早稲田大学審査学位論文
博士（人間科学）

位置情報付きツイートを利用した
気象・災害状況の可視化におけるノイズリダクション

Noise reduction for visualization of meteorological phenomena
and disaster damages using geo-tagged tweets

２０１９年７月

早稲田大学大学院 人間科学研究科

服部 充典
HATTORI, Mitsunori

研究指導担当教員： 西村 昭治 教授

目次

第1章 はじめに

1.1 背景

近年、台風の大型化や集中豪雨、局地的大雨（以降、ゲリラ豪雨と表記）、そして、地震などの事象により、日本各地において頻繁に被害が生じている。このような事象の被害としては、道路の冠水、家屋や地下街の浸水、河川の氾濫、土砂崩れ、建物の倒壊、さらには、交通機関の乱れや事故などが考えられる。そして、台風や集中豪雨については、それらによる被害を事前に回避、あるいは、最小限に抑えるために、気象庁のアメダス、気象レーダーや各自治体などの観測･監視機能に基づき、テレビやラジオ、そしてネット上のニュースなどを通じて、その状況と今後の予測などで注意が呼び掛けられている。

総務省消防庁（2010）によると、集中豪雨などによる人的被害を軽減させるために、迅速な状況の把握と、早めの避難行動の重要性が指摘されている。そのような中、ゲリラ豪雨、突風･竜巻、雹などの突発的な事象や、それらの事象および地震などによる被害状況、たとえば、河川の氾濫、浸水・冠水、交通機関の乱れや事故、停電、建物の倒壊などの可視化については、速報性･網羅性の観点から十分とは言えない状況にあった。

しかし、昨今の SNS の利用、スマートフォンの普及により、その場に居合わせた人々が、その場におけるさまざまな「状況」（現在降雨状態にあること、雨が上がったなどの状況が変化した状態にあること、現在降雨状態ではないことなど）を発信しやすい環境が整備されてきた。したがって、このような「状況」が迅速かつ網羅的に可視化･共有化されない、という課題解決を補完する環境が構築されつつある。

こうした中、SNS の活用という点では、特に、Twitter を活用した研究事例が数多く見られる。情報の速報性が高い、スポーツ中継やニュース番組などにおいて一体感を感じられる、緊急時の連絡手段としても活用できるなどの特徴があるからである。「ソーシャルメディア白書 2012」によると、Twitter 上の特徴として、１０代･２０代の若者は、他の年代と比べて、自宅でのツイート以外に、移動などの隙間時間を利用して、より多くの雑感を発信しているという調査結果が報告されている。

さらに、Twitter には、スマートフォンに付いている GPS 機能を利用し、位置情報を付与したツイートが可能であり、これにより、ツイートの投稿場所に関連した分析が可能となった。位置情報付きツイートの取得においては、API のエンドポイントとして、sample を指定する方法と locations を指定する方法（2.3.2 参照）の２つがある。

予備調査において、sample 指定の場合、日本語のツイートは、１時間で 75 件程度の抽出であったが、locations 指定の場合、１時間で 11,000 件程度の抽出が得られる結果となった。そのため、locations 指定での抽出により、位置情報付きツイートの取得数もある程度見込まれ

るため、位置情報付きツイートによる「状況」可視化のアプローチは、前述した、速報性･網羅性の課題を解決する１つのソリューションとなり得ることが考えられる。

一般的に、位置情報を利用して、特定の「状況」の可視化を行うには、ツイートからその「状況」を示すワードを指定して抽出し、そこに付加されている位置情報を用いて、地図上にプロット、もしくは、大字･町丁目レベルに変換して可視化することが考えられる。蛭田ら (2013) は、位置情報付きデータを抽出する際に、前述の方法にて、状況を抽出している。たとえば、降雨状況を知りたければ、次のような方法で「雨」を指定して抽出する。抽出例を図１に示す。

```
cat 2014_6.data | ggrep "雨"
2014-06-30T23:53:56+09:00 36.5744007 139.8572833 雨すごいな
2014-06-30T23:55:42+09:00 36.34333094 139.1559263 雨まじきちがい(´Д｀)
2014-06-30T23:55:55+09:00 43.7908456 143.8775251 雨やばいやばい！
```

図１ 雨の抽出例

図１は、2014 年の６月の期間において、前述の locations 指定で取得した位置情報付きツイートデータのファイル（2014_6.data）から、キーワード「雨」を指定して、抽出･表示している。cat は、ファイルの中身を出力し、「|」（パイプ）に繋げて、ggrep（ファイル内の文字列を検索するコマンド） へのインプットとして渡し、「雨」を含むレコード抽出を行う例である。また、図１において、「36.5744007 139.8572833」の部分は、位置情報（緯度、経度）である。「雨すごいな」の部分は、投稿されたツイート本文である。そして、「2014-06-30T23:53:56+09:00」の部分は、投稿された時刻（日本時間）を表している。

しかし、図１のような単純なワード指定による抽出方法では、精度に問題が生じる場合がある。このような抽出方法では、図２に示すとおり、その「状況」が生じていた場所･時間から離れた投稿、たとえば、気象情報（気温予報･降水確率など）、予想、振り返りや感想などのツイートも抽出してしまうからである（図２には、併せて、図１の例に挙げた位置情報から、大字･町丁目レベルに変換した例も示した）。また、蛭田ら（2013）は，位置情報が付加されていても、その場所とまったく関係のないツイート（ノイズ）が多いことを指摘している。

```
2016-04-30T18:07:47+09:00 沖縄県八重山郡与那国町与那国 04 日(水) 曇一時雨 最高 28℃ 最低 24℃ 降水確率 50% #天気 c4
2016-04-30T19:35:42+09:00 宮城県仙台市若林区南鍛冶町 コレが旨い！ 季節ですよね～ 明日は雨かな orz。。。
2016-04-30T20:45:00+09:00 宮城県柴田郡川崎町大字川内字藤折 昨日は時々雨だったけど今日は天気もったかな？？？
```

図２ ノイズとなるツイート例

したがって、「状況」を可視化する上では、このような可視化したい状況と直接関係しないツイート、つまり、ある目的にとって不要なツイートをノイズとして捉え、抽出時には、それらを除去することが不可欠となる（以降、ノイズリダクションと表記）。

一方、特定「地域」で生じている状況において、その地域から投稿されたツイートは、他の地域から投稿されたツイートと比較して、より信頼性が高いと考えられる。前者は 1 次情報であり、後者はリツイート、マスメディアや SNS 投稿を受けての投稿、つまり、2 次以降の情報になるからである。そのため、位置情報付きツイートをうまく活用できれば、例えば、被災地において、被災直後からのメディアでは報道されにくい現地の声を知る有用な手がかりとなり得る。

特定地域のツイート抽出による可視化においても、前述した特定「状況」の可視化と同様に、各ツイートに付与されている位置情報を利用し、事前に、広範囲で取得・変換（緯度・経度から大字・町丁目レベルに）しておく方法が考えられる。毎回、locations 指定にて特定地域のツイートを抽出するよりも、事前に変換しておくことで、地域名による地域の抽出操作が、より容易になるからである。しかし、このような方法で抽出されたツイートにおいても、精度に問題が生じる場合がある。たとえば、bot ニュース（機械的に投稿される各種情報など）、ツイート連携アプリ（Twitter と連動したアプリケーションからの投稿）などの、可視化したい状況に直接関係しないツイートまでも抽出してしまうからである。その例を図３に示す。「微小地震速報」を含むツイートは bot ニュース例、「I'm at 渋谷駅」を含むツイートは、ツイート連携アプリ例である。したがって、特定地域の状況可視化を行う上でも、可視化したい状況以外のツイート（ノイズ）のノイズリダクションが欠かせない。

2016-04-30T23:50:35+09:00 熊本県熊本市中央区出水七丁目【微小地震速報 熊本県 744/992】
2016/04/30 22:42:36 JST, 日本 熊本県 熊本市 出水８丁目 付近, M1.5, TNT2.7kg, 深さ 5.9km, MAP https://t.co/PkZYBnJmMI 1153
2016-04-30T23:58:13+09:00 東京都渋谷区桜丘町 I'm at 渋谷駅 (Shibuya Sta.) in 渋谷区, 東京都 https://t.co/px9t4P7qxj

図３ ノイズ候補例

1.2 先行研究

高橋（2013）は、そもそも、位置情報付きツイートについて、プライベート情報の問題などから、取得できる数が少なく、災害検知などの応用に使うには不十分であると言及している。

その上で、位置情報付きツイートを使わなくても、位置情報を含まないツイートからユーザーの位置情報を推定するために、局所性のあるイベントに対する発言の同期を集計することにより、各ツイートの投稿場所を推定する手法を提案している。

また、その際の実験の結果、同期発言（同一時間帯において、複数人が共通のイベントについてツイートすること）のあるイベントには、自然（雷、雪、地震、雨、虹、吹雪、台風、初雪など）、イベント（花火、魂祭、開港、七夕、祇園祭など）、災害（停電、人身、警報、冠水、勧告、断水など）、場所（福岡、札幌、名古屋、横浜、御堂筋、淀川など）、鉄道会社（相鉄、京王、西鉄、名鉄、京浜東北など）、その他（位、送料、無料、レポ、秒、容量、電気など）のグループに分けられることを明らかにしている。

ただし、本研究におけるレビューの結果、高橋（2013）の研究は、位置情報付きツイートを取得する際に、前述の sample 指定を用いており、その結果、位置情報付きツイートの抽出数が少なくなっていたのではないか、という事が判明した。しかしながら、高橋（2013）により明らかにされた、同期発言の多いイベントグループに関して、位置情報付きツイートを用いた可視化という点において、適用できる対象がより特定された事例として、非常に有用な先行研究として捉えている。その上で、本節では、状況を可視化していく上でのノイズリダクションに関する先行研究を概観する。

影澤ら（2014）は、降雨状況と電車遅延を捉えるために、位置情報付きツイートを抽出する際に、あらかじめ定義した簡易的なノイズフィルタを適用している。具体的には、降雨状況を捉えるために、「雨」、「アメ」、「あめ」を含むツイートを抽出し、電車遅延においては、「人身事故」、「遅延」、「遅れている」を含むツイートを抽出している。そして、その際のノイズリダクションとしては、ツイート文頭の「RT」もしくは「@」で始まるものは削除、複数の#が含まれるものは広告の可能性が高いと判断して削除、「I’m at」はツイート連携アプリ（チェックイン系）のための削除、そして、事前に bot として判断したアカウントによるツイートは削除している。

蛭田ら（2013）は、位置情報付きツイートを使用した可視化実験において、その場所とまったく関係のないツイートをノイズとし調査した結果、その割合が多い（44.07%）ことを明らかにした。そして、ノイズリダクションにおいては、事前に、発言の多いツイートの種類を分類し、分類された各グループに関するツイートを抽出するフィルタの構築を行い、ツイート抽出時のノイズを軽減するアプローチを採っている。つまり、影澤ら（2014）のノイズ用のフィルタ適用に対し、蛭田ら（2013）は、抽出したい対象をより限定的に絞り込むことによるノイズ

リダクションのアプローチである。具体的には、居場所の報告に関するツイートとして、現在地の場所や施設などについて言及している発言を、食事、天候、帰宅、地震のグループに分けている。たとえば、帰宅に関しては、「帰宅」、「帰った」、「帰着」、「帰還」、「朝帰り」のキーワードを事前に用意して、フィルタとして定義している。

岸ら（2013）は、東日本大震災発生直後一週間のツイートデータから、位置情報付きツイートを取得し、bot などのアカウント、いわゆるノイズの原因となるツイートを除外し、時刻、位置情報、user ID、内容などを含むツイートデータのリストを作成した。このときのノイズ除去については、事前にノイズフィルタを作成する際に、同一 userID での発言が多いものを bot として定義している。そのうえで、各ツイートデータを周辺地域の人間活動指標と捉え、カーネル密度推定を用いた位置情報付きツイート密度の逐次的推定による分布図を作成することで、震災直後からの情報の発信状況の可視化を試み、位置情報付きツイートによる復旧・被害状況の逐次的な把握可能性を示している。

Guo ら（2014）においても、岸ら（2013）と同様に、bot などの非個人ユーザーのツイートをノイズとして捉え、事前に bot ID を特定・抽出する方法を提言している。

Ward ら（2012）は、特定ドメイン（TV のタイトル）に関するツイート抽出率の向上のために、事前に作成した 1 組の検索ワードを用いている。さらに、抽出精度を上げるために、ツイート外からのリソースを取得し、分類器を作成・使用している。

1.3 研究目的

前述した先行研究において、従来では、特定「状況」の可視化には、事前にノイズとなるキーワードや、抽出したい状況に関するキーワードを定義したノイズリダクション、および、botなどの非個人ユーザーのアカウントを特定するノイズリダクションのアプローチが適用されてきた。しかし、個人ユーザーの発言において、その「状況」が、今現在もその場所で続いているのか考慮するための「時間」と「場所」との両側面からのノイズリダクションは考慮されていない。そのため、「明日は雨かな？」、「今朝の雨は酷かった」、「埼玉では豪雨らしいね」などのツイート除去には対応できず、そのため、誤抽出といった抽出精度への影響が懸念される。

また、従来のイズリダクションでは、ツイート内のテキスト（BotID 含む）処理を実施する自然言語処理のみのアプローチとなっており、この場合、抽出したツイートについては、その「確からしさ」（そのツイートは、本当に正しいのかという度合い）に疑念が生じる。そのため、デマ情報や嘘のツイートを抽出してしまう可能性も考えられる。

さらに、従来手法でのノイズフィルタ生成においては、フィルタの候補となるキーワードの機械的収集、および、そこからフィルタを機械的に生成する、といった機械的な処理は行われていない。そのため、そこには、少なからず時間と労力（コスト）がかかり、それにより、主観的なアプローチが入り込む可能性も考えられ、その結果、抽出精度への影響が懸念された。

このように、従来の特定「状況」の可視化におけるノイズリダクションでは、a）抽出精度、b）情報の信ぴょう性、c）人手によるフィルタ生成のための時間と労力（コスト）の課題が存在した。

一方、特定「地域」の状況可視化には、多様な状況を捉えるために、フィルタ機能において、ノイズをリダクションしながらも、より多くの正解データを抽出する、つまり、再現率（4.3.1 参照）を重視したアプローチが考えられる。そして、再現率を重視する上では、フィルタ生成において、事前に、より多くのノイズツイートを集め、そこから、ノイズツイートだけに該当する条件設定が重要となる。しかし、前述した先行研究において、従来では、より多くのノイズツイートの識別･収集とノイズツイートだけに該当する条件設定においては、事前の人手による処理（フィルタ生成）が行われており、少なからず時間と労力（コスト）の課題（前述 c））が存在した。

そこで、本研究では、特定状況の可視化、および、特定地域状況を可視化するための２つのノイズリダクション手法について、これら３つの課題である、a）抽出精度、b）情報の信ぴょう性、c）人手によるフィルタ生成のための時間と労力（コスト）を解決するために、従来手法と異なるアプローチに取り組むこととした。具体的には、特定状況可視化のためのノイズリダクションでは、主に、従来の自然言語処理だけの手法に位置情報をベースとした近距離法（後

述）を加えて、a）抽出精度、b）情報の信ぴょう性の課題解決を試みた。また、特定地域の状況可視化のためのノイズリダクションでは、特定地域のツイート抽出時（ノイズリダクション時）に、機械的にフィルタ生成を行うことで、c）人手によるフィルタ生成のための時間と労力（コスト）の課題解決を試みた。詳細については、次の項で説明する。

1.3.1 特定状況可視化のためのノイズリダクション

特定状況可視化の際にノイズとなるツイートは、例えば、降雨状況の可視化において、「雨」でツイートを抽出した場合、「明日は雨かな？」（予想）、「今朝の雨は酷かった」（振り返り・感想）、「雨が降ったらずぶ濡れ」（仮定）、「明日は雨降らないで〜」（願望）、「晴耕雨読」（状況ワードに紐付く名詞）、「雨止んだ」（状況変化）、などが考えられる。さらに、「【13:00 時点 22.3℃ 96%】 25.0~22.0℃ ★警報:暴風/波浪__★注意報:大雨/洪水/雷＝宮崎県宮崎市 #miyazaki #宮崎」などの、bot 系ニュースに関するツイートも考えられる。また、「え、雨降っちゃうの？ タイミングわるー」、「雨〜」などを抽出した場合、一見正しいように見えるツイートにおいて、本当に正しいのか、という疑念が少なからず生じる。

このような、今、その場所で生じている状況とは異なるノイズツイートの除去漏れによる抽出精度の課題、情報の信ぴょう性に対して、本ノイズリダクションの手法では、従来の自然言語処理だけによるアプローチとは異なる、位置情報をベースとした近距離法を適用した。近距離法とは、位置情報（各ツイート発生地点間の距離）、時間軸（同一時間帯のツイート）、複数人のツイート（複数人による同じ話題の投稿）に着目した、ノイズリダクションである。複数の人が、同じ場所で、同じ時間帯において、ある同じワード（たとえば「雨」）を含むツイートのみを抽出した場合、その結果については、情報の信ぴょう性の向上につながるだけではなく、前述した、その場所で生じている状況とは異なるノイズツイート除去、つまり、抽出精度向上にも、高い効果が見られるのではないか、という仮説に基づくアプローチである。

