一安心	1.000	名詞	22	-0.714

表 3.7: R_1 を正例としたときの SVM の特徴量の重み

Positive Features			Negative Features		
品詞	特徴量	重み	品詞	特徴量	重み
名詞	混雑	1.951	名詞	レベル	-1.031
動詞	混ん	1.634	名詞	通常	-1.014
動詞	混み	1.632	名詞	程度	-1.000
名詞	ラッシュ	1.458	動詞	で	-0.972
名詞	満員	1.211	名詞	ホテル	-0.925
動詞	こみ	1.111	副詞	まあ	-0.855
名詞	人	1.054	名詞	危険	-0.850
名詞	千代田線	1.000	副詞	まだ	-0.831
名詞	ごみ	1.000	名詞	具合	-0.824
名詞	列	0.985	名詞	京橋	-0.809

表 3.8: C_1 を正例としたときの SVM の特徴量の重み

度の F 値は確認できず，今回の実験においては半自動の手法は実用的でないことが分かった．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．

3.2. 分類器の作成

Positive Features			Negative Features		
品詞	特徴量	重み	品詞	特徴量	重み
動詞	空い	2.191	名詞	コミコミ	-0.925
名詞	ガラガラ	2.076	名詞	日比谷線	-0.912
動詞	すい	1.798	動詞	死ん	-0.821
名詞	がら空き	1.769	助詞	なあ	-0.595
名詞	がらがら	1.572	名詞	せい	-0.588
動詞	座れ	1.196	名詞	お腹	-0.587
動詞	座れる	1.037	名詞	台風	-0.583
動詞	すかす	1.000	名詞	フラグ	-0.575
動詞	すいとる	1.000	名詞	スカイツリーライン	-0.561
動詞	で	0.920	名詞	地獄	-0.561

表 3.9: C_2 を正例としたときの SVM の特徴量の重み

復旧状況および混雑状況については、いずれの場合も F 値において人手がベースラインを上回った。また、正例と負例の数の偏りが非常に大きいにもかかわらず、F 値は約 0.65 から 0.70 程度であった。

3.2.2 誤分類の考察

T_1 を正例とした分類器の結果を例に、誤分類について考察する。以下に偽陽性 (誤って全線見合わせと判定) と偽陰性 (誤って全線見合わせではないと判定) のツイートの例を 3 つずつ示す。なお、ツイートの正規化の際に “QUERYWORD” に置き換えた路線名を太字で示す。

- 偽陽性
 - － **東西線**止まっちゃったの？
 - － メトロ運行情報 【遅延】**銀座線**、丸ノ内線、日比谷線 (折り返し運転)、**【運転見合わせ】** 東西線、千代田線、有楽町線、半蔵門線 (人身事故)、南北線、副都心線

3.2. 分類器の作成

Target Label	Method	Precision	Recall	F-value
T_1	人手	0.695	0.666	0.680
	半自動	0.426	0.689	0.527
	ベースライン	0.174	1.000	0.296
T_2	人手	0.607	0.573	0.589
	半自動	0.698	0.371	0.485
	ベースライン	0.092	1.000	0.169
T_3	人手	0.583	0.534	0.557
	半自動	0.416	0.184	0.255
	ベースライン	0.132	1.000	0.234
$T_1 + T_2$	人手	0.715	0.711	0.713
	半自動	0.850	0.567	0.680
	ベースライン	0.266	1.000	0.420
$T_1 + T_2 + T_3$	人手	0.759	0.746	0.752
	半自動	0.959	0.434	0.597
	ベースライン	0.398	1.000	0.569
R_1	人手	0.710	0.671	0.690
	ベースライン	0.051	1.000	0.097
C_1	人手	0.674	0.619	0.645
	ベースライン	0.066	1.000	0.123
C_2	人手	0.789	0.652	0.714
	ベースライン	0.037	1.000	0.071
$C_1 + C_2$	人手	0.743	0.684	0.712
	ベースライン	0.102	1.000	0.097

表 3.10: 10 分割交差検定の結果

- － わーん！中央線も総武線も東西線も止まってる！東西線は地下鉄のくせに何で止まってるんだよ！銀座線が動いてても帰れないよ！

- 偽陰性

- － 日比谷線運転再開まで時間潰す
- － 有楽町線も全線死亡

3.2. 分類器の作成

- － メトロ**東西線**、都営新宿線はともに運転見合わせ中。ぎょえー。丸ノ内線、南北線は折り返し運転をやってる。有楽町線も和光市行きが動き始めた。このへんを乗り継げば何とか帰宅できるだろう。

誤分類されたツイートには大きく3つの特徴が見られた。

1つ目は偽陽性の最初の例に示すような、運行状況についての疑問文や、投稿者の想像、予測を含むツイートである。本実験では投稿に含まれる形態素のみを特徴量としている。そのため、投稿内容が疑問、想像、予測であるかどうかの判定ができず誤分類が生じたと考えられる。

2つ目は、偽陰性の1つ目、2つ目に示すような、婉曲的な表現を含むツイートである。偽陰性の1つ目の例では、「運転再開まで時間を潰す」という表現から、今現在運転を見合わせているということを推定しなければならない。この推定を今回の実験で用いた特徴量のみを用いて行うことは困難である。また、2つ目の例の「全線死亡」のようにTwitterで多く見られる崩れた日本語表現にも対応する必要もある。学習データを増やすことで、数多くの表現に対応することが解決策として挙げられる。

3つ目は、偽陽性の2つ目、3つ目および偽陰性の3つ目に示すような、複数の路線の運行状況について述べられているツイートである。本実験ではQUERYWORDの位置や、文脈を考慮するような特徴量を用いていないため、どの路線について述べているのかを明確に区別することは難しい。表3.4に示した通り、 T_1 を正例とした分類器は「見合わせ」という形態素の重みが大きい。したがって、偽陽性の2つ目の例では、銀座線は遅延しているが運転見合わせは行っていないにもかかわらず、後半に述べられている「運転見合わせ」によって全線見合わせと誤分類されたと考えられる。偽陽性の3つ目の例も同様である。偽陰性の3つ目の例は「有楽町線」という形態素の重みが小さいため、全線見合わせでないと誤分類されたと考えられる。

第4章 鉄道運行トラブルの検出

本章では，バースト検出手法を用いた運行トラブルの発生時間および復旧時間の検出を行う．対象とするトラブルは， T_1 : 全線運転見合わせ， T_2 : 一部区間運転見合わせ， $T_1 + T_2$: 見合わせの3種類である．

4.1 バースト検出手法

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 [15], [16], [17], [18], [19] が行われている．本論文では，トラブル関連ツイートの出現をイベントと定義し，ツイートの投稿時間の時系列に対してバースト検出手法を適用する．鉄道の運行トラブルをバースト検出によって早期に検出するには，リアルタイム性の高いバースト検出手法を適用する必要がある．そこで，本論文では蝦名ら [19] が提案したリアルタイムバースト検出手法を用いた．蝦名らの手法は Zhang ら [16] が提案した Aggregation Pyramid を用いた手法を，リアルタイムにバースト検知ができるように改良されたものである．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．以上のことから，蝦名ら

4.1. バースト検出手法

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となるイベントの検出に適していると言える．

4.1.1 データ構造

蝦名らの手法は図 4.1 のような Aggregation Pyramid と呼ばれるピラミッド状のデータ構造を持つ．最新のデータがセルに格納され，各レベルの右側に追加される．

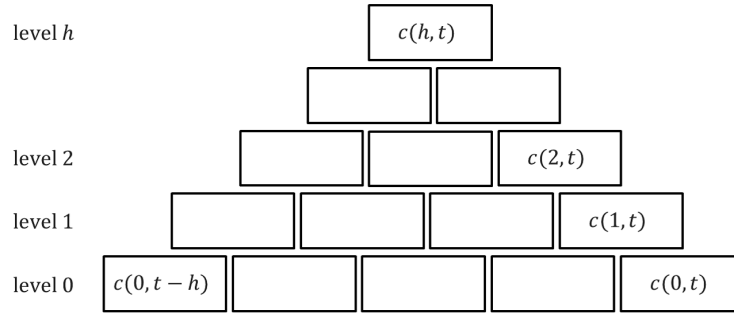


図 4.1: Aggregation Pyramid の構造例

その後，各セルの左側のセルが破棄される．ここで，ピラミッドは N 個のレベルから構成され，時間 t に終了するレベル h のセルを $c(h,t)$ と定義する．レベル 0 は N 個のセルをもち，これらは実際のデータに対応する．各セルは合計到着間隔 $gaps$ ，到着時間 $arrt$ ，間隔個数 $gapn$ の 3 つのデータをもつ．一連の $n+1$ 個のイベントに対して，各イベントの n 個の発生間隔を $\mathbf{x} = (x_1, x_2, \dots, x_n)$ と表す．また，大量のイベントが集中発生した際の計算量を削減するため，解析するウインドウサイズの最小値 W_{min} を定義する． W_{min} よりも短い期間内の情報を 1 つのセルに圧縮することによって，更新回数を一定以下に抑えることができる．図 4.2，4.3 に Aggregation Pyramid の構築方法を示す．

4.1. バースト検出手法

```
1 if  $x_i \geq W_{min}$ 
2    $c(0, t).gaps \leftarrow x_i$ 
3    $c(0, t).arrt \leftarrow i + 1$  番目のイベントが発生した時間
4    $c(0, t).gapn \leftarrow 1$ 
5    $i ++$ 
6    $t ++$ 
7 else
8    $c(0, t).gaps \leftarrow W_{min}$ 
9    $c(0, t).arrt \leftarrow c(0, t-1).arrt + W_{min}$ 
10   $c(0, t).gapn \leftarrow c(0, t-1).arrt$  から  $c(0, t).arrt$  直前までのイベン  
    ト数
11  if  $c(0, t).arrt$  でイベントが発生していない
12     $c(0, t).gapn --$ 
13   $i \leftarrow i + c(0, t).gapn$ 
14   $t ++$ 
15   $x_i \leftarrow c(0, t).arrt$  から次のイベント発生までの経過時間
16  if  $c(0, t).arrt$  で複数イベントが発生
17     $x_i \leftarrow 0$ 
```

図 4.2: レベル 0 のセルの生成方法

```
1  $c(h, t).gaps \leftarrow c(h-1, t-1).gaps + c(0, t).gaps$ 
2  $c(h, t).arrt \leftarrow c(0, t).arrt$ 
3  $c(h, t).gapn \leftarrow c(h-1, t-1).gapn + c(0, t).gapn$ 
```

図 4.3: レベル 1 以上のセルの生成方法

4.1.2 バースト判定方法

到着間隔が重複していない直前の状態よりも急激に小さくなっている期間をバーストが発生していると定義する．ここで，平均到着間隔を式 (4.1) によって定義する．

$$arg(c(h, t)) = \frac{c(h, t).gaps}{c(h, t).gapn} \quad (4.1)$$

4.2. パラメータの調整

```

1 for  $t$  do
2    $h \leftarrow 0$ 
3   while
4      $c(h, t)$  を生成
5      $c(h - 1, t - 1)$  を破棄
6     if  $c(h, t).gapn \geq A_{min}$  かつ  $c(h, t)$  が式 (4.2) を満たす
7        $c(h, t)$  がバーストしていると判定
8     if  $h == N - 1$ 
9        $c(N - 1, t - 1 - h)$  を破棄
10     $c(h, t)$  をデータ構造に追加
11     $h ++$ 
```

図 4.4: バースト検出アルゴリズム

不等式 (4.2) を満たすとき、バーストが発生していると定義する。

$$arg(c(h, t)) \leq \beta \times arg(c(N - 1, t - 1 - h)) \quad (4.2)$$

ただし、 $\beta (0 < \beta < 1)$ はバーストの判定基準を調整するパラメータである。また、バーストと判定するイベント発生間隔の最低個数 A_{min} を設定し、 $c(h, t).gapn < A_{min}$ となる場合は、バースト判定を行わない。図 4.4 にバースト検出アルゴリズムを示す。本論文では最初にバースト判定されたセルの到着時間をトラブルの検出時間とする。

4.2 パラメータの調整

蝦名らの手法では、予め4種類のパラメータを与える必要がある。パラメータはそれぞれ、Aggregation Pyramid の高さ N 、バースト検出の判定基準 β 、Aggregation Pyramid の1つのセルにイベントを圧縮するウィンドウサイズ W_{min} 、バーストと判定するイベント発生間隔の最低個数 A_{min} である。評価実験では W_{min} は1に固定した。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

4.3. 評価実験

の組 θ を与えたときの、バースト検出の結果の評価値を式 (4.3) によって定義する.

$$score_n(\theta) = |T_{actual}(n) - T_n(\theta)| \quad (4.3)$$

ただし, $T_{actual}(n)$ は実際の運行トラブルの発生時間, $T_n(\theta)$ はパラメータ θ が与えられたときのバースト検出手法による検出時間である. ここで, 時間は全て秒単位である. トラブルが全部で K ケースあるとき, あるケース n のバースト検出には, 式 (4.4) を最小化するようなパラメータを用いる.

$$\sum_{i=1, i \neq n}^K score_n(\theta) \quad (4.4)$$

パラメータの探索には N と A_{min} をそれぞれ 1 から 10 まで 1 刻み, β を 0.01 から 1.00 まで 0.01 刻みで変化させるグリッドサーチによって行った.

4.3 評価実験

4.3.1 実験設定

評価実験として運行トラブルの発生時間および復旧時間の検出を行う. 対象とする運行トラブルの種類は, T_1 : 全線運転見合わせ, T_2 : 一部区間運転見合わせ, $T_1 + T_2$: 見合わせの 3 種類である. 対象とするトラブルは 2011 年 6 月から 2013 年 10 月までに東京メトロ 9 路線のいずれかにおいて発生したトラブルのうち, goo 路線運行情報¹ から発生時間と復旧時間を分単位で取得できたトラブルのみとした. 以下に, goo 路線運行情報に掲載された文章の例を示す.

- T_1 : 全線運転見合わせ
 - ー 東京メトロ千代田
21:40 頃、北千住駅で発生した人身事故の影響で、現在も運転を見合わせています。なお、振替輸送を行っています。

¹<http://transit.goo.ne.jp/unkou/>

4.3. 評価実験

- T_2 : 一部区間運転見合わせ
 - － 東京メトロ千代田線
13:17 頃、乃木坂駅で発生した人身事故の影響で、現在も表参道～霞ヶ関間の運転を見合わせています。
- R_1 : 復旧
 - － 東京メトロ千代田線
乃木坂駅で発生した人身事故の影響で、表参道～霞ヶ関間の運転を見合わせていましたが、14:46 頃、運転を再開しました。なお、列車に遅れが出ています。

バースト検出手法は、3.2 節の分類器によって抽出されたトラブル関連ツイートの投稿時間の時系列に対して適用する。トラブル関連ツイートの抽出には、トラブル発生検出においては、対象とするトラブルに応じた分類器を用いる。トラブル復旧検出においては、復旧関連ツイートを抽出する分類器を用いる。ここで、分類器によって抽出されたトラブル関連ツイート数が 10 に満たないトラブルは実験の対象から除外した。表 4.1 に対象とする運行トラブルの数を示す。

	T_1	T_2	$T_1 + T_2$
Trouble	52	40	89
Recovery	36	26	54

表 4.1: バースト検出を行うトラブル数

蝦名らの手法はバースト検出を行う前に Aggregation Pyramid のデータ構造内に一定のデータを蓄積する必要がある。バースト検出対象の日に投稿されたツイートを用いて Aggregation Pyramid を構築すると、その日の早い時間帯で実際のバーストがあった際に、その時間帯のツイートが Aggregation Pyramid の構築に用いられるため、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

4.3. 評価実験

の平均到着間隔で予め Aggregation Pyramid を構築する．トラブル関連ツイートの平均到着間隔は式 (4.5) によって求める．

$$\frac{20 \times 3600}{AVG(m) \times FPR} \quad (4.5)$$

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決定した．

4.3.2 実験結果と考察

表 4.2, 4.3 に，各種トラブルのそれぞれのケースに対してパラメータ調整をした上でバースト検出を行ったときの，式 (4.3) の中央値を示す．

	T_1	T_2	$T_1 + T_2$
Burst Detection	286.0	1404.5	254.0
Threshold	420.0	1260.0	420.0