ただし、これだけでは、抽出精度の課題への対応が不十分であると判断し（3.1.3 参照）、この近距離法に、従来からの自然言語処理のアプローチを組み合わせることとした。その際に、さらに、その自然言語処理にも改良を加えることとした。具体的には、前述にて指摘した、「予想」、「振り返り･感想」、「願望」などの時間軸（「今朝」、「明日」などのワードの特定）への対応、機械的処理（フィルタの基となるデータの機械的収集、フィルタの機械的生成）、および、特定状況に関するストップワードに着目した改良である。このように、従来手法と近距離法を組み合わせることで、従来の自然言語処理だけの手法と比べて、情報の信ぴょう性の課題を解決し、さらに、抽出精度が向上するのではないかと考える（仮説）。

そして、これらの実験方法として、現時点の降雨状態（以後、降雨状況と表記）の実験を行い、対象とした２つの課題（抽出精度、情報の信ぴょう性）解決の検証を通して、開発した手

法が従来手法より優れているかの有用性を評価した。気象現象における降雨状況を選定した理由は、アメダスの Web サイトから、過去の降雨量データを取得･利用することが容易であり、これを利用することで、より客観的に手法の精度評価にも使える利点があるからである。

手法の評価では、機械的に評価･検証を実施するために、事前に評価システムを構築し、そのシステムに、前述した気象庁･アメダスから取得した降雨データを登録して利用した。さらに、開発した手法について、４つの降雨タイプ（4.3.1 参照）についても適用し評価を行った。

1.3.2 特定地域の状況可視化のためのノイズリダクション

一般的にノイズリダクションには、次の方法が考えられる。ａ）調査用（取得済み）ツイートから中身を確認しながらノイズとなるツイートを特徴づけるキーワードの特定･設定を行う方法（影澤ら（2014）など）。ｂ）教師あり学習による分類器を作成する方法。ｃ)教師なし学習による分類器を作成する方法。ただし、ｃ)においては、抽出したいツイートの特徴を指定できないため、意図しない分類の可能性が考えられる。したがって、本目的でのノイズリダクションには、ａ)、ｂ)の方法が適当と考えられるが、人手による事前の作業が必要となる。なお、本稿では、ノイズリダクション時のツイートを用いず、調査用ツイートから生成したフィルタや分類器生成などを事前の作業と定義する。一方、ノイズリダクション時のツイートのみを用いたフィルタなどの生成処理を、事前ではなく、実行時の処理と定義する。

また、前述のとおり、特定「地域」の状況可視化には、再現率を重視した抽出方法が考えられる。これは、多少のノイズが混入してでも、より多くの正解（状況を表す）ツイートを抽出するアプローチである。その場所で何が起きているかを捉えるために、より多くの正解ツイートの抽出が求められるからである。そして、再現率を重視する上では、フィルタ生成において、事前に、より多くのノイズツイートを集め、そこから、ノイズツイートだけに該当する条件設定を行う必要がある。前述のとおり、従来、これらの作業は、事前の人手による時間と労力（コスト）を要している、という課題に対して、本ノイズリダクションの手法では、従来手法とは異なる、事前のコストをかけずに実行時に機械的に生成されるフィルタを用いるノイズリダクションを試みた。

実現方法には、名詞の形態素 n-gram 組合せ頻度の高いものを利用する生成方法を適用した。その理由として、フィルタの機械的生成の実現が容易であり、これに該当するツイートは、ノイズの可能性が高く（少なくとも、リツイート系（主にデマ）、bot 系、ツイート連携アプリが考えられる）、また、該当するツイートのみが除去されるため、その結果、事前のコスト削減に加え、再現率の高い抽出が期待できるからである（仮説）。さらに、n-gram のｎ数での正解ツイートの抽出（再現率）制御も考えられる。

本ノイズリダクションの手法では、昨今、頻発している災害による被災地を対象とした。様々な被災地の状況（災害状況）をより早く捉えたい場合に、実行時に機械的に生成される本手法は、課題である事前の人手による時間と労力（コスト）をかけた準備作業を省くことが可能となり、それにより、従来手法と比べて、その有用性は高いと考えられるからである。

実験としては、2016 年 4 月 14 日に発生した熊本地震において被災した熊本市、2015 年 9 月に発生した関東･東北豪雨において鬼怒川の越水･漏水･決壊により水没した、茨城県の常総市･境町から投稿されたツイートを使用し、本手法でのノイズリダクションにおいて、一定程度の再現率（精度）が見られるかについての検証を実施した。

手法の評価において、本研究で可視化したい状況は、人から投稿された被災地現場の避難･生活状況であり、次のとおり定義した（同時に正解の判定基準となる）。それらは、地震のゆれ、避難所、炊き出し、ライフライン（電気、ガス、水道、電話、通信設備、鉄道、物流機関）、断水、風呂･トイレ、道路、飲食料、日用品、および、飲食店･コンビニ･スーパー･ガソリンスタンドの店内情報（品薄･入荷，営業状況）を含むツイート（もしくは、ツイート内に URL が記載されていた場合、その参照先が存在し、これらの状況を示す画像や文章が含まれるツイート）とした。

そのため、避難･生活状況に直接関連しない話題、感想（例:怖かった）、意見、声がけ（例:どうかご無事で）、質問･確認（例:無事？）、デマ（例:地震のせいでうちの近くの動物園からライオン放たれたんだ）、bot（人以外から機械的に投稿される各種情報）、ツイート連携アプリのツイートをノイズとして定義する。ただし、感想において「ゆれが凄くて怖かった」など第三者からみて状況が分かる表現が含まれている場合、非ノイズとする。また、昨今使われている「やばい」を含むもの（例:ゆれやばい）も状況を表現する扱いとして非ノイズとする。

1.4 本論文の構成

本論文の構成は、以下のとおりである。

第２章では、本研究における方法について述べる。具体的には、準備作業、手法開発に使用したシステム環境、データの取得方法についてである。

第３章では、特定状況、および、特定地域状況の可視化において、前述の各課題に対し、従来手法とは異なる（新規性のある）、２つのノイズリダクション手法を提案する。

第４章では、提案した手法の評価実験について述べる。具体的には、実験に使用するツール・データセット、方法についてである。

第５章では、実験結果について整理する。

第６章では、提案した２つの手法において、従来手法による結果（課題）と比較し、本手法がより優れており課題解決に寄与するのか（有用性があるか）、について考察を行う。

最後に、第７章で本論文の結論（まとめ・結論）を述べる。

なお、付録として、本研究にて構築した４０程度のツールのソースコードを加えた。

第2章 方法

2.1 準備作業

本研究における手法開発の準備作業は、次のとおりである。最初に、ツイートデータを取得・蓄積するためのシステム環境を構築した。つぎに、ツイートデータを取得するためのコンピュータープログラム（以降ツールと表記）の構築、取得したツイートデータを蓄積するためのツールの構築、分析を容易にするためのツールの構築を行った。

さらに、取得したツイートデータが、今後の可視化手法の開発に使えるのかという検証をするためのツール、具体的には、取得したツイートデータの件数を取得するなどの操作を行うためのツール、そして、ある時間帯から事象・状況を抽出するためのツールの構築を行った。

また、抽出後に可視化するためのツール（地図上へプロットするもの、位置情報から大字・町丁目レベルの場所情報に変換するもの（以降、逆ジオコーダと表記））についても構築した。

また、本研究で構築した各ツールは、「4.1 ツール」の節で整理した。

2.2 システム環境

ツイート取得のためのシステム環境（以降、サーバと表記）は、Mac mini late 2012 (2.5GHz デュアルコア Intel Core i5、メモリ 4GB 1600 MHz DDR3)、OS は OS X 10.9.4 (Mavericks)、文字コード（エンコーディング）は UTF-8 を使用した。また、フィルタを含め、各種ツールの構築には、Python 言語：2.7.6 とシェル（bash）スクリプト、OS コマンドを使用した。

位置情報付きツイートデータの取得には、「【php】twitter Streaming API の statuses/filter を試す」を参考に、日本全体の範囲を指定した Streaming API を使用した。その際に、取得するツールを容易に構築するために、tweepy ライブラリ（ver. 2.2）を使用した。取得したツイートデータの保存用データベースは、MongoDB 2.4.10 を使用した。データベース操作は、Python から MongoDB を利用するために pymongo ライブラリ（ver. 2.7）を使用した。さらに、位置情報（例：緯度 34.785, 経度 135.529）から大字・町丁目レベル位置参照情報に変換するために「簡易的な逆ジオコーディング」を参考に逆ジオコーダを構築した。変換用の DB には SQLite 3.7.13 を使用した。

特定状況可視化のためのノイズリダクション手法における抽出精度の評価やコーパス作成の前提として、気象庁・アメダスの Web サイトから、「過去の気象データ」を取得した。そして、当手法の実験では、東京都における過去の降雨情報（1 時間毎）について、データベース化を行った。このデータベースにおいても、SQLite 3.7.13 を使用した。また、抽出精度の分析・検定には、R 3.3.2 (2016-10-31)、および、「こんにちは統計学：Python による χ２乗検定・ｔ検定・Ｕ検定・分散分析・多重比較・相関係数の計算」サイトの「対応のあるノンパラメトリック・データの多重比較」機能を活用した。

2.3 データ取得方法

2.3.1 サーバ運用

サーバの運用において、サーバ起動時に必要なツールを起動させるための仕組みを整備した。この仕組みは、Mac 上のシステム環境設定の『ユーザーとグループのログイン項目』で、起動させたいコンピュータープログラム：startup_prod.sh を登録することで実現できる。startup_prod.sh 内で、メッセージを取得するバッチファイル（get_message_prod.sh）を呼び出し、呼び出された get_message_prod.sh は、メッセージを取得する本体の機能（test4xx.py）を呼び出すようにした。この本体の機能は、60 秒間処理して、終了するようにした。そして、get_message_prod.sh は、本体機能が終了した後、30 秒間 Wait（Sleep）してから、再度、test4xx.py を呼び出すようにした。この理由は、主に、プロバイダーのネットワーク負荷軽減への配慮からである。

次に、サーバ監視ツールとして、manage_tweet.sh、そして、この中から呼び出される Python のプルグラムである system_manage.py を構築した。このツールは、1 時間の処理状況やデータベースへの蓄積件数を照会して、その結果を、gmail を使用して自分宛にメールを送信する機能である。また、作成した監視ツールを、１時間に１回、サーバ上で実行させるために、Mac 上で提供されている launchd を使用した。

次に、外部からのアクセスを遮断するために、ファイアウォールも稼働させた。OS X 10.9.4（Mavericks）では、デフォルトのファイアウォール機能が付いているが、ポート番号単位での細かい設定ができないため、パケットフィルタ系ファイアウォール:PF を使用した。その際に、操作を簡単にするために、IceFloor というツールを導入した。自宅のネットワークは、192.168.1.x の設定であるが、サーバへの Inbound アクセスは、他のネットワーク（192.168.2.x）からのアクセスのみしか許さない、という設定で運用した。これにより、外部からのアクセスを遮断できるからである。しかし、この状況では、ツールを構築しているクライアント環境からのアクセスができなくなってしまうため、都度、ファイアウォールを解除して運用することとした。そのため、サーバの mac mini 用の操作用に、液晶モニターを別途用意した。

データベースの運用においては、Index を適宜作成した。MongoDB は、スキーマレスでデータを store できるという利点がある。一方、データの照会において、たとえば、数百から数千万件のデータの中から、『条件なし』での照会についての応答時間は、とくに問題はなかった。しかし、『条件付き』での照会においては、応答時間が悪化した。そのため、よく使用する条件に対しての Index を作ることにより、この問題が軽減されたため、データを１度蓄積してからの分析処理を行うアプローチの場合には、Index の検討･実装が１つのキーポイントとなる。

2.3.2 データ取得

データを常に取り続けるために、Twitter から提供されている Streaming API を使用することにした。Streaming API は、User streams、Sites streams、Public streams の３種類が存在していた（2014 年５月時点）。

User streams は、単一ユーザーの Tweets/Timeline を取得できる API である。Site streams は、『Site Streams is currently in a limited beta. Access is restricted to whitelisted accounts.』ということで、利用においては、不明な点があるため、本研究では、見送っている。Public streams は、API の利用制限がなく、Twitter 上のより多くのユーザーのツイートデータが取得できる、という点から、この方法を採用した。

Public streams には、さらに、３つのエンドポイント（接続先）が存在していた。それは、statuses/firehose、statuses/sample、statuses/filter の３つである。そして、それぞれに対応する URL が用意されていた。statuses/firehose は、全ユーザーのすべてのツイートを取得できるが、これは、特別なアカウントだけ（企業など）に提供される。statuses/sample は、『Returns a small random sample of all public statuses.』ということで、statuses/firehose をより軽量化した位置づけと考えられる。statuses/filter は、フィルタを set すると、それにマッチしたツイートが取得できる。

したがって、本研究では、Streaming API での statuses/sample と statuses/filter の２つの方法で実験を試みた。その際に、自宅のネットワーク環境においては、『帯域に断続的かつ大幅な負荷をかけるようなことをしてはいけない』というプロバイダーの規約があったため、前述のとおり、Streaming API の使用方針を次のようにした。それは、Streaming API により 60 秒間ツイートデータを取り続け、その後、データ取得ツールの処理を終了し、30 秒後に、再度、データ取得ツールを起動する、ということを繰り返し実行することで、帯域の負荷を軽減させるという方針である。また、１度処理を終了させることで、システム上において確実に処理の同期点が取られ、ネットワーク障害やシステム障害時におけるデータ消失などのリスク軽減にもつながると考えている。

このようなことを踏まえ、研究当初は、statuses/sample を使用して、ツイートデータの取得を試みた。そして、１日に取得できるツイートデータは、280 万件程度であるという結果が得られた。しかし、日本語、かつ、位置情報付きツイートデータ、つまり、概ね国内でつぶやかれていると考えられる位置情報付きツイートデータは、１日、わずか 2,000 件程度（1 時間あたり約 80 件）であることが分かった。そのため、取得したツイートデータから、ある時間帯の状況を捉えるために、例えば、1～2 時間の時間間隔内で、ある状況を特徴づける特定のキーワードで絞って抽出した際に、想定以上にデータが少なく可視化できなかった。

そのため、エンドポイントを statuses/filter、オプションを locations に set して、日本列島をカバーする座標（緯度･経度）情報を指定し、データ取得を試みた。その結果、2014/06/11 から 6/25 までの 2 週間において、１日約 27 万件（1 時間あたり約 1.1 万件）の位置情報付きツイートデータの抽出結果が得られた。図４にその状況を示す。

座標の指定は、抽出したい範囲をブロック（南西の経度・緯度、北東の経度･緯度）で囲むことで、ツイートデータを取得できることが分かった。本研究では、日本全土のツイートを抽出するために、ツール内で、２つのブロックを連続して指定（locations=[123, 24, 146. 2, 39. 0, 138. 4, 33. 5, 146. 1, 46. 20]）した。図５にその抽出範囲例を示す。

以上により、本研究では、エンドポイントを statuses/filter、オプションを locations に set する方法で進める事とした。

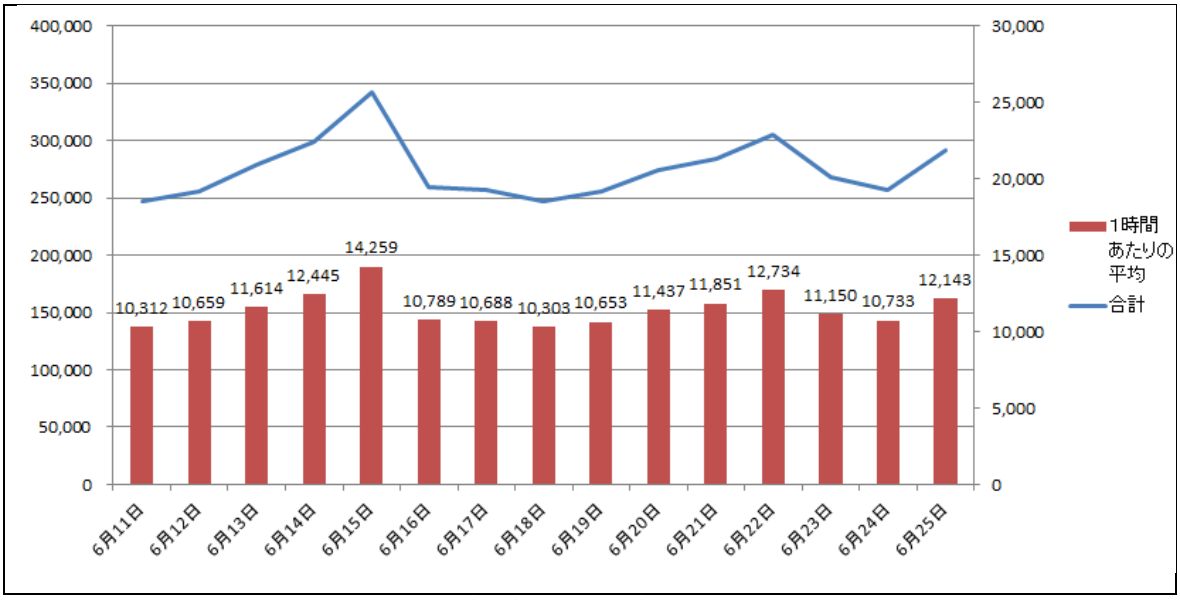


図４ 2014/06/11 から 6/25 までの取得状況

図５ 抽出範囲

2.3.3 データ蓄積

研究当初は、tweepy の on_status メソッドを使用して、一つのツイートデータ上から１０項目程度を String 属性として取得し、それをデータベースに保存していた。しかし、今後の研究の広がりを考慮し、すべての項目を取得することとし、そのために、on_data を使用してデータベースに保存するように変更した。データベースに保存した JSON 構造のツイートデータを図６に示す。