表 4.2: トラブル発生検出の評価値の中央値

	T_1	T_2	$T_1 + T_2$
Burst Detection	313.0	1903.0	452.0
Threshold	241.0	3539.0	301.0

表 4.3: トラブル復旧検出の評価値の中央値

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によって中央値で 5 分以内の誤差でトラブルの検出ができており，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

4.3. 評価実験

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。

各ケースの評価値の分布を図4.5, 4.6に示す。トラブルの発生検出では、バースト検出手法による T_1 の検出では5分以内に全体の5割強、 $T_1 + T_2$ の検出においては5分以内に全体の6割弱のトラブルを検出していることが確認できた。 T_2 においては、バースト検出によって5分以内に検出できたのは全体の数パーセント程度であり、実用的でないことが分かった。トラブルの復旧検出においては、 T_1 および $T_1 + T_2$ の検出において、5分以内に検出できたケース数はバースト検出手法、閾値による検出手法いずれもほぼ同程度であることが確認できた。 T_2 の検出においては、5分以内に検出できたケースはバースト検出手法が全体の1割程度、閾値による検出手法が全体の2割程度であり、発生検出に比べれば高いことが分かった。

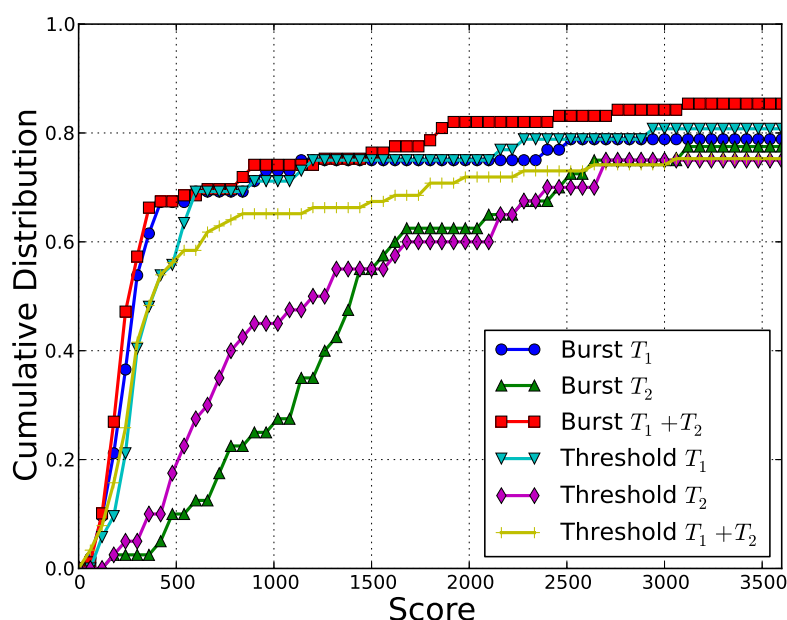


図 4.5: トラブル発生検出における評価値の分布

4.3. 評価実験

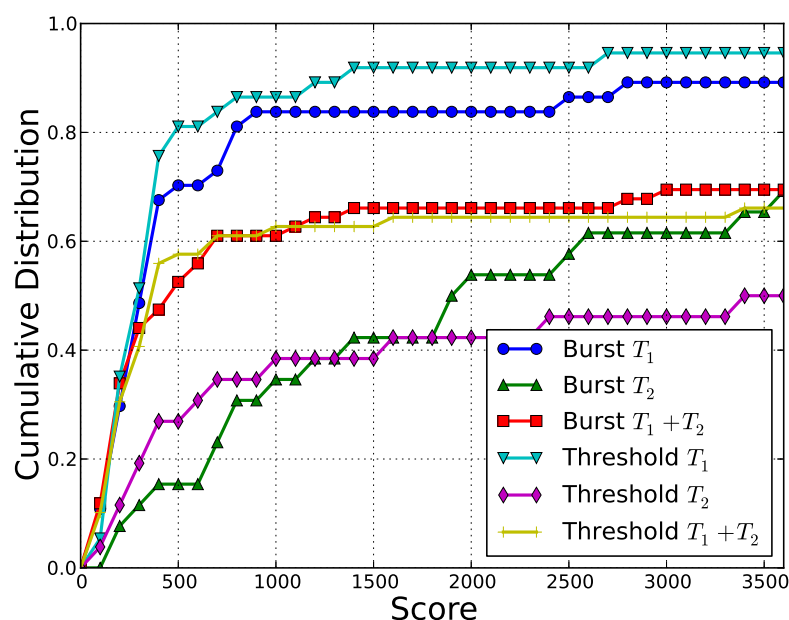


図 4.6: トラブル復旧検出における評価値の分布

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 4.7 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、誤差が非常に大きい。

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 4.8 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

4.3. 評価実験

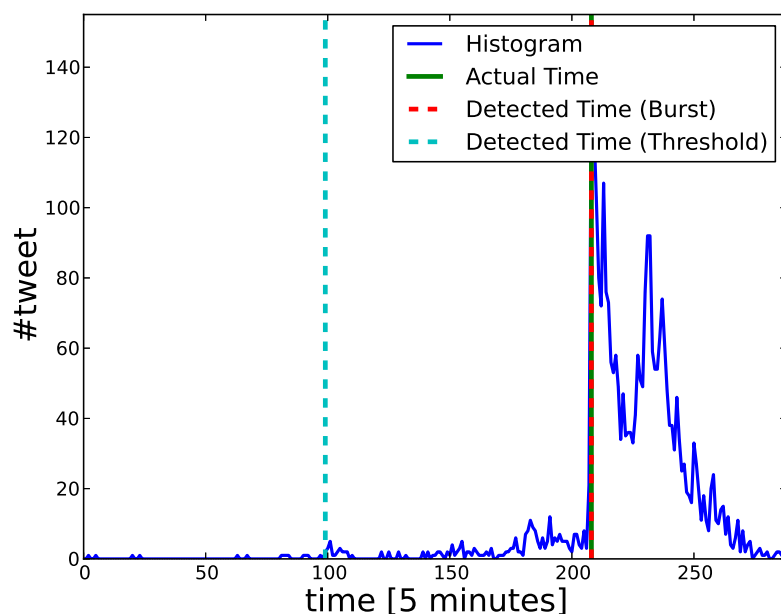


図 4.7: 2012 年 4 月 3 日東西線の運転見合わせのヒストグラムおよび検出時間

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るものである。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と考えられる。この日は台風によって東西線のダイヤが大きく乱れたため、トラブル関連ツイートが他のケースに比べ多く投稿された。このようなケースではバーストが複数見られ、検出が困難になるという傾向が確認できた。

最後に、一部見合わせの検出が有効に機能しなかった原因について考察を行う。一部見合わせの発生検出において、評価値が 30 分を超えたトラブル 14 件について解析したところ、5 つの原因が見られた。1 つ目は、実際には全線で運転を見合わせたケースである。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