また、Twitter から取得したツイートデータの作成日（created_at）が、UTC（GMT）時間になっており、また、そのフォーマットも『Mon, 23 May 2011 12:23:50 +0000』という出力形式であったため、分析においては扱いづらい状況にあった。そのため、データベースへの保存時には、作成日を JST に変換し、また、『2014-04-19T01:00:36+09:00』というような ISO/W3C 形式に変換した。これにより、ある時間帯のツイートデータについて、容易に抽出できるようになった。

```
{
	"_id" : ObjectId("5397204f9fd571033a84e6b3"),
	"contributors" : null,
	"truncated" : false,
	"text" : "@kulukuluhaya @kakinotane1127 てか就職できたん？",
	"in_reply_to_status_id" : NumberLong("476380854483046400"),
	"id" : NumberLong("476381334592421888"),
	"favorite_count" : 0,
	"source" : "<a href=¥"http://twitter.com/download/iphone¥" rel=¥"nofollow¥">Twitter for iPhone</a>",
	"retweeted" : false,
	"coordinates" : {
		"type" : "Point",
		"coordinates" : [
			139.63444126,
			35.65257683
		]
	},
	"entities" : {
		"symbols" : [ ],
		"user_mentions" : [
			{
				"id" : 635268223,
				"indices" : [
					0,
					13
				],
				"id_str" : "635268223",
				"screen_name" : "kulukuluhaya",
				"name" : "はやと"
			},
			{
				"id" : 1968364230,
				"indices" : [
					14,
					29
				],
				"id_str" : "1968364230",
				"screen_name" : "kakinotane1127",
				"name" : "しょうふぇい"
			}
		],
		"hashtags" : [ ],
		"urls" : [ ]
	},
	"in_reply_to_screen_name" : "kulukuluhaya",
	"id_str" : "476381334592421888",
	"retweet_count" : 0,
	"in_reply_to_user_id" : 635268223,
	"favorited" : false,
	"user" : {
		"follow_request_sent" : null,
		"profile_use_background_image" : true,
		"default_profile_image" : false,
		"id" : 261121621,
		"profile_background_image_url_https" : "https://abs.twimg.com/images/themes/theme1/bg.png",
		"verified" : false,
		"profile_image_url_https" :
"https://pbs.twimg.com/profile_images/475649111987720192/2BiNoYLM_normal.jpeg",
		"profile_sidebar_fill_color" : "DDEEF6",
		"profile_text_color" : "333333",
		"followers_count" : 264,
		"profile_sidebar_border_color" : "C0DEED",
		"id_str" : "261121621",
		"profile_background_color" : "C0DEED",
		"listed_count" : 1,
		"is_translation_enabled" : false,
		"utc_offset" : 32400,
		"statuses_count" : 6541,
```

```
                "description" : "3中→芋高→農大ばいびじ3年",
                "friends_count" : 288,
                "location" : "群馬ー世田谷",
                "profile_link_color" : "0084B4",
                "profile_image_url" :
"http://pbs.twimg.com/profile_images/475649111987720192/2BiNoYLM_normal.jpeg",
                "following" : null,
                "geo_enabled" : true,
                "profile_banner_url" : "https://pbs.twimg.com/profile_banners/261121621/1399344835",
                "profile_background_image_url" : "http://abs.twimg.com/images/themes/theme1/bg.png",
                "name" : "TG",
                "lang" : "ja",
                "profile_background_tile" : false,
                "favourites_count" : 309,
                "screen_name" : "taiga0213",
                "notifications" : null,
                "url" : null,
                "created_at" : "2011-03-05T17:24:18+09:00",
                "contributors_enabled" : false,
                "time_zone" : "Irkutsk",
                "protected" : false,
                "default_profile" : true,
                "is_translator" : false
        },
        "geo" : {
                "type" : "Point",
                "coordinates" : [
                        35.65257683,
                        139.63444126
                ]
        },
        "in_reply_to_user_id_str" : "635268223",
        "lang" : "ja",
        "created_at" : "2014-06-11T00:12:15+09:00",
        "filter_level" : "medium",
        "in_reply_to_status_id_str" : "476380854483046400",
        "place" : {
                "full_name" : "世田谷区, 13",
                "url" : "https://api.twitter.com/1.1/geo/id/1b0cfebe3424e14f.json",
                "country" : "日本",
                "place_type" : "city",
                "bounding_box" : {
                        "type" : "Polygon",
                        "coordinates" : [
                                [
                                        [
                                                139.58364900596,
                                                35.5907059995056
                                        ],
                                        [
                                                139.58364900596,
                                                35.6790149962051
                                        ],
                                        [
                                                139.685946997143,
                                                35.6790149962051
                                        ],
                                        [
                                                139.685946997143,
                                                35.5907059995056
                                        ]
                                ]
                        ]
                }
    }
```


図６ JSON 構造のツイートデータ例

第3章 開発したノイズリダクション手法

本章では、本研究において開発した２つの手法について述べる。最初に、特定状況を可視化するためのノイズリダクションの手法、次に、特定地域の状況を可視化するためのノイズリダクションの手法について説明する。

3.1 特定状況可視化のためのノイズリダクション

特定状況可視化のためのノイズリダクションにおいて、本手法では、抽出精度・情報の信ぴょう性の課題に対応するために、従来手法の自然言語処理（単ワード・複数ワード指定による抽出・除去）に、新たに、主に、位置情報と時間軸をベースに考慮した処理、および、機械的処理を加えた。

その結果、構築した手法は、次の２つとなった。１つ目は、自然言語処理に、ノイズフィルタの基となるデータの機械的収集、ノイズフィルタの機械的生成処理、時間軸（「今朝」、「明日」などのワードの特定）、特定状況に関するストップワードを加えた手法で、本研究では、NLP（Natural Language Processing）法と定義した。２つ目は、これまでの自然言語処理とは別に、位置情報（同一場所）、時間軸（同一時間帯）、複数人の投稿（複数人が同じ話題で投稿している）の条件にてフィルタする手法で、本研究では、近距離法と定義した。

そして、この２つを組み合わせた手法が、本研究で開発・提案する１つ目の「特定状況可視化のためのノイズリダクション」の手法である。以降、組み合わせ法を構成する各手法（NLP 法、近距離法）、および、組み合わせ法（特定状況可視化のためのノイズリダクション）について説明する。

3.1.1 NLP 法

NLP 法の概念図を図 7 に示す。本手法は、一定の時間で取得したツイート群から特定状況ワードでの抽出後（図 7 内①）に、事前に生成した 3 つのフィルタを適用するノイズリダクション（図 7 内②）である。各フィルタについて、順を追って説明する。1 つ目はメインフィルタ（図 7 内ａ））である。このフィルタは、ノイズとなり得るデータを人手によらず機械的に収集する部分と、収集したデータ（ノイズコーパス）からノイズフィルタを機械的に生成する部分とに分けられる。機械的収集には、より客観的なノイズフィルタ生成につなげるために、評価システム（4．1．4 参照）を利用している。また、ノイズフィルタの機械的生成時には、形態素 N-gram（詳細は後述）を利用している。文字単位ではなく、形態素を単位とすることで、単なる N-gram での抽出と比べて、ツイートの文意を、より捉えやすくなることが期待できるからである。2 つ目は時間軸フィルタ（図 7 内ｂ））である。このフィルタは、特定状況の可視化において、ノイズとなり得る過去・未来時制を含むワードをノイズフィルタとして定義してい

る。3 つ目はストップワードフィルタ（図 7 内ｃ））である。このフィルタは、可視化したい特定状況に合わせて、主に、bot によるツイート（似た文体で繰り返し投稿されるのが特徴）に含まれるワードを識別・設定している。

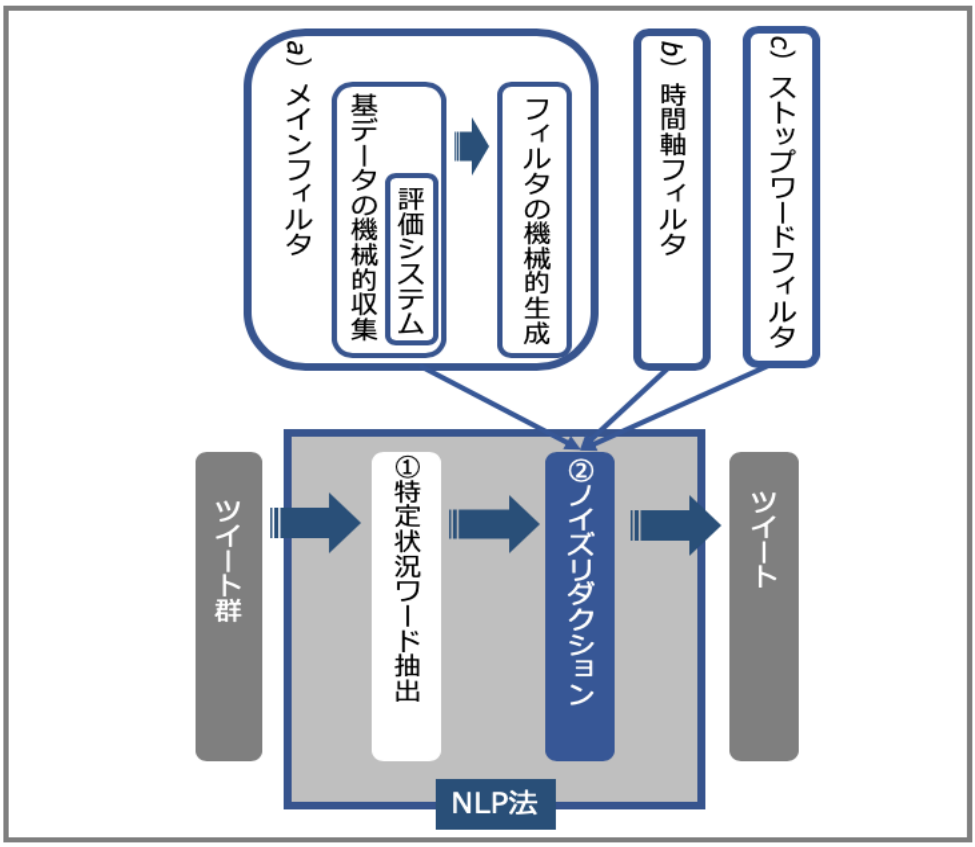


図 7 NLP 法概念図

本手法の実験においては、降雨状況を対象とし、1 つ目のメインフィルタにおいて、ノイズコーパスを作成する上で使用したデータは、東京都の観測所のある地域から、降雨量のデータが取得できた西多摩郡、八王子市、世田谷区、千代田区、江戸川区、大田区、青梅市、練馬区、府中市の 9 地域のツイートである。また、作成したノイズコーパスから、自然言語処理にて、機械的に、「雨が止んだ」、「今朝の雨は凄かった」、「明日は雨かな」などのノイズとなり得るツイートをノイズフィルタとして抽出・生成するために、N-gram については、3-gram、つまり、形態素 3-gram とした。（図８ ②形態素 3-gram 頻度部分）。なお、形態素 3-gram とは、形態素解析後の単語（品詞などで分解された意味を持つ最小限の単位）と N-gram における 3-gram とを組み合わせた手法である。また、N-gram は、任意のツイートなどの文章において、先頭から順次、任意の n 文字で連続した文字列で検索していく手法である。たとえば、「今

朝の雨は凄かった」について、形態素ではなく文字単位 3-gram で処理する場合、図９①のように分解される。また、形態素単位での 3-gram の場合は、図９ ②のようになる。

また、河居ら（2013）によると、形態素 3-gram および 4-gram で短文検索を実施した際に、高い精度での抽出結果（最大で 79.5%）が報告されている。さらに、本手法では、各形態素文字列から、頻度の高い、かつ、「状況」を特定するワード（本実験では、「雨」）と共起する文字列を抽出･適用した。その「状況」をより捉えられると考えたからである。加えて、本手法では、図９ ③で示すとおり、形態素解析において、助詞の文字を除くこととした。助詞がなくとも、文意への影響がないと判断したからである。

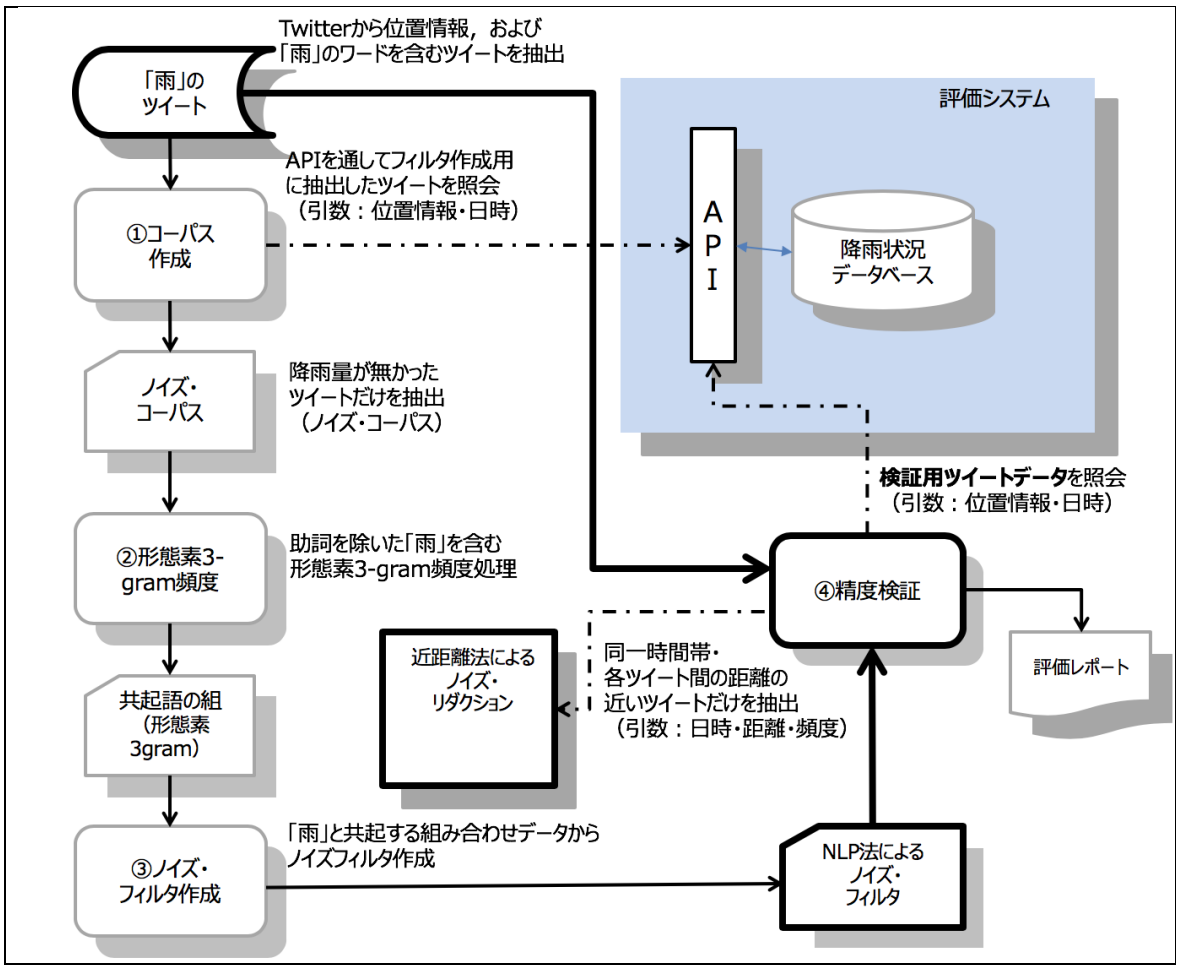


図８ 評価システムを利用した実験の流れ

２つ目の時間軸フィルタにおいて、本手法の実験では、抽出した「雨」と共起する文字列の組から、「雨」で開始する組（例：雨凄かった）と「状況」可視化のノイズとなり得る過去・未来時制を含むワード（「明日」、「昨日」、「今朝」、「先日」、「夕方」、「昨晩」、「深夜」、「明け方」）で開始する組（例：「今朝」「雨」「凄かっ」）を抽出した。その際に、「雨」で開始する文字列組の場合のみ、その組の出現頻度２以上のみをフィルタとして適用す

ることとした。ノイズコーパス作成時の観測所管轄内での誤抽出（同じ観測所でも、降雨状況にある地域と無い地域が生じる可能性も考えられるため）の影響を考慮して、正しいツイートから作成された形態素の組（例：「雨」「降っ」「いる」）が入りこんでいた場合に、極力排除するためである。本手法では、この実装として、ggrep -v -E″xxxxx″ における xxxxx に該当する部分を機械的に構築した。図１０に、具体例を示す（図８ ③ノイズフィルタ生成部分）。

今朝の雨は凄かった
① 3-gram（文字単位）
→ 今朝の 朝の雨 の雨は 雨は凄 は凄か 凄かっ かった った今 た今朝
② 3-gram（形態素単位）
→ 今朝の雨 の雨は 雨は凄かっ は凄かった 凄かった今朝 た今朝の
（参考）形態素解析による分解
→ 今朝（名詞） の（助詞） 雨（名詞） は（助詞） 凄かっ（形容詞） た（助動詞）
③ ②から，助詞を除く
→ 今朝雨凄かっ 雨凄かった 凄かった今朝 た今朝雨