4.3. 評価実験

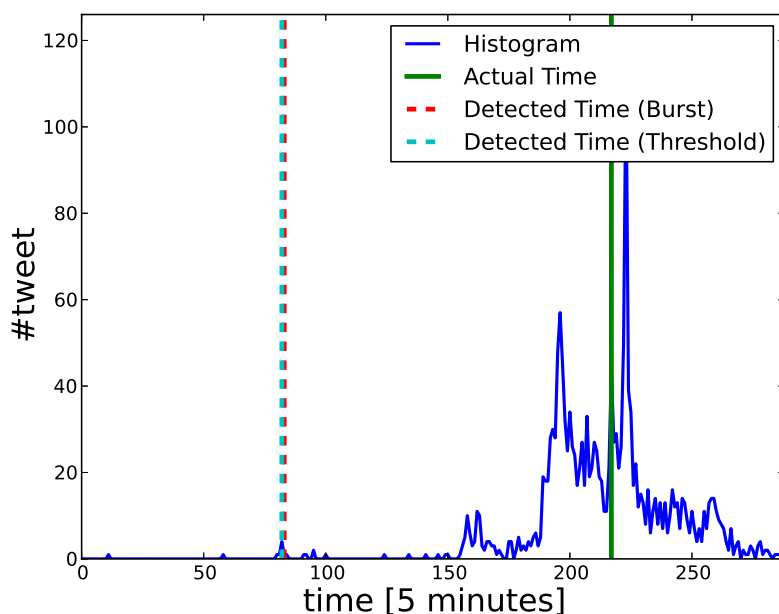


図 4.8: 2011 年 9 月 21 日東西線の全線見合わせのヒストグラムおよび検出時間

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．2つ目は，誤分類が原因である一スで，3件確認された．分類器が誤ってトラブル関連ツイートと判定したツイートが集中することによって生じたバーストが検出されることで，評価値が悪くなった．3つ目は情報の反映が遅いケースである．早朝5時台に発生したトラブルや，本線ではなく支線で発生したトラブルなど，トラブルに巻き込まれた人が少ない場合はTwitterへの反映が遅れることが確認できた．このようなケースは3件あった．4つ目は，情報反映が早いケースである．これは正解データが誤っていることが原因であると考えられる．実際のツイートではトラブルが発生したというツイートが多く投稿されているが，正解データではこのような投稿がなされた時間よりも後の時間をトラブル開始時間としていた．このようなケースは2件確認された．最後は情報が錯そうしているケースである．全線見合わせおよび一部見合わせについてのツ

4.3. 評価実験

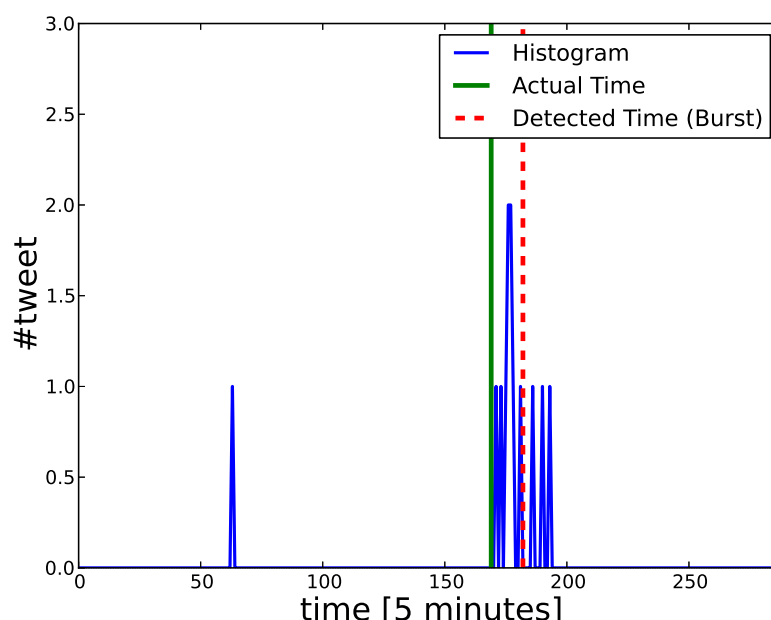


図 4.9: 2011 年 7 月 6 日半蔵門線の一部見合わせのヒストグラムおよび検出時間

イトが混在しており，このようなケースにおいては検出が困難であることが確認できた．このことが原因で検出誤差が大きかったケースは 1 件見られた．また，以上で述べた原因に加えて，一部見合わせのに関するツイートは他のトラブル関連ツイートに比べ，数が少ない傾向にあることも一部見合わせの検出性能を下げていると考えられる．図 4.9 に 2011 年 7 月 6 日に半蔵門線で発生した一部見合わせ関連ツイートのヒストグラムと検出時間を示す．この日投稿された一部見合わせ関連のツイートは 13 件のみであった．一方で，見合わせ関連のツイートは 171 件であることから，一部見合わせ関連のツイートは著しく少ないことが分かる．ツイート数が少ないことが原因で閾値による検出手法によってトラブル発生と検出された時間はなかった．また，正解時間付近でツイート数が 1 であり，バースト検出手法によって正しく検出することは困難であると考えられる．

第5章 鉄道運行トラブルの継続時間予測

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， $T_1 + T_2$: 見合わせを対象にした実験によって評価を行う．提案する手法はトラブル発生後初期段階において予測する手法と，トラブル発生後数分間隔で逐次予測を行う手法の2つである．

5.1 初期段階における継続時間予測

5.1.1 提案タスクと予測手法

本論文では運行トラブル発生後，初期段階で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分間に投稿されたトラブル関連ツイートを取得する．次に，5分間に投稿され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．トラブルの継続時間にはトラブル全体の継続時間から5分を引いた時間を用いた．分類器には線形カーネルのSVMを用いた．

5.1.2 評価実験

5.1.2.1 実験設定

対象とする運行トラブルの種類は $T_1 + T_2$: 見合わせである．対象とするトラブルは2011年6月から2013年10月までに東京メトロ9路線のいずれかにおいて発生したトラブルのうち，goo 路線運行情報からトラブルの発生時間および復旧時間の両方が分単位で取得できたトラブルのみとした．goo 路線運行情報に掲載された情報の例は4.3.1項を参照されたい．取得した運行トラブルは全部で60件である．これらのトラブルの継続時間についての統計を図5.1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．

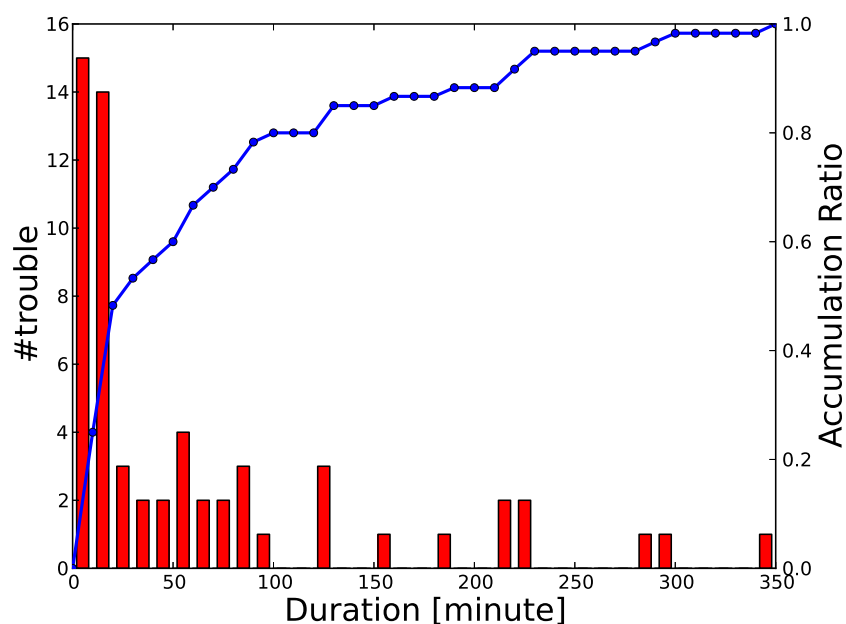


図 5.1: 継続時間の分布

5.1. 初期段階における継続時間予測

5.1.2.2 実験結果と考察

表 5.1, 5.2, 5.3, 5.4 に実験結果を示す. $t = 15$ 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, 各クラスの F 値および精度においては, 用いる特徴量として WA を用いた方が WC よりも良い性能を示す傾向があることが分かった.

Method	15 分未満			15 分以上			Accuracy
	Precision	Recall	F-value	Precision	Recall	F-value	
WA	0.629	0.710	0.667	0.640	0.552	0.593	0.633
WC	0.595	0.710	0.647	0.609	0.483	0.538	0.600
Baseline	0.650	1.000	0.788	-	-	-	0.650

表 5.1: $t = 15$ 分における予測結果

Method	30 分未満			30 分以上			Accuracy
	Precision	Recall	F-value	Precision	Recall	F-value	
WA	0.636	0.519	0.571	0.658	0.758	0.704	0.650
WC	0.706	0.444	0.545	0.651	0.848	0.737	0.667
Baseline	-	-	-	0.533	1.000	0.696	0.533

表 5.2: $t = 30$ 分における予測結果

Method	45 分未満			45 分以上			Accuracy
	Precision	Recall	F-value	Precision	Recall	F-value	
WA	0.706	0.500	0.585	0.721	0.861	0.785	0.717
WC	0.706	0.500	0.585	0.721	0.861	0.785	0.717
Baseline	-	-	-	0.583	1.000	0.737	0.583

表 5.3: $t = 45$ 分における予測結果

次に, 表 5.5 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. その結果, 予測精度が良かったと考えられる.