図９ 3-gram と形態素 3-gram の分解例

３つ目のストップワードフィルタには、状況毎に関するニュースや情報を特徴づけるワードを、ストップワードとして登録しておき、抽出時にツイート内にそのワードを含む場合は、ノイズと判断して除去することとした。本実験では、降雨状況を対象としたため、「時点」、「降水確率」、「天気」、「予報」、「警報」、「注意」、「雨量」、「ニュース」、「news」、「速報」をストップワードとして登録した。これらの登録基準として、主に、bot によるツイート（似た文体で繰り返し投稿されるのが特徴）の気象ニュースに含まれるワードを設定した。

雨.*だ.*今日|雨.*な.*です|雨.*な.*晴れ|雨.*だっ.*た|雨.*でし.*た|雨.*やん.*た|雨.*やん.*だ|雨.*嫌い.*だ|雨.*打た.*れ|雨.*止ん.*た|雨.*止ん.*だ|雨.*降っ.*た|雨.*降ら.*ず|雨.*降ら.*れ|雨.*降る.*前|雨.*降っ.*ない|雨.*降っ.*なく|雨.*降っ.*止ん|雨.*降ら.*ない|雨.*降ら.*なく|雨.*降り.*そう|雨.*止め.*ぇぇぇ|雨.*のち.*晴レルヤ|雨.*上がっ.*た|雨.*弱まっ.*た|雨.*ござい.*まし|雨.*大丈夫.*だっ|雨.*やばかっ.*た|雨宿り.*し.*たら|明日.*わ.*雨|明日.*朝.*雨|明日.*雨.*だ|明日.*雨.*な|昨日.*な.*雨|夕方.*雨.*言う|夕方.*雨.*言っ|夕方.*雨.*降っ|明日.*雨.*かー|明日.*雨.*ふら|明日.*雨.*降る|昨日.*な.*雷雨|昨日.*昼.*雨雲|昨日.*雨.*だっ|昨日.*雨.*止ん|昨日.*雨.*風邪|明日.*雨.*っぽく|明日.*雨.*らしい|明日.*雨.*ーーーー|夕方.*以降.*雨|明日.*トカ.*雨|明日.*雨降.*ら|明日.*雷雨.*な|明日.*雷雨.*嫌|明日.*雷雨.*見|昨日.*雷雨.*時|明日.*雨中.*だっ|明日.*雨か.*なぁ～|明日.*がっつり.*雨|明日.*雨天中止.*なっ|昨日.*ゲリラ雷雨.*ひどかっ

図１０ NLP 法によるフィルタ例

3.1.2 近距離法

一般的に、ツイートについては、その内容についての「確からしさ」について、疑念が生じる。なぜなら、その正確性は、ツイートした本人しか分からないからである。そのため、NLP 法だけでは、その疑念の解消は難しい。

そこで、近距離法では、一定の時間で取得したツイート群から特定状況ワードでの抽出後に、ツイートに付加されている GPS データを利用して、各ツイート発生地点の距離を計算し、近いツイート同士のみを抽出するアプローチを検討した。その際には、「距離」だけではなく、「時間（同一時間帯）」、「複数ユーザー」によるツイートも、条件としてノイズリダクション機能に加えることとした。図１１に概念図を示す。つまり、このアプローチは、ツイート内の位置情報を利用して、３つの条件にてノイズリダクションする処理である（図１１内②）。具体的には、ある同じワードを含むツイート（図１１内①の処理後のツイート）が、ある時点において（図１１内ｂ））、同じ場所（図１１内ａ））から、複数ユーザー（図１１内ｃ））により投稿されていた場合、少なくとも、そこでは複数の人が同じことに関心を持っていたはずであるという考えにもとづき、ツイートを抽出する手法である。その結果、抽出された「状況」ツイートは、「確からしさへの疑念」の軽減（より信頼性の高いツイート）につながるだけではなく、ノイズ削減効果の高い抽出精度での可視化も期待できる。

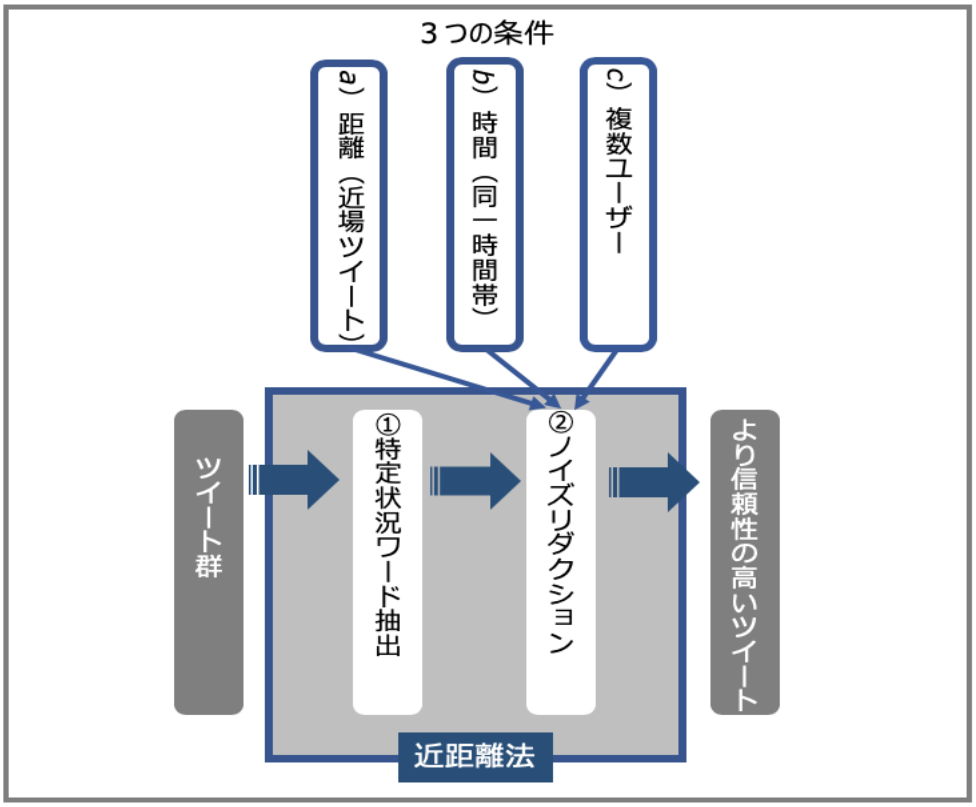


図１１　近距離法概念図

本実験では、具体的な条件として、（１）各ツイート発生地点間における距離が、3km 以内のツイート（GPS で使われる WGS84 を利用して、緯度経度からの各ツイート発生地点の距離を計算）、（２）同一時間帯（10 分間）におけるツイート、（３）複数 ID での投稿数が２件以

上のツイート（10 分以内の同一 ID の連続ツイートはボットとみなし除去）の条件に合致したツイートのみを抽出することとした。

なお、（１）の 3km としたのは、それより短い場合、取得できるツイート件数が少なくなる、また、それより長い場合、違う話題について拾ってしまう、つまり、誤抽出の可能性が高くなることが考えられるからである。（２）の同一時間帯を 10 分間としたのも、（１）の 3km と同様な理由である。（３）の２件以上のツイートとしたのは、ツイートの確からしさの観点から、少なくとも２人以上の投稿数は必要であるという理由からである。

3.1.3 組み合わせ法

NLP 法では、フィルタの作成にあたって、作成時のデータ量が少ない場合や偏りがある場合などに、抽出精度への影響が考えられる。

また、近距離法では、「状況」における振り返りや感想、降雨状態変化（「雨が止んだ」など）などのツイートに対する除去ができないケースが考えられる。例えば、平日の夜間帯:22:00 頃は、多くの人が、ニュース番組などを見た後に、感想をつぶやく可能性があり、この場合、誤抽出してしまう可能性がある。また、雨が止んだ際に、近場における複数ユーザーが、「雨が止んだ」というツイートを投稿する可能性が考えられ、その結果、そのツイートを拾ってしまうからである。

したがって、NLP 法と近距離法との組み合わせ法により、さらなる精度向上が期待できる。組み合わせにおける実行順序については、最初に、近距離法を適用してから、その後、NLP 法を適用した。

3.2 特定地域の状況可視化のためのノイズリダクション

本研究で開発･提案する２つ目の手法が、「特定地域の状況可視化のためのノイズリダクション」である。本手法は、特定地域の状況可視化を対象に、再現率を重視（多様な状況を捉えるために、より多くの正解データを抽出）する上で、事前のコストをかけずに、実行時に機械的に生成されるフィルタを生成･使用するアプローチである。実現には名詞の形態素 n-gram 頻度を適用している。これらに該当するツイートは、ノイズの可能性が高くフィルタの機械的生成の実現が容易であり、さらに n-gram のｎ数で正解ツイートの抽出（再現率）を制御できる可能性が考えられるからである。

再現率を重視したノイズリダクションでのフィルタ生成プロセスにおいて、本手法の位置づけを図１２に示す。従来手法では、最初に調査用ツイートから中身を確認しながらノイズツイートを識別･分類する。これは、最もコストがかかる人手の処理でもある（図１２内①）。次に識別･分類したノイズツイートを特徴づけるキーワード設定（単一･複数）や bot ID を抽出する（図１２内②）。この時、再現率を重視する上では、より多くのノイズツイートの識別と

ノイズツイートだけに該当する条件設定が重要となる。最後に実装ツールに合わせたパラメータ（フィルタ）を生成する（図１２内③）。それに対し、本手法では、可視化対象ツイートのみを用いて機械的に名詞の形態素 n-gram フィルタを生成する（図１２内②③）。

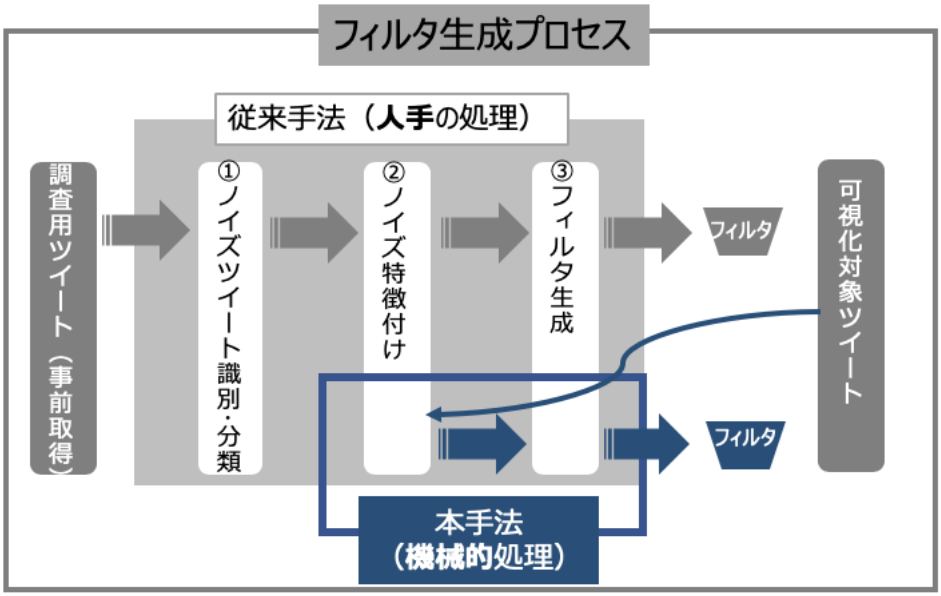


図１２ 再現率を重視したフィルタ生成における本提案手法の位置づけ

本手法は、具体的には２段階処理となる。第１段階では、日単位などある期間に特定地域から投稿されたツイートの抽出を行う。第２段階では、最初に抽出した可視化対象ツイートから名詞の形態素 n-gram（以降、“名詞の形態素”を省略し、3-gram などで表記）解析を行う。次に解析後の n-gram データから名詞の組合せ頻度が高いものだけ（閾値はパラメータで指定）（図１２内②）、フィルタとして生成する（図１２内③）。つまり、ある期間に名詞の組合せ頻度が高いものをノイズと見なした処理である。

本手法では、事前に Streaming API にて、すべての項目を取得した各ツイートから投稿者 ID、言語コード、時間、緯度･経度（その後、大字･町丁目に変換）を抽出した。そして、それをメタデータとし、本文と文字列連結して１つのツイートデータとした。これはまた、フィルタ生成時の入力データでもある。図１３に例を示す（投稿者 ID は伏せ字（＊）とした）。

```
******* ja 2016-04-14T22:44:47+09:00 熊本県熊本市中央区本山二丁目 お風呂入っていいかな
```

図１３ フィルタ生成時の入力データ形式例

本手法は、２つ目の段階が主であり、それは、大きく４つに分けられる。１つ目は、図１３の入力データ内に含まれるメタデータの除去と除去後のツイート本文内のストップワード（形態素解析の対象外ワード）の除去処理である。これは、形態素解析での精度向上のためには必

須の作業である。具体的には、メタデータの除去後にツイート本文内に HTML タグ、RT 文字、全角記号、改行文字が存在した場合、その文字を Null で置き換える。図１４に処理後の出力例を示す。

```
現在地 L:茨城県常総市上蛇町 #imacoconow [ImacocoNow! Android]
I'm at セーブオン常総きぬの里店 in Ibaraki-ken
Just posted a video @ きぬ医師会病院
ネット復旧してた
```

図１４ メタデータ･ストップワード除去後のデータ例

２つ目は、図１４の出力データを用いた n-gram 処理と、処理後の組合せ頻度を生成する処理である。本手法では、アプリ連携としての出現（I’m at ～）が既知の事実であったため、「at」については、事前に名詞として辞書登録を行った。図１５に出力例を示す。

```
1,[茨城県-I-m-at],6,[茨城県-I-m-at]
2,[in-常総市-茨城県-I],5,[in-常総市-茨城県-I]
3,[常総市-茨城県-I-m],5,[常総市-茨城県-I-m]
4,[店-in-常総市-茨城県],4,[店-in-常総市-茨城県]
```

図１５ 4-gram の組合せ頻度リスト例

３つ目は、図１５の出力データを用いたフィルタ生成処理である。あらかじめ指定した頻度閾値以上の組合せのみをフィルタとして生成する。前述のとおり、一定時間内に n-gram の組合せ頻度が高い場合、ノイズと見なす。図１６に出力例を示す。

```
茨城県.*I.*m.*at|in.*常総市.*茨城県.*I|常総市.*茨城県.*I.*m|店.*in.*常総市.*茨城県|Sta.*in.*常総市.*茨城県
```

図１６ 4-gram によるノイズフィルタ例

４つ目は、機械的に生成されたフィルタを用いた、図１３データを入力とするノイズリダクションである。入力データには、記号（ID 名）、数字（時間）、大字･町丁目レベルのメタデータが含まれるため、誤処理防止として、メタデータ以外の処理が必要となる。

第4章 評価実験

本章では、開発した２つの手法の評価実験方法について述べる。最初に、実験に使用したツールとデータセットについて説明する。そのつぎに、特定状況可視化のためのノイズリダクション手法の実験方法、特定地域可視化のためのノイズリダクション手法の実験方法について説明する。

4.1 ツール

4.1.1 逆ジオコーダ

位置情報データ（緯度：ex 34.785、経度：ex 135.529）から、大字･町丁目レベル位置参照情報（例えば、千葉県市川市市川南一丁目。以降、大字･町丁目レベルデータと表記）に変換するために、逆ジオコーダを構築した。その際に、国土交通省で提供している、位置参照情報ダウンロードサービスを利用した。逆ジオコーダ構築の手順･仕様･ポイントを以下に整理する。なお、逆ジオコーダの構築においては、ネット上の記事『簡易的な逆ジオコーディング』を参考にした。

- 位置参照情報ダウンロードサービスから、都道府県別に提供されている、大字･町丁目レベルのデータ（CSV 形式）をすべてダウンロードして、それらを一つのファイルに統合し、さらに、各ファイルにあるヘッダー部分を取り除き、最後に、utf-8 形式に変換して利用した。使用したのは、2013/07/31 バージョンである。
- Twitter から取得できる緯度（ex 34.78585089）、経度（ex 135.52950444）データは、最大小数点 8 桁であるが、国土交通省の位置情報で提供されている緯度･経度は、最大小数点 6 桁なので、桁数を小さくして利用する。
- 小数点 3 桁（ex 34.785）のレベルは、100m 単位であるが、4 桁の場合（10m）のレベルでは、位置情報にマッチしないことがあるため、3 桁までのレベル、つまり、34.785 の桁数まで削ることとする。
- 複数のマッチしたデータが戻される場合、もともとの位置情報と近いデータを取得するため、Twitter から取得した位置情報とマッチした複数のツイートにおいて、それぞれの緯度･経度情報との差の絶対値を求め、その合計値の一番小さい値を使用する。（構築したツールでは、最小値を return するようにした。また、海外からの位置情報については、None を return することとした。）
- この機能を実装するために、SQL が使えると便利なので、SQLite（本研究では、SQLite3）を使用する。これは、実行時にプロセスが起動する簡易的な RDB である。
- ツールは、python で構築する。