5.1. 初期段階における継続時間予測

Method	60 分未満			60 分以上			Accuracy
	Precision	Recall	F-value	Precision	Recall	F-value	
WA	0.833	0.526	0.645	0.813	0.951	0.876	0.817
WC	0.714	0.526	0.606	0.804	0.902	0.851	0.783
Baseline	-	-	-	0.650	1.000	0.788	0.650

表 5.4: $t = 60$ 分における予測結果

30 分未満			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 5.5: $t = 30$ 分における WA を用いた SVM の特徴量の重みの例

一方で、提案手法が有効に機能しなかった 15 分以上長引くかどうかを判定する、特徴量に WA を用いた分類器の重みを表 5.6 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 値を見てみると、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 であり、やはり 15 分以上のトラブル分類時の性能が良くないことが確認できる。したがって、提案手法が有効に機能しなかったと考えられる。

15 分未満			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	車輻	-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 5.6: $t = 15$ 分における WA を用いた SVM の特徴量の重みの例

5.2 継続時間の逐次予測

5.2.1 提案タスクと予測手法

本項では，トラブル発生後，トラブルの継続時間を5分おきに逐次予測する手法について提案する．提案手法では，運行トラブルが終了するまでの分単位の残り時間を従属変数とした回帰モデルによって，継続時間の予測を行う．運行トラブルが発生してから5分ごとに，その5分間に投稿されたトラブル関連ツイートすべてを1つのドキュメントとみなし，特徴量を算出する．本論文では，特徴量の算出方法が異なる4つの手法を試みる．ここで， \mathbf{x}_i をトラブルが発生してから*i*番目のドキュメントに出現した形態素の出現頻度を表すベクトルとする．

1つ目の手法 (以下，Single と呼ぶ) は，直近の情報のみを用いる手法である．*n* 番目のドキュメントの特徴量は式 (5.1) によって求める．

$$\mathbf{x}_n \quad (5.1)$$

2つ目の手法 (以下，Addition と呼ぶ) は運行トラブル発生以降，各ドキュメントの特徴量を全て足していく手法である．*n* 番目の時間帯の特徴量は式 (5.2) によって求める．

$$\sum_{i=1}^n \mathbf{x}_i \quad (5.2)$$

3つ目の手法 (以下，Window と呼ぶ) はウインドウサイズ *W* を設定し，*W* だけ過去のドキュメントの特徴量を加える手法である．*n* 番目の特徴量は式 (5.3) によって求められる．

$$\begin{cases} \frac{W}{n} \sum_{i=1}^n \mathbf{x}_i & (1 \leq n \leq W) \\ \sum_{i=n-W+1}^n \mathbf{x}_i & (n > W) \end{cases} \quad (5.3)$$

4つ目の手法 (以下，Exponent と呼ぶ) は過去のドキュメントの特徴量を全て用いるが，Addition とは違い，直近のドキュメントの特徴量ほど重視するような手法で

5.2. 継続時間の逐次予測

ある． n 番目の時間帯の特徴量は式 (5.4) によって求められる．ここで， θ は過去のドキュメントの特徴量をどの程度織り込むかを調整するパラメータである．

$$\sum_{i=1}^n \mathbf{x}_i e^{-\theta(n-i)} \quad (5.4)$$

5.2.2 評価実験

5.2.2.1 実験設定

対象とする運行トラブルの種類は， $T_1 + T_2$ ：見合わせである．使用したデータは 5.1.2 項と同じである．回帰モデルの学習は，予測を行うトラブル以外の全てのトラブルで行った．回帰モデルには L2-regularized L2-loss SVR を用いた．また，SVR のライブラリには LIBLINEAR¹ を用いた．Window のウインドウサイズ W および，Exponent のパラメータ θ は，実験的にそれぞれ $W = 3$ ， $\theta = 0.2$ と設定した．提案手法の評価は，回帰モデルによって得られたトラブル収束までの残り時間が t 分以上かどうかで，そのトラブルが以後 t 分以上継続するかどうかを予測する実験で行う． t は 15 分，30 分，45 分，60 分の 4 通りを試みた．また，ベースラインには実際の残り時間が t 分以上であるケースが半数以上の場合には全て t 分以上，そうでない場合には全て t 分未満と判定する手法を用いた．

5.2.2.2 実験結果と考察

実際の継続時間と予測した継続時間の相関係数を表 5.7 に示す．過去のドキュメントの特徴量を，直近のドキュメントの特徴量と同等に扱う Addition の相関が最も低くなった．一方で，過去の情報をある程度限定して用いる Window および Exponent は，過去の情報を一切使用しない Single に比べ，相関係数が高くなった．このことから，直近のドキュメントの情報に加えて，過去のドキュメントをある程度限定して考慮することの有効性が確認できた．

¹<http://www.csie.ntu.edu.tw/~cjlin/liblinear/>

5.2. 継続時間の逐次予測

	Single	Addition	Window	Exponent
Correlation Coefficient	0.273	0.163	0.361	0.410

表 5.7: 回帰モデルの相関係数

表 5.8 に実験で得られた運行トラブル継続時間予測の精度を示す。継続時間が 45 分よりも長いケースを判定する場合、提案した全ての回帰モデルにおいて、ベースラインよりも精度が高くなった。特に Window と Exponent は全ての時間において精度が 7 割以上であり、提案手法の有効性が確認できた。相関係数は Window よりも Exponent の方が高い結果であったが、継続時間の判定においては、全ての場合において Window の精度が高い結果となった。また、Window および Exponent の精度は、全ての場合において Single の精度よりも高くなり、過去の情報をある程度限定して織り込むことの有効性が確認できた。

	Baseline	Single	Addition	Window	Exponent
$t = 15$	0.816	0.734	0.590	0.770	0.740
$t = 30$	0.699	0.699	0.587	0.736	0.708
$t = 45$	0.600	0.676	0.623	0.711	0.706
$t = 60$	0.510	0.696	0.642	0.723	0.711

表 5.8: 回帰モデルによる継続時間の予測精度

次に回帰モデルによる予測結果と実際の継続時間の誤差が小さい場合についての考察を行う。2011 年 9 月 21 日の東西線の運転見合わせの継続時間を Single によって予測した際、実際の継続時間が 102 分であるのに対し、予測結果は 105.4 分であり、ある程度正しく予測ができていた。このときにドキュメントに含まれていたツイートの例を以下に示す。

- 誤差が小さい場合のドキュメントに含まれたツイートの例
 - － よけいにひどくなっている… #kotoku 東西線 【運転見合わせ】18 時 04 分頃、台風接近に伴う混雑のため、全線で運転を見合わせています。
運行情報 | 東京メトロ URL

5.2. 継続時間の逐次予測

- － 東西線、地上区間を徒歩で点検中、地下区間も地上の点検が終わらないと再開できないとのこと。**一時間単位で掛かる**だろうね。

これらのツイートには、太字で示したようなトラブルが長引くことを示唆する内容が含まれていることが確認できた。このように、トラブルの継続時間を予測出来得る形態素が含まれる場合は、ある程度正しい予測が可能になると考えられる。

次に、予測結果と実際の継続時間の誤差が大きい場合についての考察を行う。2011年9月21日の千代田線の運転見合わせの継続時間は136分であるのに対し、予測結果は5.7分であった。このときにドキュメントに含まれていたツイートの例を以下に示す。

- 予測結果が実際よりも短い場合のドキュメントに含まれたツイートの例

- － **【台風】** 千代田線は霞が関～北千住の折り返し運転再開。

このツイートには一部区間での運転再開に関する情報が含まれており、その結果、実際の継続時間よりもはるかに短い予測を行ったと考えられる。また、2013年5月9日の千代田線においては、この例とは逆に実際の継続時間が1分であるのに対し、予測結果は273.8分と非常に長かった。これは、見合わせを行った時間帯が通勤時間帯ということもあり、非常に多くのトラブル関連ツイートが投稿されたことが原因であると考えられる。提案手法では形態素の出現回数を特徴量としているため、ツイート数の情報も間接的に織り込まれる。大規模で復旧までに時間が掛かるトラブルが発生した際は、トラブル関連ツイートが多く投稿されるため、ツイート数が多い場合は予測時間が長くなる傾向にあった。

第6章 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 $T_1 + T_2 + T_3$: 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トラブルとみなしたもの）を対象にした評価実験により，提案手法の有効性を示す．

6.1 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．なお，goo 路線運行情報は1時間に1回クローリングしている．

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．goo 路線運行情報からは〈掲載時刻，トラブルが発生した路線，トラブルの詳細〉を取得することができる．トラブルの詳細は自然言語で記述されており，詳細の中にトラブルの発生時間や，連鎖の有無に関する情報が含まれる場合がある．以下に，トラブルの連鎖について述べられている例を示す．

- 連鎖元のトラブル発生時間が明確な場合

- － 小田急江ノ島線

- 09:05 頃、小田急小田原線内で発生した信号関係故障の影響で、一部列車に遅れが出ています。

- 連鎖元のトラブル発生時間が不明な場合

6.1. 実験データの作成

－ 湘南新宿ライン

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。

トラブルの詳細に対する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 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。

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。

● トラブル発生時間が明確で、連鎖に関する記述がない場合

－ 京浜東北根岸線

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

6.2. 鉄道運行トラブルの連鎖予測手法

含まれない路線は除外した．表 6.1 に作成した正解データの統計を示す．また，図 6.1 に連鎖ありの正解データから作成した，連鎖元から連鎖先の路線を繋いだ有向グラフの一部を示す．

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 6.1: 連鎖予測で用いるデータの統計

6.2 鉄道運行トラブルの連鎖予測手法

本論文では，路線 x_i で発生したトラブルが路線 y_j への連鎖するかどうかの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 t 分間の間に投稿された運行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) およびトラブルの発生時間 (以下 T と呼ぶ) を用い，これらの組み合わせによって WA, WA+T, WC, WC+T の4通りの実験を行う．トラブルの発生時間を特徴量に用いる理由は，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たためである．次に，連鎖先の路線ごとに分類器を作成する．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，学習時に 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，学習データにおい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．学習データにおいて発生した連鎖の連鎖元から連鎖先の路線への有向グラフは図 6.2 のようであるとする．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 (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) に用いるデータは x_1 および x_2 で発生

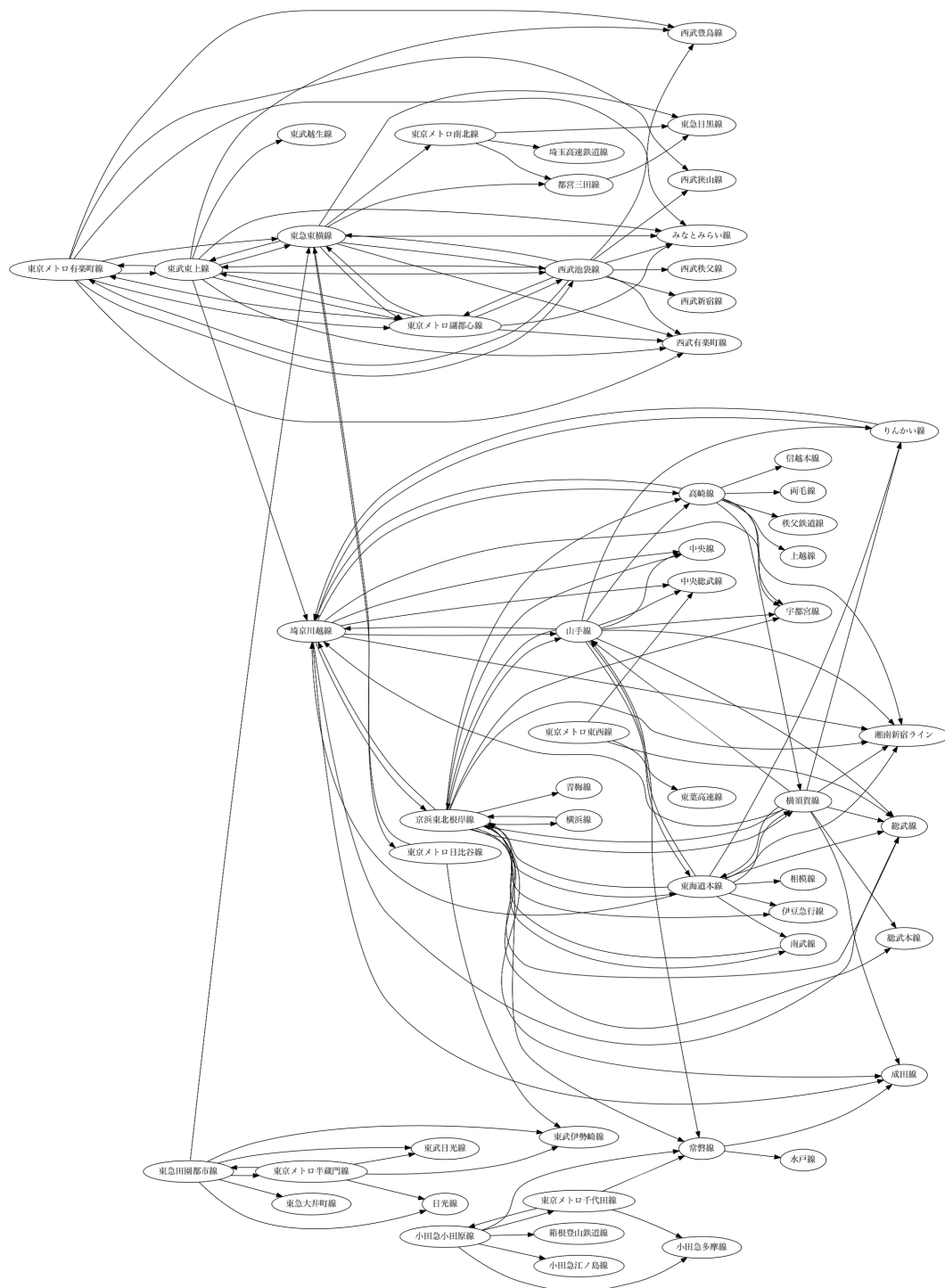


図 6.1: 路線間の連鎖関係を表す有向グラフ

6.2. 鉄道運行トラブルの連鎖予測手法

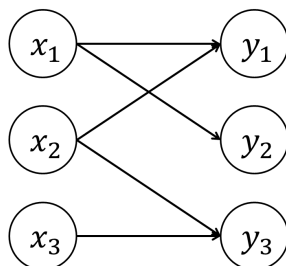


図 6.2: 連鎖関係を表す有向グラフの例

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．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 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、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．この手法では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．そこで、学習時に発生しなかったパターンでも予測可能な手法を2つ目の手法として提案する．

2つ目の手法は、連鎖先の路線 y_j の分類器の学習に、学習時に発生したトラブル全てを用いる手法である．その際、学習時の連鎖率を特徴量として加える．1つ目の手法の例と同様の状況を考え、連鎖率を矢印の上に記載したものを図 6.3 に示す．学習時に連鎖が発生しないようなパターンの連鎖率は0とする．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．予測を行う際は、学習時の連鎖率をテストデータに加えた上で、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．この手法では、学習時に発生しなかった連鎖のパターンを予測する

6.3. 評価実験

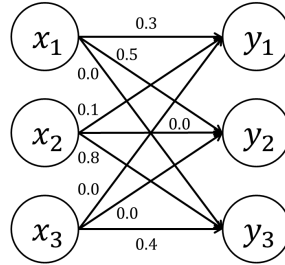


図 6.3: 連鎖関係を表す有向グラフの例 (連鎖率あり)