また、位置参照情報ダウンロードサービスから取得したCSVファイルのイメージを図１７に示す。

```
"01","北海道","01101","札幌市中央区","011010001001","旭ケ丘一丁目","43.042230","141.319722","0","3"
"01","北海道","01101","札幌市中央区","011010001002","旭ケ丘二丁目","43.039768","141.321733","0","3"
"01","北海道","01101","札幌市中央区","011010001003","旭ケ丘三丁目","43.039569","141.319617","0","3"
"01","北海道","01101","札幌市中央区","011010001004","旭ケ丘四丁目","43.038819","141.323040","0","3"
"01","北海道","01101","札幌市中央区","011010001005","旭ケ丘五丁目","43.036547","141.322217","0","3"
```

図１７ 大字・町丁目レベルデータのCSVファイル例

4.1.2 Google Maps

ある状況に関して抽出したツイートデータの位置情報から、Google Maps JavaScript API v3を使用して、Googleマップ上に複数のマーカーをプロットするツールを、以下の観点で作成した。そして、最終的に、本実験では使用しなかったが、時間経過の表現においては、色のグラデーションでの表現方法を用いて、複数の異なる色のマーカーを同時に扱いプロットするようにした。また、同時に複数の状況を表現するために、単なる複数の異なる色のマーカーに加え、状況を説明する文字も入れるようにした。

- ある状況について、複数の時間帯を同時に可視化できる（状況の変遷を捉える）
- ある時間帯における複数の状況を可視化できる
- 状況の強弱を可視化できる

また、地図の表示においては、日本列島全体が地図内に収まるように、Google マップのズームレベルを5で設定した。その際のHTMLのソースコード例を、図１８に示す。

```
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<title>Google Maps つぶやき Trend</title>
<script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false"></script>
<script type="text/javascript">
    function initialize() {
        var latlng = new google.maps.LatLng(35.630442,139.882951);
        var opts = {
            zoom: 5,
            center: latlng,
            mapTypeId: google.maps.MapTypeId.ROADMAP
            };
        var map = new google.maps.Map(document.getElementById("map_canvas"), opts);
var m_latlng1 = new google.maps.LatLng( 32.887509 , 130.104719 )
var marker1 = new google.maps.Marker({
 position: m_latlng1,
 animation: google.maps.Animation.DROP,
 map: map,
 icon: 'http://chart.apis.google.com/chart?chst=d_map_spin&chld=0.25||04122f|13||'
});
 (略)
```

図１８ Googleマップ用のHTMLコード例

4.1.3 形態素環境

形態素解析を行うために、MeCab 0.996 を使用し、辞書は、mecab-ipadic-neologd 2016-05-02 研究バージョンを利用した。こうすることで、ツイート解析において、最新の用語（名詞）を拾えるようにした。さらに、形態素 N-gram 実施の前提ライブラリとして、NTLK 2.0.4 を使用した。形態素 N-gram の処理を実施するうえでは、「MeCab と NLTK を使って最瀬語と共起関係を出力する」を参考にツールを構築した。この際に、形態素解析においては、名詞、形容詞、動詞、助動詞、副詞、形容動詞のみを扱った。

4.1.4 評価システム

構築した評価システム（図８参照）の用途は、２つに分けられる。このシステムは、特定状況可視化のためのノイズリダクション手法において、使用するものである。１つは、ノイズコーパスを作成する際に利用することである（図８①コーパス作成部分）。本研究では、前述のとおり、気象庁・アメダスが提供している毎時の降雨量データを利用し、「雨」で抽出したツイートと照合して、事実に反していた場合、そのツイートを蓄積し、それを利用してノイズコーパスを作成するといった使い方である。

もう１つは、本来の目的である実験の評価に利用することである。この場合は、ノイズリダクション機能を通して抽出したツイートについて、機械的に評価システムから正誤を検証するといった使い方である（図８ ④精度検証部分）。

4.1.5 構築したツールのまとめ

ここでは、本研究において構築･利用したツールについて、表に整理する。表１には、特定状況可視化のためのノイズリダクション手法用に構築したツールを示す。表２には、特定地域の状況可視化のためのノイズリダクション手法用に構築したツールを示す。表３には、データ取得用に構築したツールを示す。

表１ 特定状況可視化のためのノイズリダクションに構築･利用したツール一覧

No.	構築･利用したツール	説明
1	geoTest_99y.py	当ツールは、Mongo DB に格納した位置情報付きツイートを抽出するプログラムを生成する。当ツールは、取得したい日時情報を受け取り、それを反映した python のプログラムを生成する。生成されるプログラムは、exportTwitter.py。 （使用例）*python geoTest_99y.py 2014-06-01T00:00+9:00 2014-06-30T23:59+9:00*
2	exportTwitter.py	geoTest_99y.py にて生成されるプログラム。当プログラムは、Mongo DB に格納された位置情報付きツイートを抽出する。 （使用例）*python exportTwitter.py > 2014_6.data*

3	cat 2014_6.data ｜ ggrep -E "雨"	抽出したツイートファイルから、「雨」に関するツイートを抽出する UNIX コマンドの組み合わせ。 （使用例）*cat 2014_6.data	ggrep -E "雨" > 2014_6_rain.data*
4	returnNumber.py	抽出したツイートから 2/3 の数を計算するプログラム。 （使用例）*cat 2014_6_rain.data |wc -l | python returnNumber.py* *=> 32948 16474*
5	perl -MList::Util=shuffle -e 'print shuffle(<>)'	ノイズコーパス作成用のツイートデータをランダムに 2/3 だけ抽出するプログラム。 （使用例）*perl -MList::Util=shuffle -e 'print shuffle(<>)' < 2014_6_rain.data	tail -n 32948 > qqqqq*
6	comm -23	２つのファイルを業単位で比較する UNIX コマンド。これを用いて、残り 1/3 の検証用のツイートデータを抽出する。ただし、取り出す前に、ソートする必要あり。以下の例では、2014_6_rain.data と qqqqq とに存在しないレコードを抽出する。 （使用例） *sort 2014_6_rain.data > tmp111* *sort qqqqq > tmp222* *comm -23 tmp111 tmp222 > corpus_test_tmp.data*
7	export_3.py	逆ジオコーディング用プログラム。緯度経度から大字町丁目レベル（例：愛知県豊明市新栄町五丁目）に変換する。特定地域のツイートを抽出するには、grep コマンドと組み合わせる。当プログラムは、さらに、ReverseGeocoding.py を呼び出す。 （使用例） *cat qqqqq	sort -k3	python export_3.py	ggrep -E "東京都西多摩郡|東京都八王子市|東京都世田谷区|東京都千代田区|東京都江戸川区|東京都大田区|東京都青梅市|東京都練馬区|東京都府中市" > 2014_6_tokyo_rain.data* また、検証用のデータの抽出にも使用する。 （使用例） *cat corpus_test_tmp.data	sort -k3	python export_3.py	ggrep -E "東京都西多摩郡|東京都八王子市|東京都世田谷区|東京都千代田区|東京都江戸川区|東京都大田区|東京都青梅市|東京都練馬区|東京都府中市" > corpus_test.data*
8	ReverseGeocoding.py	SQLite 上に登録された位置情報から大字町丁目レベルを抽出する。 （使用例） *import ReverseGeocoding* *rb = ReverseGeocoding.ReverseGeocoding()* *print "******* " +* *rg.getAddress("34.78585089", "135.52950444")*
9	loadConvertionTable.py	位置情報データ（緯度経度、大字・町丁目レベルデータ）を SQLite 上の DB（変換テーブル）にロードする。 （使用例）*python loadConvertionTable.py*
10	makeNoiseCorpus_*xxxxx*_V2.sh	ノイズコーパスを生成する Script。この Script は、さらに、makeNoiseCorpus_V2.sh を呼び出す。 （使用例）東京地区のデータ（2014_6_tokyo_rain.data）を用いて作成する場合 *./makeNoiseCorpus_Tokyo_V2.sh*
11	makeNoiseCorpus_V2.sh	makeNoiseCorpus_*xxxxx*_V2.sh 内から実行される Script。当 Script は exportNoiseComment_V2.py を呼び出す。 （使用例） *./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data 東京都西多摩郡 西多摩郡% 2014_6_rain-noise_Tokyo.data* *cat 2014_6_rain-noise_Tokyo.data	cut -d" " -f3-20 > temp_2014_6_rain_noise_Tokyo.data*

12	exportNoiseComment_V2.py	「雨」で抽出した各ツイートについて、実際に降雨状況であったのか、評価システムに問い合わせるプログラム。当プログラムは、makeNoiseCorpus_V2.sh 内から実行される。 （使用例） *echo "$1 $2 $3"* *cat $1 | ggrep -E $2 | python exportNoiseComment_V2.py $3 n >> $4*
13	makeFilter_V3.sh 2	makeNoiseCorpus_*xxxxx*_V2.sh により作成されたノイズコーパスから、ノイズフィルタを生成する Script。 当 Script は、さらに、getTrend_co_freqz.py、generatCard_tri_zz.py を呼び出す。 （使用例） *cp temp_2014_6_rain_noise_Tokyo.data temp.data* *./makeFilter_V3.sh 2*
14	getTrend_co_freqz.py	形態素 n-gram の組を出力するプログラム。 当プログラムは、さらに、mecab_tokenizer.py を呼び出す。 （使用例） *python getTrend_co_freqz.py tri $1 n | cut -d" " -f2-4 | ggrep -E "^雨" | ggrep -E -v "[0-9]＊＊" | ggrep -E -v "＊[0-9]＊" | ggrep -E -v "＊＊[0-9]" | ggrep -E -v "[０-９]＊＊" | ggrep -E -v "＊[０-９]＊" | ggrep -E -v "＊＊[０-９]" | ggrep -v "(" | ggrep -v ")" | sort > zzz.data*
15	mecab_tokenizer.py	形態素解析後のワードのどのデータ（表層形、原形など）を使用するか指定する。当プログラムは、mecab_library.py を呼び出す。 （使用例） *import mecab_tokenizer* *tokens = mecab_tokenizer.tokenize(unicode(raw,'utf-8'))* *trigrams = nltk.trigrams(tokens)* *fd = nltk.FreqDist(trigrams)*
16	mecab_library.py	形態素解析後のワードを保存するライブラリ。 （使用例） *import mecab_library* *words = mecab_library.Tokens(sents[0])*
17	generatCard_tri_zz.py	フィルタ用としてのパラメータを含む grep コマンドを生成するプログラム。 （使用例）*cat zzz.data | python generatCard_tri_zz.py > ggg*
18	checkAlgorithm_V2_1_*x*.sh	作成したフィルタを用いて検証するプログラム。 当 Script は、さらに、rainProc_V2x.py、evaluateFilter_V2.py を呼び出す。 （使用例）corpus_test.data のデータにて検証する場合 *./checkAlgorithm_V2_1_1.sh 東京都 東京都*
19	rainProc_V2x.py	東京都の観測所のある地域に限定して、フィルタにて抽出されたツイートが、本当に正しかったのか、評価システムに問い合わせて確認するプログラム。 （使用例） *cat corpus_test.data | python rainProc_V2x.py | ggrep "◎" | wc | cut -d" " -f1-8 >> cnt;*
20	evaluateFilter_V2.py	適合率、再現率、F 尺度を算出するプログラム。 （使用例） *python evaluateFilter_V2.py $2*
21	makeDistanceEvaluationCard_1.py	"2014-06-01"などの形式で記載された日にちデータファイルを読み込み、10 分間隔で近距離法を実施するためのパラメータ群を作成するプログラム。 （使用例） *python makeDistanceEvaluationCard_1.py |* *python makeDistanceEvaluation_1.py corpus_test_tmp.data 東京都西多摩郡 > t_nishitama.sh*

22	makeDistanceEvaluation_1.py	近距離法を実施するための Script を生成するプログラム。 （使用例）同上
23	t_*xxxxxxx*.sh	makeDistanceEvaluationCard_1.py、makeDistanceEvaluation_1.py により生成された Script。当 Script は、getEvent_xx.sh、export_3.py を呼び出す。 （使用例）*./t_nishitama.sh > t_nishitama_1.data*
24	getEvent_xx.sh	当 Script は、t_*xxxxxxx*.sh により呼ばれる。 当 Script は、omitID.py、testGetDistance_3xx.py を呼び出す。 （使用例） *./getEvent_xx.sh corpus_test_tmp_4.data "2014-06-01T00:0" "雨" "2" "3000" "2" | python export_3.py | ggrep -E 東京都 西多摩郡*
25	omitID.py	当プログラムは、同一時間帯において、同一 ID での複数のツイートがあれば、その ID を除去するパラメータと共に、grep コマンドを生成する。 （使用例） *cat $1 | ggrep -E "$2" | ggrep -E "$3" | sort | uniq | cut -d' ' -f1 | uniq -c | sort | python ../omitID.py $4 > omitData.out;*
26	testGetDistance_3xx.py	当プログラムは、各ツイート発生地点間における距離がパラメータで指定した距離内に収まり、かつ、パラメータで指定した複数 ID によるツイート件数条件を満たす場合のみ抽出する。 （使用例） *cat $1 | ggrep -E "$2" | ggrep -E "$3" | eval `cat omitData.out` | python testGetDistance_3xx.py $5 $6 | sort -k2,3 | uniq | sort -k3,3;*
27	checkAlgorithm_V2_2_*x*.sh	当 Script は、近距離フィルタを検証する。さらに、rainProc_V2x.py、evaluateFilter_V2.py を呼び出す。 （使用例） *cat t_*_2.data > total_2.data* *./checkAlgorithm_V2_2_2.sh total_2.data 東京都*
28	checkAlgorithm_V2_3_*x*.sh	当 Script は、組み合わせ法を検証する。さらに、rainProc_V2x.py、evaluateFilter_V2.py を呼び出す。 （使用例） *./checkAlgorithm_V2_3_2.sh total_2.data 東京都*
29	loadMLData.py	評価システムにおける観測データを SQLite の各テーブルへロードするプログラム。内部では、insert 文を発行。 （使用例）*python loadMLData.py*
30	.import	評価システムにおける観測所の住所関連データを SQLite が提供している import 機能にてロード。 （使用例） *.separator ,* *.import 都県_観測所.csv pref_ml* *.import ame_master_f_utf8.csv ml_t_address*
31	generateMapzz_xx.py	ツイート内の緯度・経度情報から、Google マップで表示するためのコードを生成する。 （使用例） *./getEvent_xx.sh 2014_09_10_20.data "2014-09-10T16" "雨" "2" "3000" "2" | `eval cat ggg` | `eval cat qqq` | python generateMapzz_xx.py "3" > geotTemp.txt* *cat generateMapHeader.txt geotTemp.txt generateMapFooter.txt > test_filter_`date "+%Y-%m-%dT%H_%M"`_combi.html*

表２特定地域の状況可視化のためのノイズリダクションに構築・利用したツール一覧

No.	構築･利用したツール	説明
1	omitStopWord.py	ツイート本文内のストップワードを除去する（HTML タグ、RT が存在した場合、その文字を Null で置き換える）プログラム。 （使用例） *cat 熊本県_初日データ.txt | cut -d" " -f5- | python omitStopWord.py "Dummy"*
2	s <- Ngram()	全体として、ノイズリダクション用のフィルタの素データを生成する。Ngram()は、R パッケージの RMeCab で提供されている関数で、ファイルを読み込み、テキストの N-gram 処理を行う。type が 1 の場合、形態素原形を単位として N-gram をカウントする。なお N は引数 N で指定する。Pos にて、品詞の指定を行う。 （使用例） *s <- Ngram("熊本県_初日データ_step_1.txt", pos = c("名詞"), type = 1, N=3)*
	g <- s %>% group_by() %>% summarise() %>% arrange() %>% mutate()	group_by()、summarise()、arrange()、mutate()は、R パッケージの dplyr で提供されている関数。 group_by()は、変数 s に格納されているデータを列名：Ngram で分類し、summarise()は集計を行い、arrange()で並び替えを行い、mutate()で新たな列を追加する。 （使用例） *s %>% group_by(Ngram) %>% summarise("TF" = sum(Freq)) %>% arrange(desc(TF)) %>% mutate(Pos = factor(Ngram, levels = .$Ngram))*
	write.table(g)	R で提供されている関数。変数の中身をファイルに出力する。 （使用例） *g <- s %>% group_by(Ngram) %>% summarise("TF" = sum(Freq)) %>% arrange(desc(TF)) %>% mutate(Pos = factor(Ngram, levels = .$Ngram)) **write.table**(g, file="熊本県_初日データ_step_2_N3.txt",quote=F,sep = ",")*
	awk -F',' '{print $2","$3}'	awk はテキストの加工とパターン処理を行う UNIX のコマンド。テキストファイル内の各レコードを、「,」をセパレータとして区切り、２番目、３番目の文字列を抽出する。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print **$2**","**$3**}'*
	sed 's/¥[//g'	sed は主にテキストファイル内の文字の置き換えを行う UNIX のコマンド。ここでは、テキストファイル内の「[」文字を削除する。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print $2","$3}' | sed 's/¥[//g'*
	sed 's/¥]//g'	テキストファイル内の「]」文字を削除する。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print $2","$3}' | sed 's/¥[//g' | sed 's/¥]//g'*
	sed 's/-/.*/g'	テキストファイル内の「-」文字を「.*」に置き換える。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print $2","$3}' | sed 's/¥[//g' | sed 's/¥]//g' | sed 's/-/.*/g'*

	awk -F',' '{ print $2″ ″$1}'	出力の順番を変える処理。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print $2","$3}' | sed 's/¥[//g' | sed 's/¥]//g' | sed 's/-/.*/g' | awk -F',' '{ print **$2″ ″$1**}'*
	awk '$1 >= 5{print}'	指定した数（頻度）以上の文字列を抽出する処理。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print $2","$3}' | sed 's/¥[//g' | sed 's/¥]//g' | sed 's/-/.*/g' | awk -F',' '{ print $2″ ″$1}' | awk '$1 >= **5**{print}'*
3	omitCard_2.py	ノイズリダクション用フィルタ（awk のマッチパターン生成）を生成する。 （使用例） *tail -n +2 熊本県_初日データ_step_2_N3.txt | grep -i -v -E "¥(¥@"| awk -F',' '{print $2","$3}' | sed 's/¥[//g' | sed 's/¥]//g' | sed 's/-/.*/g' | awk -F',' '{ print $2″ ″$1}' | awk '$1 >= 5{print}' | python **omitCard_2.py***
4	run.sh	omitCard_2.py により生成されたフィルタを使用して、特定地域で抽出したツイートのノイズリダクションを行う。 （使用例） *cat 熊本県_初日データ_w_正解.txt | ./run.sh*
5	RMeCabFreq( )	全体として、抽出したツイート群から、ワードクラウドを生成・表示。RMeCabFreq( )は、R パッケージの RMeCab で提供されている関数で、指定されたテキストファイルを形態素解析して、その活用形を原形に変換した上で、その頻度を数えて、結果をデータフレームとして返す関数。 （使用例） *word <- RMeCabFreq("before.data")*
	word[(word[, 2] == ″名詞″ & word[, 3] != ″非自立″) & word$Freq > 1,]	名詞かつ非自立のもの、そして、頻度指定より大きなものを抽出する。 （使用例） *res <- word[(word[, 2] == "名詞" & word[, 3] != "非自立") & word$Freq > 1,]*
	order(res[4]$Freq,decreasing=F)	R のデータフレームを昇順にソートする。 （使用例） *sortlist <- order(res[4]$Freq,decreasing=F)*
	res5 <-res[sortlist,]	形態素を頻度順にソートする。 （使用例） *res5 <-res[sortlist,]*
	d3wordcloud()	ワードクラウドで表示する。 *d3wordcloud(res5[,1],res5[,4],rotate.min = 0, rotate.max = 0, padding = 1)*