ことが可能である．例えば，テストデータには $x_1 \rightarrow y_3$ という連鎖パターンは含まれていないが，テストデータにこのパターンが含まれている場合は，テストデータの連鎖率の特徴量を 0 にした上で， y_3 の分類器の出力結果によって予測を行うことができる．

6.3 評価実験

6.3.1 実験設定

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．表 6.3.1 に連鎖率を用いない手法，用いる手法それぞれについて，学習データとテストデータの路線ごとのトラブル数を示す．

連鎖先の路線	連鎖率を用いない手法				連鎖率を用いる手法			
	学習		テスト		学習		テスト	
	正例	負例	正例	負例	正例	負例	正例	負例
みなとみらい線	27	2	16	1	27	759	16	343
りんかい線	31	100	14	50	31	755	14	345
伊豆急行線	8	112	4	38	8	778	4	355
宇都宮線	22	202	7	80	22	764	7	352
横須賀線	10	165	6	57	10	776	6	353
横浜線	2	70	1	18	2	784	1	358
京王高尾線	34	9	27	3	34	752	27	332

6.3. 評価実験

連鎖先の路線	連鎖率を用いない手法				連鎖率を用いる手法			
	学習		テスト		学習		テスト	
	正例	負例	正例	負例	正例	負例	正例	負例
京王新線	36	15	23	9	36	750	23	336
京王線	6	2	1	1	6	780	1	358
京王相模原線	46	5	28	4	46	740	28	331
京急久里浜線	5	7	0	1	5	781	0	359
京急空港線	10	2	1	0	10	776	1	358
京急逗子線	3	9	0	1	3	783	0	359
京急本線	11	1	1	0	11	775	1	358
京成押上線	10	2	1	0	10	776	1	358
京成本線	9	3	1	0	9	777	1	358
京浜東北根岸線	38	157	15	81	38	748	15	344
京葉線	14	9	6	6	14	772	6	353
高崎線	18	151	6	60	18	768	6	353
埼京川越線	25	241	5	105	25	761	5	354
埼玉高速鉄道線	0	0	2	0	0	786	2	357
山手線	23	176	7	72	23	763	7	352
小田急江ノ島線	59	26	19	15	59	727	19	340
小田急小田原線	5	12	0	4	5	781	0	359
小田急多摩線	43	59	12	26	43	743	12	347
湘南新宿ライン	92	214	35	91	92	694	35	324
上越線	1	54	1	20	1	785	1	358
常磐線	15	273	5	108	15	771	5	354
信越本線	0	0	1	0	0	786	1	358
水戸線	4	58	2	28	4	782	2	357
成田スカイアクセス	6	6	1	0	6	780	1	358
成田線	5	91	3	45	5	781	3	356
西武狭山線	6	47	2	26	6	780	2	357
西武新宿線	0	0	1	0	0	786	1	358
西武池袋線	16	93	11	49	16	770	11	348
西武秩父線	3	37	0	24	3	783	0	359
西武豊島線	36	17	15	14	36	750	15	344
西武有楽町線	45	64	24	36	45	741	24	335
青梅線	1	71	0	19	1	785	0	359
相模線	1	47	0	23	1	785	0	359
総武線	33	192	23	77	33	753	23	336
総武本線	2	32	2	15	2	784	2	357
秩父鉄道線	0	0	2	0	0	786	2	357
中央線	6	163	0	66	6	780	0	359
中央総武線	10	61	8	34	10	776	8	351

6.3. 評価実験

連鎖先の路線	連鎖率を用いない手法				連鎖率を用いる手法			
	学習		テスト		学習		テスト	
	正例	負例	正例	負例	正例	負例	正例	負例
都営三田線	4	28	1	10	4	782	1	358
都営新宿線	24	19	16	14	24	762	16	343
東海道本線	35	123	10	51	35	751	10	349
東急大井町線	3	26	1	9	3	783	1	358
東急田園都市線	1	1	2	0	1	785	2	357
東急東横線	4	31	10	14	4	782	10	349
東急目黒線	4	29	1	11	4	782	1	358
東京メトロ千代田線	32	53	8	26	32	754	8	351
東京メトロ南北線	2	27	1	7	2	784	1	358
東京メトロ日比谷線	19	10	3	5	19	767	3	356
東京メトロ半蔵門線	25	4	10	0	25	761	10	349
東京メトロ副都心線	50	53	19	37	50	736	19	340
東京メトロ有楽町線	50	46	18	35	50	736	18	341
東武伊勢崎線	7	30	5	11	7	779	5	354
東武越生線	0	0	1	0	0	786	1	358
東武東上線	11	48	4	29	11	775	4	355
東武日光線	4	27	3	9	4	782	3	356
東葉高速線	14	12	10	9	14	772	10	349
南武線	1	47	2	22	1	785	2	357
日光線	4	27	3	9	4	782	3	356
箱根登山鉄道線	4	81	0	34	4	782	0	359
武蔵野線	4	2	4	0	4	782	4	355
北総線	8	4	1	0	8	778	1	358
両毛線	0	0	1	0	0	786	1	358

表 6.2: 連鎖先の路線ごとのトラブル数

評価はテストデータ全てに対してと、連鎖元の路線ごとに行う。対象とするトラブルは見合わせ、遅延、直通運転中止などを全て同一とみなしたトラブルである。トラブル関連ツイートの抽出には3.2節で作成した、 $T_1 + T_2 + T_3$ をターゲットラベルとした分類器を用いた。トラブル発生後のツイートの解析時間は5分とした。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。連鎖率を用いない手法では、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

6.3. 評価実験

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 として扱う。

6.3.2 実験結果と考察

テストデータ全てに対する実験結果を表 6.3, 6.4 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、連鎖率を用いない手法では、特徴量にトラブルの発生時間も加えた方が分類器の性能が向上することが確認できた。さらに、連鎖率を用いない手法は、未知の連鎖パターンの予測を行えないため、連鎖率を用いる手法に比べて、連鎖ありの再現率は低い。適合率と F 値は高いことが確認できた。

Method	連鎖あり			連鎖なし			Accuracy
	Precision	Recall	F-value	Precision	Recall	F-value	
WA	0.733	0.469	0.572	0.757	0.907	0.825	0.752
WA+T	0.736	0.496	0.593	0.766	0.903	0.829	0.759
WC	0.749	0.479	0.584	0.762	0.912	0.830	0.759
WC+T	0.738	0.516	0.608	0.773	0.900	0.832	0.764

表 6.3: 連鎖率を用いない手法のテストデータ全体の予測結果

Method	連鎖あり			連鎖なし			Accuracy
	Precision	Recall	F-value	Precision	Recall	F-value	
WA	0.587	0.456	0.513	0.983	0.990	0.986	0.973
WA+T	0.539	0.516	0.528	0.985	0.986	0.985	0.971
WC	0.629	0.544	0.583	0.986	0.990	0.988	0.976
WC+T	0.607	0.554	0.579	0.986	0.989	0.987	0.975

表 6.4: 連鎖率を用いる手法のテストデータ全体の予測結果

次に、連鎖元の路線ごとに予測を行った結果を表 6.5 および 6.6 に示す。いずれの手法も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

6.3. 評価実験

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、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結果は悪くなる傾向にあった。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。

6.3. 評価実験

連鎖元の路線	特徴量	連鎖あり			連鎖なし			精度
		適合率	再現率	F 値	適合率	再現率	F 値	
常磐線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
京浜東北線	WA+T	1.000	0.118	0.211	0.545	1.000	0.706	0.571
京葉線	WA	N/A	N/A	N/A	N/A	N/A	N/A	N/A
武蔵野線	WA	0.545	1.000	0.706	1.000	0.167	0.286	0.583
南武線	WA	N/A	0.000	N/A	0.667	1.000	0.800	0.667
埼京線	WC	0.857	0.429	0.571	0.918	0.989	0.952	0.913
高崎線	WA+T	0.727	0.533	0.615	0.781	0.893	0.833	0.767
東海道線	WA+T	0.643	0.310	0.419	0.789	0.938	0.857	0.771
山手線	WC	0.500	0.273	0.353	0.902	0.961	0.931	0.875
横浜線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
横須賀線	WA	1.000	0.263	0.417	0.632	1.000	0.774	0.674
京王線	WA	0.958	1.000	0.979	N/A	0.000	N/A	0.958
東京メトロ千代田線	WC	0.500	0.500	0.500	0.833	0.833	0.833	0.750
東京メトロ副都心線	WC+T	0.750	0.750	0.750	0.750	0.750	0.750	0.750
東京メトロ半蔵門線	WA	N/A	N/A	N/A	N/A	N/A	N/A	N/A
東京メトロ日比谷線	WA	N/A	0.000	N/A	0.500	1.000	0.667	0.500
東京メトロ南北線	WA	N/A	N/A	N/A	N/A	N/A	N/A	N/A
東京メトロ東西線	WC+T	1.000	0.682	0.811	0.750	1.000	0.857	0.837
東京メトロ有楽町線	WC+T	1.000	0.545	0.706	0.000	N/A	N/A	0.545
小田急線	WC+T	0.667	0.821	0.736	0.863	0.733	0.793	0.768
西武池袋線	WC+T	0.732	0.714	0.723	0.789	0.804	0.796	0.765
東武東上線	WA+T	0.480	0.462	0.471	0.846	0.856	0.851	0.767
都営浅草線	WC	1.000	1.000	1.000	N/A	N/A	N/A	1.000
都営三田線	WA	N/A	N/A	N/A	N/A	N/A	N/A	N/A
都営新宿線	WA	0.500	1.000	0.667	N/A	0.000	N/A	0.500
東急田園都市線	WA+T	1.000	0.667	0.800	0.000	N/A	N/A	0.667
東急東横線	WA	1.000	0.231	0.375	0.000	N/A	N/A	0.231

表 6.5: 連鎖率を用いない手法の路線ごとの予測結果

6.3. 評価実験

連鎖元の路線	特徴量	連鎖あり			連鎖なし			精度
		適合率	再現率	F 値	適合率	再現率	F 値	
常磐線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
京浜東北線	WA+T	0.333	0.176	0.231	0.958	0.982	0.970	0.942
京葉線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
武蔵野線	WC+T	0.400	0.667	0.500	0.995	0.986	0.991	0.981
南武線	WA	N/A	0.000	N/A	0.991	1.000	0.995	0.991
埼京線	WC	0.714	0.357	0.476	0.988	0.997	0.993	0.985
高崎線	WC	0.650	0.867	0.743	0.997	0.991	0.994	0.988
東海道線	WA	0.500	0.172	0.256	0.971	0.994	0.982	0.965
山手線	WC+T	0.500	0.318	0.389	0.984	0.992	0.988	0.976
横浜線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
横須賀線	WA	0.706	0.632	0.667	0.987	0.991	0.989	0.979
京王線	WC	0.854	0.891	0.872	0.990	0.986	0.988	0.978
東京メトロ千代田線	WC	0.500	0.500	0.500	0.993	0.993	0.993	0.986
東京メトロ副都心線	WC+T	0.429	0.750	0.545	0.990	0.962	0.976	0.954
東京メトロ半蔵門線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
東京メトロ日比谷線	WA+T	0.333	0.500	0.400	0.993	0.986	0.989	0.979
東京メトロ南北線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
東京メトロ東西線	WC+T	0.700	0.636	0.667	0.988	0.991	0.989	0.980
東京メトロ有楽町線	WC	1.000	0.818	0.900	0.985	1.000	0.993	0.986
小田急線	WC+T	0.564	0.795	0.660	0.993	0.980	0.986	0.974
西武池袋線	WC+T	0.554	0.857	0.673	0.992	0.965	0.978	0.959
東武東上線	WC+T	0.448	0.500	0.473	0.986	0.982	0.984	0.969
都営浅草線	WC	1.000	1.000	1.000	1.000	1.000	1.000	1.000
都営三田線	WA	N/A	N/A	N/A	1.000	1.000	1.000	1.000
都営新宿線	WA	0.500	1.000	0.667	1.000	0.971	0.986	0.972
東急田園都市線	WA+T	1.000	0.667	0.800	0.997	1.000	0.999	0.997
東急東横線	WA+T	0.429	0.231	0.300	0.964	0.985	0.975	0.951

表 6.6: 連鎖率を用いる手法の路線ごとの予測結果

第7章 おわりに

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、鉄道運行トラブルの検出および影響の予測を行った。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および復旧の検出を試みた。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することが確認できた。また、トラブルの継続時間予測においては、継続時間の初期段階における予測手法と逐次予測手法の2つを提案した。初期段階の予測においては30分、45分、60以上長引くかどうかの予測において提案手法の有効性が確認できた。逐次予測においても、30分、45分、60以上長引くかどうかを予測する場合において提案手法の有効性が確認できた。また、逐次予測では直近の情報を重視しつつ、過去の情報を織り込むことの有効性も確認することができた。トラブルの連鎖予測においては、過去に発生した連鎖パターンのみ予測する手法と、未知の連鎖パターンにも対応できる予測手法の2つを提案した。評価実験の結果、過去に発生した連鎖パターンのみ予測する手法の方が再現率は下がるが、適合率およびF値は高いことが確認できた。また、比較的単純な連鎖パターンの予測において提案手法の有効性を示すことができた。

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、誤分類の考察で述べた通り、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．

参考文献

- [1] 株式会社シード・プランニング. 2012-2013 年版スマートフォン/タブレットのグローバル市場展望. 2012.
- [2] <https://twitter.com/twitterads/status/210867782361948161>.
- [3] <https://twitter.com/twitterads/status/210868480361242624>.
- [4] Adam Sadilek, Henry Kautz, and Vincent Silenzio. Predicting disease transmission from geo-tagged micro-blog data. *AAAI*, 2012.
- [5] D Clayton, M Hills, and A Pickles. Statistical models in epidemiology. *Oxford university press Oxford*, 41, 1993.
- [6] Eytan Adar, Daniel S. Weld, Brian N. Bershad, and Steven. Why we search: Visualizing and predicting. *WWW*, pages 161–170, 2007.
- [7] Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. Learning causality for news events prediction. *WWW*, pages 909–918, 2012.
- [8] Panagiotis T. Metaxas, Eni Mustafaraj, and Daniel Gayo-Avello. How (not) to predict elections. *IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing*, 2011.
- [9] C. Bizer, T. Health, and T. Berners-Lee. Linked data - the story so far. *IJSWIS*, 2009.
- [10] Eric Gilbert and Karrie Karahalios. Widespread worry and the stock market. *AAAI*, 2009.

-
- [11] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science* 2(1), pages 1–8, 2014.
- [12] Daniel Gruhl, R. Guha, and Ravi Kumar. The predictive power of online chatter. *KDD*, pages 78–87, 2005.
- [13] Haipeng Zhang, Mohammed Korayem, and David J. Crandall. Mining photo-sharing websites to study ecological phenomena. *WWW*, 2012.
- [14] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. *WWW*, pages 851–860, 2010.
- [15] J Kleinberg. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397, 2003.
- [16] X. Zhang and Shasham D. Better burst detection. *ICDE Proceedings of the 22nd International*, pages 146–149, 2006.
- [17] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. *KDD Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 336–345, 2003.
- [18] D. Shasha and Y Zhu. High performance discovery in time series. *Techniques And Case Studies (Monographs in Computer Science)*, 2004.
- [19] 蝦名 亮平, 中村 健二, and 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*, 9(2), 2010.

発表文献

1. 土屋圭, 豊田正史, 喜連川優. マイクロブログを用いた鉄道の運行トラブル状況抽出に関する一検討. 電子情報通信学会データ工学研究会, 研究報告情報基礎とアクセス技術 (IFAT), Vol.111(31), pp.175-180 (2013.07).
2. 土屋圭, 豊田正史, 喜連川優. マイクロブログからの鉄道の運行トラブル発生検出および継続状況の抽出. 第91回人工知能基本問題研究会 (2013.11)
3. 土屋圭, 豊田正史, 喜連川優. 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. 第6回データ工学と情報マネジメントに関するフォーラム (2014.03) (to appear)