表３データ取得用に構築・利用したツール一覧

1	test4yy.py	Twitter の Streaming API を使用して、日本のエリアを指定して、ツイートを抽出し Mongo DB へ保存する。このプログラムは、60 秒処理して、終了する。
2	get_message_prod.sh	このツールは、test4yy.py を繰り返し実行する。test4yy.py は、60 秒間処理して終了する。終了後、このツールは、30 秒間 Wait（Sleep）してから、再度、test4yy.py を実行する。
3	startup_prod.sh	サーバの起動時に、Mongo DB を起動する。また、システム状況を管理する system_manage.py を定期的（１時間毎）に実行するように登録する。
4	system_manage.py	ツイートの取得状況をメールで送信する。件数が伸びていなければ、障害が発生していると判断できる。

4.2 データセット

本節では、研究全体で分析に利用した位置情報付きツイート数について整理する。まず、予備調査では、2014 年 6 月 11 日から 2014 年 6 月 25 日の間に取得した 4,122,468 件のツイートを利用した。そして、特定状況可視化のためのノイズリダクションの実験では、2014 年 6 月 11 日から 2014 年 6 月 30 日の間に取得した 49,422 件、2014 年 9 月 10 日 16 時から 17 時の間に取得した 383 件、2014 年 10 月 6 日 10 時から 11 時の間に取得した 384 件、2014 年 11 月 1 日 12 時から 13 時の間に取得した 384 件、2014 年 10 月 31 日 12 時から 13 時の間に取得した 144 件のツイートを利用した。最後に、特定地域の状況可視化のためのノイズリダクションの実験では、2016 年 4 月 14 日 21 時から 2016 年 4 月 18 日 24 時の間に取得した 9,127 件、2015 年 9 月 10 日 12 時 50 分から 2015 年 9 月 14 日 24 時の間に取得した 111 件のツイートを利用した。各実験用データセットの詳細については、次の項で説明する。

4.2.1 特定状況可視化のためのノイズリダクション用

本手法の実験で使用したデータは、取得システムを構築して、Twitter から Streaming API（緯度・経度による日本列島を範囲指定：

locations=[123.24,146.2,39.0, 138.4,33.5,146.1,46.20]）で取得したもの（位置情報付き）を利用した。

ノイズコーパス作成、および、各手法における多重比較検定においては、2014 年６月のツイートデータ：5,317,659 件から、東京都の「雨」で抽出した 49,422 件を使用した。また、各手法についての降雨タイプ別検証においては、2014/9/10 16:00-17:00 における日本列島を範囲指定して取得した全ツイートデータから「雨」で抽出した 383 件、同様に、2014/10/6 10:00-11:00：384 件、2014/11/01 12:00-13:00：384 件、2014/10/31 12:00-13:00：144 件を抽出し使用した。

4.2.2 特定地域の状況可視化のためのノイズリダクション用

本手法の実験で使用したデータは、「特定状況可視化のためのノイズリダクション用」と同様な方法で取得したものを利用した。本手法では、２セットの実験を計画したが、１セット目には、2016 年 4 月 14 日の 21 時から 2016 年 4 月 18 日の 24 時までの熊本県から投稿されたデータの 9,127 件を用いた。２セット目は、2015 年 9 月 10 日の 12 時 50 分から 2015 年 9 月 14 日の 24 時までの茨城県常総市および茨城県猿島郡境町から投稿された 111 件のデータを用いた。

4.3 実験方法

4.3.1 特定状況可視化のためのノイズリダクション

本手法における実験では、2014 年６月の雨に関するツイートデータから、東京都の降雨量データを抽出して行った。実験の方針として、極力、フィッシャーの実験計画法に沿うようにした。具体的には、使用するデータを、ランダムにフィルタ生成用と検証用に分けるようにした。また、偶然のばらつき（誤差）による影響を取り除くために、同一データによる手法ごとの実験を、計１０回実施した。

NLP 法でフィルタ生成用に使用するデータとして、2014 年６月の「雨」を含むツイート（49,422 件）から、ランダムに 2/3（32,948 件）を取り出して、残りの 1/3（16,474 件）を手法の検証用データとして用いた。

さらに、ノイズリダクション機能の客観的な評価の側面から、前述のとおり、観測所がある地域（東京都西多摩郡、東京都八王子市、東京都世田谷区、東京都千代田区、東京都江戸川区、東京都大田区、東京都青梅市、東京都練馬区、東京都府中市）のデータを利用した。そのために、抽出したデータは、逆ジオコーダを用いて、位置情報から大字･町丁目レベルに変換し、その後、観測所のある９つの地域のみを抽出･利用した。

観測所がある地域にて抽出されるデータは、毎回、フィルタ生成用には約 1,700 件、検証用には約 900 件となった。近距離法、組み合わせ法においても（フィルタ無しの場合も）、この検証用データを用いた検証を実施した。

また、機能の評価においては、適合率（抽出したツイートの正しさ）を使用した。一般的に、検索やフィルタ性能の指標として、多少取りこぼしがあっても正しいものだけを抽出する指標（適合率）、多少間違いが混じっていても、取りこぼしがなく、できるだけ正しいものを抽出する指標（再現率）とのバランスを取ったＦ尺度（適合率と再現率との調和平均）が用いられることが多い。式は以下のようになる。

$$F\text{尺度} = （2 \times \text{適合率} \times \text{再現率}）／（\text{適合率} + \text{再現率}）$$

しかし、本評価では、何かしらの事象が生じている「状況」においては、ツイート数が多いはずであるという想定（ただし人口密度が低い場合はその限りでは無い）と、可視化する際にノイズが混じっていると分かりづらくなる、という点から、評価指標には、適合率を重視した。

また、従来手法との比較においては、従来手法の自然言語処理を一般化するのが難しいため（適合率は、フィルタのノイズ処理用ワード（状況ワード、bot id など）の数･網羅性に依存

するが、この設定は状況によって異なるため）、本手法では、自然言語処理法を拡張した NLP 法との比較を行うこととした。

その結果、評価としては、①フィルタ無し、②従来手法（NLP 法）、③近距離法、④組み合わせ法を適用した実験結果の適合率に有意差があるか、４群による多重比較検定を用いることとした。ただし、本実験は、４群全体での有意差を捉えるのが目的ではなく、各群（各手法）間における有意差の確認を目的とするため、分散分析は用いず、初めから、多重比較検定を実施することとした。

検定において、同一データにて各手法の繰り返し実験を行うため、「対応あり」の多重比較検定を実施した。その際に、片側検定を採用することとした。ノイズリダクション機能を適用することで、適合率においてはプラスの効果しか表れない、というのがその理由である。

また、実験数が１０と小さく、各群の母集団の正規性が認められにくいという点から、ノンパラメトリック検定にて実施した。そして、その中から、対応のある多重比較検定として、ライアン法による補正を伴うウィルコクソンの符号付順位検定を用いて実施した。

さらに、各手法について、別のデータを用いての検証も実施した。具体的には、全国の「雨」で抽出したデータを用いて、１．2014 年 9 月 10 日に発生した東京地区のゲリラ豪雨（70～100 ミリ/時間）、２．2014 年 10 月 6 日の台風 18 号に伴う関東地方の強い雨（20～30 ミリ/時間）、３．2014 年 11 月 1 日の関東地区の普通の雨（3～10 ミリ/時間）、４．2014 年 10 月 31 日の関東地区に降雨状況がほぼ無い時間帯、の４つの降雨タイプに適用し、評価を行った。

4.3.2 特定地域の状況可視化のためのノイズリダクション

本手法における実験では、地震や集中豪雨により被災した地域から投稿されたツイートデータを処理して評価を行った。

本実験では、２セットを計画した。１回目は地震で被災した広域（県レベル）、2 回目は集中豪雨で被災した狭域（市･町レベル）での実験である。広域･狭域と分けたのは、ツイート数の影響（広域の場合、狭域に比べてよりツイート数が多いと想定）を検証するためである。

各実験では、被災直後から 1 日単位で計 5 日分実施した。さらに、実験ごとに、3-gram、4-gram の検証を行った。その際の n-gram の組合せ頻度の閾値（閾値 5 の場合、頻度数が 5 以上となる組合せを用いてフィルタを生成）は、ツイート量を考慮して変動させた。ツイート数が極端に少ない場合、閾値を一律にするとフィルタが生成されない可能性が考えられるからである。

さらに、従来手法との比較には、本手法と同程度の精度を得るために、最もツイート量の多い 1 回目･3 日目の 3,088 件を用いて、「3.2 特定地域の状況可視化のためのノイズリダクショ

ン」で示した従来手法（今回は、キーワード設定（複数）、bot ID 抽出にて実施）にて、フィルタを生成した。そして、そのフィルタを 1 回目･5 日目の 1,522 件に適用した。

評価においては、適合率、再現率、F 尺度、および、適合率向上を用いた。なお、適合率向上は、フィルタ適用前後での適合率向上の倍率とした。

正解情報は、前述した正解判断基準、ノイズ定義（1.3.2 参照）に従い、フィルタ未使用時に抽出した全ツイートに対し手作業で付与した。また、前述のとおり（1.3.2 参照）、本手法での可視化対象は、人から投稿された被災地現場の避難･生活状況である。

第5章 結果

5.1 特定状況可視化のためのノイズリダクション手法の評価実験

本節では、特定状況可視化のためのノイズリダクション手法の評価実験の結果について、整理する。

5.1.1 各手法における適合率とF尺度

本実験では、４つの手法について実施した。実施したのは、①未フィルタ、②従来手法（NLP法）、③近距離法、④組み合わせ法である。実験は、計１０回実施した。その結果を、図１９に示す。

当図には、各手法における適合率（①のみ正答率）、および、F尺度（図中に下線にて表示）について整理した。棒グラフは適合率、折れ線グラフはF尺度を表している。

各回数の結果において、左から１番目は①未フィルタ、左から２番目は②従来手法(NLP法)、左から３番目は③近距離法、最後は④組み合わせ法を示している。たとえば、１回目の結果においては、①未フィルタでは適合率（正当率）が0.47、②NLP法では適合率が0.52、F尺度が0.62、③近距離法では適合率が0.76、F尺度が0.49、④組み合わせ法では適合率が0.82、F尺度が0.43を示している。

全体傾向としては、各実験において、ほぼ同様な結果となった。具体的には、各実験において、④組み合わせ法での適合率が一番高く、逆に、F尺度について、一番低い値となっている。

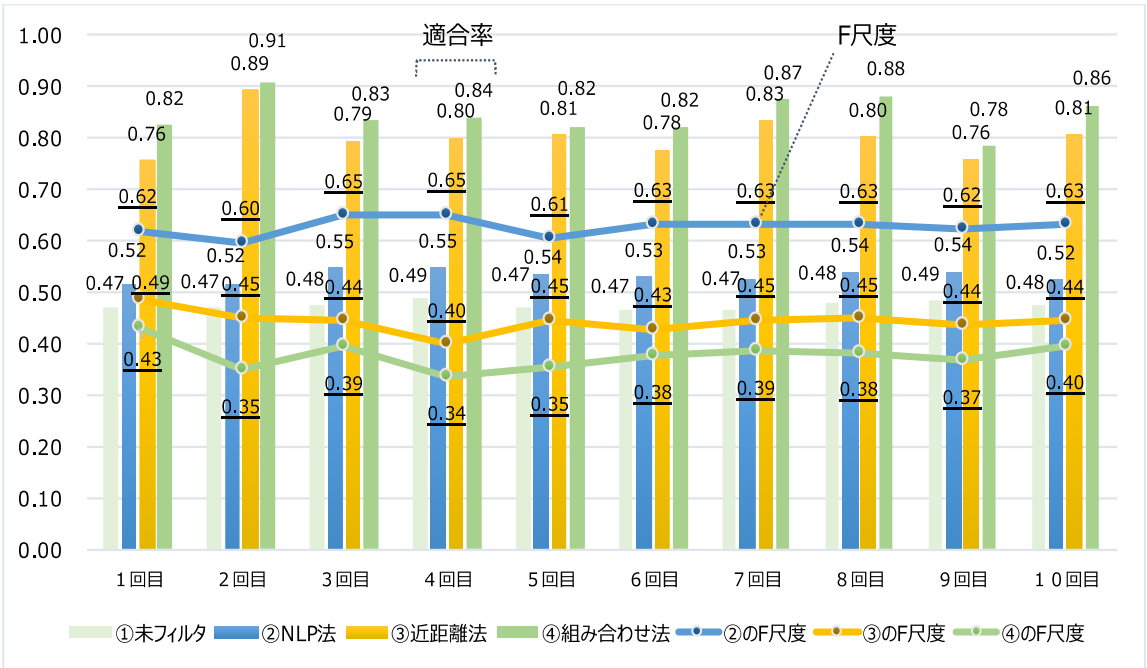


図１９ 特定状況可視化のためのノイズリダクションにおける手法別適合率とF尺度

5.1.2 多重比較検定

図２０には、箱ひげ図を用いて、各手法間における多重比較の検定結果を示した。本検定では、名義的有意水準を用いるにあたり、片側有意確率（P 値）は 0.0025 となった。その結果、①未フィルタと②NLP 法との間では片側検定として、**p<0.01（名義上の有意水準：0.005）が認められた。②NLP 法と③近距離法との間にも片側検定として、**p<0.01（名義上の有意水準：0.005）が認められた。さらに、③近距離法と④組み合わせ法との間にも片側検定として、**p<0.01（名義上の有意水準：0.005）が認められた。ただし、②NLP 法と④組み合わせ法との間においては、片側検定として、*p<0.05（名義上の有意水準：0.0125）が認められた。

また、当図から、適合率において、全体において、①未フィルタを除いて、②NLP 法と④組み合わせ法との差が最も開いていた。

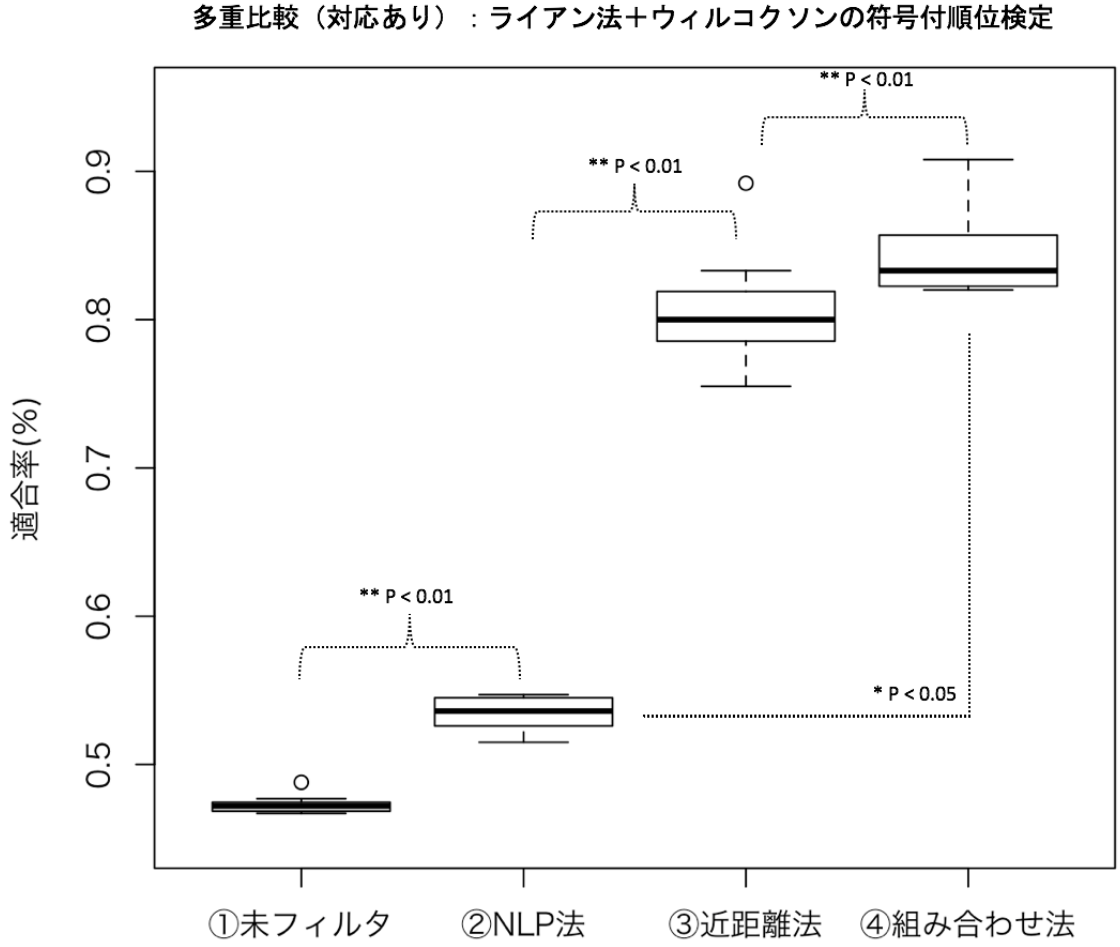


図２０ 各手法間における多重比較検定結果

5.1.3 降雨タイプ別検証

つぎに、構築した手法を用いて、前述した降雨タイプ別について検証した結果を、表４に示す。表には、抽出したツイート数、ノイズ数、未フィルタ時の正答率、各手法における抽出件数、誤抽出件数、適合率、F 尺度の結果について整理した。

検証において抽出されたツイートのノイズ（誤抽出）の判断基準は、各ツイートに、予報などのニュース（例：「最高 22℃ 最低 10℃ 降水確率 50%」）、現時点の降雨状況ではないことを表現しているもの（例：「雨降ってない」、「止んだ」、「明日は雨かな」、「今朝の雨は酷かった」）、降雨状況と関係ないもの（例：「スープ春雨」）、現時点で降雨状況の判断ができないもの（例：上原店雨の日スペシャル）のどれか１つでも含まれているものとした。

図２１には、表３の「１．東京のゲリラ豪雨」タイプに対応した、日本気象協会の Web サイトで公開されている 2014 年 9 月 10 日 17 時における降水量のマップを示す。なお、東京のゲリラ豪雨タイプと称しているが、この時間帯には、他地域でも降雨量が観測されている。

図２２には、図２０で示した「状況」をツイートから可視化するために、同日の 16 時台の位置情報を含むツイートから、組み合わせ法にて「雨」のワード、かつ、各ツイート発生地点間の距離が 3km で抽出し、そのデータを Google Maps API を使用して Google マップ上にプロットした結果を示す。

表４ 降雨タイプ別検証結果

抽出期間		1.東京のゲリラ豪雨（70～100ミリ/時間） 2014/9/10 16:00-17:00		2.台風18号による関東地方における強い雨（20～30ミリ/時間） 2014/10/6 10:00-11:00		3.関東地区の普通の雨（3～10ミリ/時間） 2014/11/01 12:00-13:00		4.関東に降雨状況がほぼ無い時間帯 2014/10/31 12:00-13:00	
		各ツイート発生地点間における距離：*3km*	各ツイート発生地点間における距離：*5km*	各ツイート発生地点間における距離：*3km*	各ツイート発生地点間における距離：*5km*	各ツイート発生地点間における距離：*3km*	各ツイート発生地点間における距離：*5km*	各ツイート発生地点間における距離：*3km*	各ツイート発生地点間における距離：*5km*
抽出したツイート数		383		384		384		144	
ノイズ数		77		119		107		131	
未フィルタ時の正答率		0.80		0.69		0.72		0.09	
*nlp*法	抽出件数	292		282		277		27	
	誤抽出	26		42		24		17	
	適合率	0.91		0.85		0.91		0.37	
	F尺度	0.78		0.71		0.72		0.12	
近距離法	抽出件数	241	280	70	127	56	87	0	4
	誤抽出	22	32	10	28	3	7	0	4
	適合率	0.91	0.89	0.86	0.78	0.95	0.92	-	0.00
	F尺度	0.70	0.76	0.29	0.45	0.24	0.34	-	-
組み合わせ法	抽出件数	198	228	60	103	53	79	0	3
	誤抽出	7	12	3	13	1	3	0	3
	適合率	0.96	0.95	0.95	0.87	0.98	0.96	-	0.00
	F尺度	0.63	0.68	0.25	0.39	0.23	0.32	-	-

図２１　2014 年 9 月 10 日 17 時の降雨状況

図２２　組み合わせ法による抽出（2014/09/10 16:00-17:00）

5.2 特定地域の状況可視化のためのノイズリダクション手法の評価実験

本節では、特定地域の状況可視化のためのノイズリダクション手法の評価実験の結果について、整理する。

5.2.1 条件別フィルタにおける再現率とＦ尺度

実験の結果を、表５～８に示す。表５、６は、熊本県のツイートを用いた結果である。表７、８は、茨城県常総市･境町の結果である。表内 N は n-gram の n 数（N=3 は、形態素 3-gram）を表す。表５、７でのフィルタ生成条件は 3-gram と頻度閾値（以降、閾値と表記）5、4-gram と閾値 5、また、表６、８では 3-gram と閾値 3、4-gram と閾値 3 である。表６の４日目の「-」は、未処理の意味である（生成されたフィルタにツールで処理できない文字が含まれていた）。

表５ 熊本県での実験結果（頻度閾値５）

	フィルタなし			フィルタ適用(N=3，頻度閾値5)						フィルタ適用(N=4，頻度閾値5)					
地域	抽出件数	正解数	適合率	抽出件数	正解数	適合率	再現率	適合率向上	F尺度	抽出件数	正解数	適合率	再現率	適合率向上	F尺度
熊本県_初日	814	40	0.05	91	39	0.43	0.98	8.72	0.60	99	40	0.40	1.00	8.22	0.58
熊本県_2日目	1915	89	0.05	196	76	0.39	0.85	8.34	0.53	276	87	0.32	0.98	6.78	0.48
熊本県_3日目	3088	140	0.05	246	125	0.51	0.89	11.21	0.65	322	134	0.42	0.96	9.18	0.58
熊本県_4日目	1758	135	0.08	199	111	0.56	0.82	7.26	0.66	308	134	0.44	0.99	5.67	0.60
熊本県_5日目	1552	98	0.06	179	81	0.45	0.83	7.17	0.58	250	93	0.37	0.95	5.89	0.53

表６ 熊本県での実験結果（頻度閾値３）

	フィルタなし			フィルタ適用(N=3，頻度閾値3)						フィルタ適用(N=4，頻度閾値3)					
地域	抽出件数	正解数	適合率	抽出件数	正解数	適合率	再現率	適合率向上	F尺度	抽出件数	正解数	適合率	再現率	適合率向上	F尺度
熊本県_初日	814	40	0.05	83	36	0.43	0.90	8.83	0.59	92	38	0.41	0.95	8.41	0.58
熊本県_2日目	1915	89	0.05	186	76	0.41	0.85	8.79	0.55	253	85	0.34	0.96	7.23	0.50
熊本県_3日目	3088	140	0.05	224	113	0.50	0.81	11.13	0.62	302	134	0.44	0.96	9.79	0.61
熊本県_4日目	1758	135	0.08	-	-	-	-	-	-	-	-	-	-	-	-
熊本県_5日目	1552	98	0.06	170	76	0.45	0.78	7.08	0.57	237	89	0.38	0.91	5.95	0.53

表７　茨城県常総市･境町での実験結果（頻度閾値５）

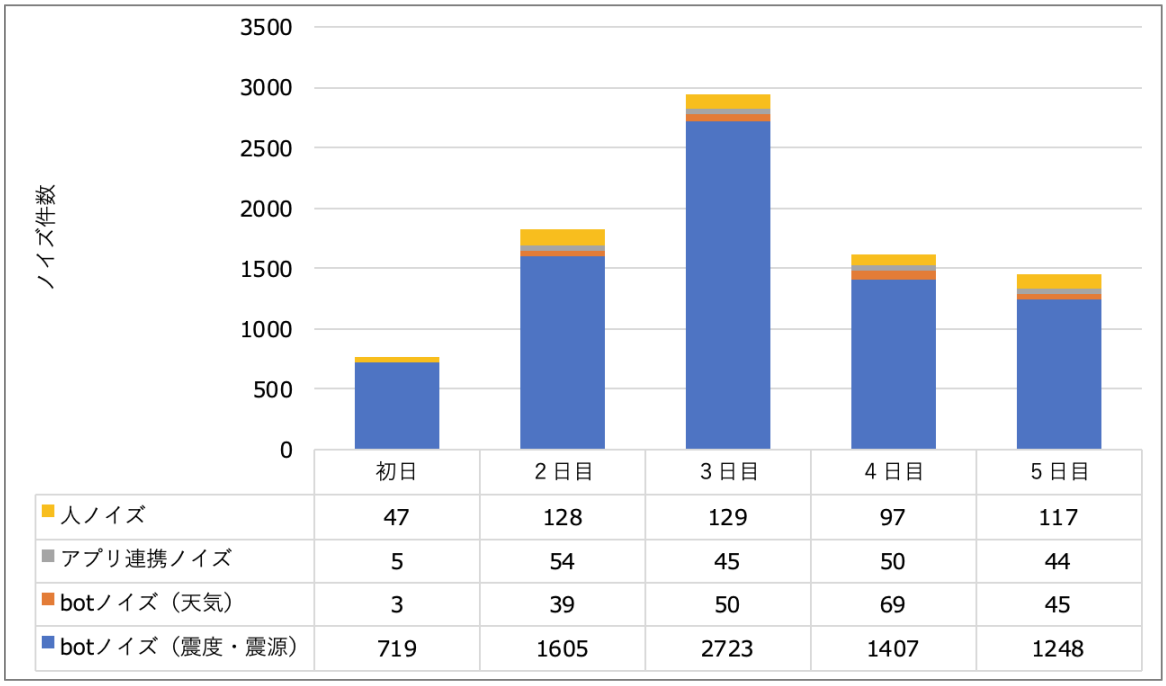
	フィルタなし			フィルタ適用(N=3，頻度閾値5)						フィルタ適用(N=4，頻度閾値5)					
地域	抽出件数	正解数	適合率	抽出件数	正解数	適合率	再現率	適合率向上	F尺度	抽出件数	正解数	適合率	再現率	適合率向上	F尺度
常総市_境町_初日	10	3	0.30	10	3	0.30	1.00	1.00	0.46	10	3	0.30	1.00	1.00	0.46
常総市_境町_2日目	31	11	0.35	18	9	0.50	0.82	1.41	0.62	31	11	0.35	1.00	1.00	0.52
常総市_境町_3日目	32	3	0.09	11	3	0.27	1.00	2.91	0.43	32	3	0.09	1.00	1.00	0.17
常総市_境町_4日目	19	6	0.32	11	5	0.45	0.83	1.44	0.59	19	6	0.32	1.00	1.00	0.48
常総市_境町_5日目	19	10	0.53	11	8	0.73	0.80	1.38	0.76	19	10	0.53	1.00	1.00	0.69
常総市_境町_5日分まとめて	111	33	0.30	60	27	0.45	0.82	1.51	0.58	94	30	0.32	0.91	1.07	0.47

表８　茨城県常総市･境町での実験結果（頻度閾値３）

	フィルタなし			フィルタ適用(N=3，頻度閾値3)						フィルタ適用(N=4，頻度閾値3)					
地域	抽出件数	正解数	適合率	抽出件数	正解数	適合率	再現率	適合率向上	F尺度	抽出件数	正解数	適合率	再現率	適合率向上	F尺度
常総市_境町_初日	10	3	0.30	10	3	0.30	1.00	1.00	0.46	10	3	0.30	1.00	1.00	0.46
常総市_境町_2日目	31	11	0.35	18	9	0.50	0.82	1.41	0.62	24	10	0.42	0.91	1.17	0.57
常総市_境町_3日目	32	3	0.09	10	2	0.20	0.67	2.13	0.31	29	3	0.10	1.00	1.10	0.19
常総市_境町_4日目	19	6	0.32	11	5	0.45	0.83	1.44	0.59	17	6	0.35	1.00	1.12	0.52
常総市_境町_5日目	19	10	0.53	11	8	0.73	0.80	1.38	0.76	19	10	0.53	1.00	1.00	0.69
常総市_境町_5日分まとめて	111	33	0.30	58	25	0.43	0.76	1.45	0.55	92	30	0.33	0.91	1.10	0.48

表７、８で網掛けされた初日データでは、２つの条件（3-gram･閾値 5，4-gram･閾値 5）のいずれにおいてフィルタが未生成であった。また、表７の４日目･4-gram、表７、８の５日目･4-gram でも、フィルタが未生成であった。表７の２日目、３日目は、4-gram･閾値５の条件にてフィルタが生成されたが、その効果は得られていない。図１５には表７の２日目の 4-gram で生成されたフィルタの基となるリストの抜粋を示す。見方として「[ ],数字,[ ]」における「[ ]」の部分が n-gram の部分である（最後の「[ ]」は使用しない）。数字の部分は頻度数を示す。例として「[茨城県-I-m-at],6,[茨城県-I-m-at]」の場合は「茨城県，I，m，at」から成る 4-gram の形態素であり頻度数は６となる。そのため、図１５から表７の２日目 4-gram･閾値５の条件では、３組の形態素の組合せからフィルタ生成が行われていた。

そして、熊本県から投稿されたツイートを用いて分析を行った結果、ノイズとして判定されたツイートは、大きく４つに分類された（図２３参照）。１つ目は、震度･震源に関する bot ツイートである。これは、さらに「【M」、「 M」、「震度」、「震源」、「規模」、「ND」、「速報 L」を含むツイートに分類された。２つ目は、「℃」を含む天気に関する bot ツイートである。３つ目はアプリ連携のものであり、「m at」、「Just posted」、「物件」を含むツイートに分類された。「m at」は Swarm/Foursquare との連携、「Just posted」は Instagram との連携、「物件」は大島てる（事故物件の情報提供ウェブサイト）との連携によるものである。４つ目は、人によるツイートである。



	初日	2 日目	3 日目	4 日目	5 日目
人ノイズ	47	128	129	97	117
アプリ連携ノイズ	5	54	45	50	44
botノイズ（天気）	3	39	50	69	45
botノイズ（震度・震源）	719	1605	2723	1407	1248

図２３ 日別によるノイズ種別の割合

表９には、従来手法との比較を示す。従来手法では、再現率を重視する上で、本手法と同程度の精度を得るために、事前作業時間（事前コスト）として 103 分を要した。その内訳は 3,088 件のノイズツイートの識別･分類に 93 分（100 件あたり 3 分）、ノイズ特徴づけ･フィルタ生成に 10 分、である。

表９ 従来手法との比較

		フィルタなし			フィルタ適用						
手法		抽出件数	正解数	適合率	事前作業時間（分）	抽出件数	正解数	適合率	再現率	適合率向上	F尺度
従来手法		1552	98	0.06	103	247	98	0.40	1.00	6.28	0.57
本手法	N=3，頻度閾値5				-	179	81	0.45	0.83	7.17	0.58
	N=4，頻度閾値5				-	250	93	0.37	0.95	5.89	0.53
	N=3，頻度閾値3				-	170	76	0.45	0.78	7.08	0.57
	N=4，頻度閾値3				-	237	89	0.38	0.91	5.95	0.53

5.2.2 フィルタ適用前後の被災地状況のワードクラウド例

参考までに、図２４に、表５の３日目のフィルタなし（ただし、HTML タグ、RT 文字、全角記号、改行文字は事前に除去）の、名詞の単語（非自立除く）によるワードクラウドを示す。さらに、図２５には、表５の３日目の 4-gram・閾値５にて抽出した結果のワードクラウドを示す。両図とも、名詞の頻度２以上を出力した。

図２４　フィルタ適用前のワードクラウド例

図２５　フィルタ適用後のワードクラウド例

第6章 考察

6.1 特定状況可視化のためのノイズリダクション手法

本節では、本手法の実験で実施した４つの手法について、その結果より、それぞれの手法について考察する。まず、東京都の降雨量データを用いた実験において、未フィルタの場合、適合率（正当率）が、0.47～0.49 あたりとなった。検証用のツイート数は、毎回 900 件程度使用していたため（4.3.1 参照）、毎回半数以上（460～480 件程度）のノイズが含まれていた。

NLP 法は、前述の説明（3.1.1 参照）に加え、事前に、抽出したい「状況」についてのノイズコーパスを整備しておく必要があり、ノイズコーパスが充実していない場合には、高い精度（適合率）での抽出が期待できない手法でもある。東京都の降雨量データを用いた実験において、図１９、２０から、適合率は 53%あたり、未フィルタと比べた適合率については、有意差（0.5%の名義的有意水準）が認められた。また、降雨タイプ別検証実験において、NLP 法は、表４からは、「4. 関東に降雨状況がほぼ無い時間帯」の降雨タイプを除いて、85%以上の高い適合率が得られた。さらに、F 尺度においては、0.7～0.8 であることが得られた。一方で、「4. 関東に降雨状況がほぼ無い時間帯」においては（関東以外の一部の地域では降雨状況が存在）、適合率が 37%となり、フィルタ効果としては高い精度とはいえない結果であった。そのため、ノイズリダクションから漏れたツイートを確認したところ、「今日雨降るん」、「今日雨やとよね」、「雨ダイジョブそぉかな～」、「スープ春雨」、「雨やんだ」などが得られた。また、降雨タイプ別検証実験においては、2014 年６月の東京地区のツイートデータを用いて作成したフィルタを用いているが、このフィルタにおいては、予想、「雨やとよね」という方言、「春雨」などの雨を含む名詞、「やんだ」というひらがな表現などには、未対応であった。したがって、このことが、適合率 37%となった要因の１つとして考えられる。

近距離法は、前述の説明（3.1.2 参照）に加え、様々な「状況」の可視化において、事前にノイズコーパスの準備が不要になるという利点がある。一方、前述のとおり（3.1.3 参照）、「状況」における振り返りや感想、状況変化のツイートまで抽出してしまうといった課題が残る。しかしながら、そのような課題を抱えながらも、東京都の降雨量データを用いた実験において、図１９、２０から、適合率は 80%あたりの結果が得られ、NLP 法と比べた適合率について、有意差（0.5%の名義的有意水準）が認められた。また、降雨タイプ別検証実験において、表４から「4. 関東に降雨状況がほぼ無い時間帯」の降雨タイプを除いて、90%前後の高い適合率が得られた（各ツイート発生地点間における距離を 3km 以内とした場合）。したがって、「状況」に関するツイート抽出数が多く見込める場合、ノイズがより多く混入する可能性があり（東京都の降雨量データを用いた実験では、毎回、約 900 件の検証データの内、５割程度がノイズ。降雨タイプ別検証実験では、「関東に降雨状況がほぼ無い時間帯」を除いて、他の３つのタイ

プでは、各検証データの約 380 件の内、２～３割程度がノイズ）、その場合、近距離法は、より精度の高い抽出ができることが示唆された。ただし、ツイート数が少ないケースでは、近距離法を適用すると、「複数人の投稿が必要」という条件に該当し、ノイズではないツイートまでもが除去される可能性が考えられる。そのため、降雨タイプ別検証実験での表４における「4. 関東に降雨状況がほぼ無い時間帯」（関東以外の一部の地域では降雨状況が存在）の抽出件数 0 件（各ツイート発生地点間における距離を 3km 以内とした場合）の結果に対しては、ツイート数が少ないことが要因の１つとして考えられる。また、東京都の降雨量データを用いた実験での図１９における F 尺度の低さ（0.4～0.5）においても、投稿されるツイート数がより多くなれば、近距離法での F 尺度もより向上するであろうことが考えられる。

組み合わせ法は、「状況」を特定するワードで抽出したツイートから、NLP 法と近距離法との組み合わせにより、ノイズとなるツイートをリダクションする手法である。組み合わせることで、従来手法（NLP 法）における情報の信ぴょう性や精度（ノイズコーパスの網羅性などに起因）の課題、近距離法での前述の状況特定精度の課題を補うことが可能となり、その結果、より精度の高いノイズリダクション効果が期待できる。東京都の降雨量データを用いた実験において、図１９、２０から、適合率は 84%あたり、NLP 法と比べた適合率においては、有意差も認められた（1.25%の名義的有意水準）。また、降雨タイプ別検証実験において、表４から「4. 関東に降雨状況がほぼ無い時間帯」（関東以外の一部の地域では降雨状況が存在）の降雨タイプを除いて、95%を超える適合率が得られた（各ツイート発生地点間における距離を 3km 以内とした場合）。さらに、表４の「1. 東京のゲリラ豪雨」の結果から見られるように、局所的な「状況」可視化においては、適合率に加え、F 尺度も 0.6 を超えており、また、図２２からも、地図上へのプロットによる概観把握が期待できる。一方、降雨タイプ別検証実験において、台風に伴う広範囲の集中豪雨や広範囲の普通の雨などは、各ツイート発生地点間における距離を 3km 以内とした場合、抽出できるツイート件数が少なくなり、F 尺度も低くなることが表４から確認することができた。このケースにおいては、地図上へのプロットによる概観把握では、NLP 法がより適していると考えられる。以上により、東京都の降雨量データを用いた実験において、組み合わせ法での抽出は、他の手法と比べて最も高い抽出精度（適合率）で抽出できることが明らかになった。そして、降雨タイプ別検証実験においても、組み合わせ法での抽出は、他の手法と比べて最も高い抽出精度（適合率）の結果が得られた。また、前述のとおり（4.3.1 参照）、従来手法との比較においては、従来手法の自然言語処理を一般化するのが難しいため（適合率は、フィルタのノイズ処理用ワード（状況ワード、bot id など）の数・網羅性に依存するが、この設定は状況によって異なるため）、本研究での比較対象を、自然言語処理法を拡張した NLP 法とした。

上記により、位置情報付きツイートを用いて状況を可視化する場合、都市部などのツイート数が多く見込まれる地域においては、抽出の目的にそぐわないノイズツイートが多く混入する可能性があり、その場合には、本手法（組み合わせ法）は、従来の自然言語処理だけの手法（NLP法）と比較して、適合率での有意差が認められ、また、抽出数においても、一定量のツイート数が得られた。さらに、降雨タイプ別検証においても、本手法（組み合わせ法）での抽出は、従来の自然言語処理だけの手法（NLP 法）と比較して、高い抽出精度（適合率）が得られ、その抽出数もある程度見込める結果となった。

この結果、研究目的で挙げた、特定状況可視化のためのノイズリダクションでは、特定状況の可視化において、本手法（組み合わせ法）を用いてノイズリダクションを行う場合、従来の自然言語処理だけの手法と比べて、情報の信ぴょう性の解決と抽出精度が向上できるといった仮説に対して、都市部などのツイート数が多く見込まれる地域においては、その有効性が明らかになった。このことにより、位置情報付きツイートを用いて状況可視化する場合、都市部などのツイート数が多く見込まれる地域においては、従来の自然言語処理だけの手法（NLP 法）と比べて、本手法のノイズリダクションは有用であることを示唆するものと考える。

今後の課題としては、NLP 法の精度向上が挙げられる。前述のとおり、振り返り･感想や状況変化における除去精度に影響を与え、本手法（組み合わせ法）の抽出精度に直結するからである。そのために、状況ごとのノイズツイートを効率良く収集するための手法の整備が不可欠となる。また、その際には、本研究では、あまり踏み込めなかった、状況ごとのノイズの調査･分析も併せて必要になると考える。また、ツイート数が少ない地域においては、本手法（組み合わせ法）における、NLP 法と近距離法との組み合わせのバランス検討を行うことで、情報の信ぴょう性の課題が残るものの、ある程度の精度かつ網羅性のある抽出が期待できる。組み合わせのバランス検討とは、近距離法のフィルタ効果を弱め（ツイート間の距離を長くする、同一時間帯を長くするなど）、その分、より NLP 法に頼った処理をさせるパラメータチューニングである。一方で、近距離法においては、状況や地域別ツイートの距離･時間間隔の特徴を分析･考慮することで、状況･地域に合わせた、より網羅性の高い抽出が実現できると考える。

6.2 特定地域の状況可視化のためのノイズリダクション手法

本節では、本手法の実験で実施した結果より考察する。表５、６から抽出単位が県レベルの熊本地震では、抽出範囲が広いためツイート数も多く、抽出したツイートの 9 割がノイズであった。n-gram の組合せ頻度閾値 5 では、適合率・再現率は 0.39～0.56・0.83～0.98 (3-gram)、0.32～0.44・0.95～1.00 (4-gram)、適合率向上・F 尺度は 7.17～11.21・0.53～0.66 (3-gram)、5.67～9.18・0.48～0.60 (4-gram) となった。閾値 3 では，適合率・再現率は 0.41～0.50・0.78～0.90 (3-gram)、0.34～0.44・0.91～0.96 (4-gram)、適合率向上・F 値は 7.08～11.13・0.55～0.62 (3-gram)、5.95～9.79・0.50～0.61 (4-gram) となった。

このように、県レベルの抽出単位では、一定程度の適合率向上、F 値を維持しながら、本手法で重視した再現率の点で、最低でも 0.78 という高い結果となった。特に 4-gram、頻度閾値 5 の条件では、0.95～1.00 という最も高い再現率となった。本手法では、名詞形態素の組合せにてフィルタを実現しているが 3-gram のように組合せ数がより少ない場合、正解ツイートにも該当する可能性が高く、それにより正解ツイートも除去され再現率が 4-gram と比べて低くなったと考えられる。頻度数も同様に、より少ない場合、正解ツイートに該当してしまい、閾値 3 にて、より再現率が低くなったと考えられる。ノイズの内訳は、図２３、表５、６からツイート数が多い場合、人以外が占める割合が高くなると考えられる。なお、本実験では、デマツイートの抽出は見られなかった。bot はノイズ全体の 9 割程度を占めた。種別では、bot は 8 種類（震源・震度系は 7、天気系は 1）、アプリ連携は 3 種類であった。したがって、ツイート数が多い場合、ノイズ混入の割合が高くなり、その占める割合は bot やアプリ連携が高いことが示された。また、従来手法では、再現率を重視し本手法と同程度の精度を得るために、事前作業時間として 103 分を要した（表９）。しかし、本手法を用いることで、この事前コストの削減が可能となる。

一方、表７、８から抽出単位が市レベルの常総市・境町では、ツイート数が少なくノイズ混入の割合も熊本地震に比べて低い。閾値 5 では、適合率・再現率は 0.27～0.73・0.82～1.00 (3-gram)、0.20～0.73・0.91～1.00 (4-gram) となった。ただし、再現率 1.0 は、表７、８より、ほぼフィルタ効果が得られていない。適合率向上は 1.00～2.91(3-gram)、1.00～1.07(4-gram) となり熊本地震に比べて効果も小さい。F 尺度は 0.43～0.76 (3-gram)、0.17～0.69 (4-gram) となった。閾値 3 では、適合率・再現率は 0.20～0.73・0.67～1.00 (3-gram)、0.10～0.53・0.91～1.00 (4-gram) となった。適合率向上は 1.00～2.13 (3-gram)、1.00～1.17 (4-gram) となりノイズ除去効果は低い。F 尺度は 0.31～0.76 (3-gram)、0.19～0.69 (4-gram) となった。この結果から抽出単位が市レベルでのツイート数が少ない場合 (30 件程度)、再現率のばらつきが見られ、フィルタ効果が得られないケースも確認できた。ただし 1 回のツイート処理数が 5 日分の 110 件程度以上であれば、再現率での 0.9 を超える抽出が期待できる。また、0.9 を

超えた際の閾値が 3 もしくは 5 であったため、表５、６を踏まえ閾値 5 の設定基準は、800 件程度以上のツイート数と考えられる。

以上により、ツイート数が少なくとも 110 件程度以上の場合、ノイズ混入の割合が高い可能性が考えられ、その場合、名詞形態素 4-gram、頻度閾値 3 もしくは 5 の条件で、より高い再現率（0.91 以上）でのノイズリダクションが確認できた。また、n 数が大きい場合、再現率がより高くなることも確認できた。一方、適合率は平均 0.4 程度であったが、再現率を重視した代償であり想定の結果と考える。また、適合率を補うための 1 つの方法として、定性的ではあるが、図２４、２５からノイズリダクション後に、ワードクラウドなどの全体俯瞰を利用することで（「避難」、「水」、「店」などが炙り出された）状況の詳細分析（ドリルダウン）につなげられる可能性も確認できた。

この結果、研究目的で挙げた、特定地域の状況可視化のためのノイズリダクションでは、特定地域状況の可視化において、本手法の実行時に機械的に生成される名詞の形態素 n-gram のフィルタを用いてノイズリダクションを行う場合、従来の人手によるフィルタ生成の手法と比べて、フィルタ生成のための時間と労力（コスト）の課題を解決し、また、高い再現率を維持した抽出ができるという仮説に対して、ツイート数が多く見込まれる地域において、その有効性が示唆された。このことにより、地震などより広域で抽出する被災地やツイート数が多く見込まれる首都圏などが被災地となる場合、従来の人手によるフィルタ生成の手法と比べて、本手法のノイズリダクションは有用であることを示唆するものと考える。

さらに、ツイート数が多い場合には、本研究で開発した、特定状況可視化のためのノイズリダクションと組み合わせることで（特に近距離法）、デマ情報の除去への寄与も考えられる。たとえば、熊本地震では、「ライオン逃げた」といったデマツイートが発生したが（熊本日日新聞（2016/7/21）によれば、１時間で２万件以上のリツイートが発生）、これについて、本手法（特定地域の状況可視化のためのノイズリダクション）を用いて、デマツイートが除去できずに抽出されてしまった場合（形態素名詞の n-gram の n 数が大きかった場合など）、さらに、近距離法を適用することで、その地域から投稿されたデマツイートを除去できる可能性が考えられるからである。具体的には、本手法の適用後に、さらに、デマかもしれないワード:「ライオン」にて抽出する。その後、近距離法を適用し、もし本当であれば、同一時間帯、同一地域、複数人のツイート条件に合致し該当ツイートを抽出できるが、デマの場合は除去される、といった流れになる。しかし、残念ながら本実験では、熊本県から投稿されたツイートには、ライオンのデマツイートは確認できなかった。そのため、ライオンに関する、ほぼすべてのデマツイートは、他県からの投稿であったと考えられる。

また、本実験では、被災地のケースが熊本県と茨城県からの 2 つであったため、結果における検定が行えなかった。したがって、今後の課題として、より多くの被災地データを集めるこ

とで、より確かな手法の開発･検証につなげていきたいと考えている。また、より性能（F 尺度など）を改善するために、図１５で示した、頻度閾値や形態素 n-gram の組み合わせについての最適な設定検討についても、課題として挙げたい。併せて、ツールでの処理不能となった特殊文字対応も検討課題としていきたい。

第7章 結論

近年、台風の大型化や集中豪雨、ゲリラ豪雨、そして、地震などの事象により、日本各地において頻繁に被害が生じている。総務省消防庁（2010）によると、集中豪雨などによる人的被害を軽減させるために、迅速な状況の把握と、早めの避難行動の重要性が指摘されている。そのような中、ゲリラ豪雨、突風･竜巻、雹などの突発的な事象や、それらの事象および地震などによる被害状況（河川の氾濫、浸水･冠水、交通機関の乱れや事故、停電、建物の倒壊など）の可視化については、速報性･網羅性の観点から十分とは言えない状況にあった。

しかし、昨今の SNS の利用、スマートフォンの普及により、その場に居合わせた人々が、その場におけるさまざまな「状況」（現在降雨状態にあることなど）を発信しやすい環境が整備されてきた。そして、SNS の活用という点では、特に、Twitter を活用した研究事例が数多く見られる。情報の速報性が高い、スポーツ中継やニュース番組などにおいて一体感を感じられる、緊急時の連絡手段としても活用できるなどの特徴があるからである。さらに、Twitter には、スマートフォンに付いている GPS 機能を利用し、位置情報を付与したツイートが可能であり、これにより、ツイートの投稿場所に関連した分析が可能となった。そのため、位置情報付きツイートによる「状況」可視化のアプローチは、前述した、速報性、網羅性の課題を解決する 1 つのソリューションとなり得ることが考えられる。

一般的に、位置情報を利用して、「状況」の可視化を行うには、ツイートからその「状況」を示すワードを指定して抽出し、そこに付加されている位置情報を用いて、地図上にプロットもしくは大字･町丁目レベルに変換して可視化することなどが考えられる。しかし、単純なワード指定による抽出方法では、精度に問題が生じる場合がある。このような抽出方法では、その「状況」が生じていた場所･時間から離れた投稿、たとえば、振り返りや感想、予報などのツイートも抽出してしまうからである。したがって、「状況」を可視化する上では、このような可視化したい状況と直接関係しないツイート、つまり、ある目的にとって不要なツイートをノイズとして捉え、抽出時には、それらを除去することが不可欠となる。

そのため、特定状況の可視化におけるノイズリダクションにおいては、従来の研究において、事前にノイズとなるキーワードを定義してノイズを除去する方法や、bot などの非個人ユーザーのアカウントを特定･除去するアプローチが取られてきた。しかし、個人ユーザーの発言において、その「状況」が、今現在もその場所で続いているのか考慮するための「時間」と「場所」との両側面からのアプローチは考慮されていない。そのため、「明日は雨かな？」、「今朝の雨は酷かった」、「埼玉では豪雨らしいね」などのツイート除去には対応できず、これによる抽出精度の課題が存在していた。また、従来手法では、ツイート内のテキスト（BotID 含む）処理を実施する自然言語処理でのアプローチとなっており、この場合、抽出したツイート

について、情報の信ぴょう性の課題が存在していた。さらに、従来の自然言語処理でのアプローチでは、フィルタの候補となるキーワードの機械的収集、および、そこから、フィルタを機械的に生成する、といった機械的な処理は行われていない。そのため、そこには、少なからず時間と労力（コスト）がかかり、それにより、主観的なアプローチが入り込む可能性も考えられ、その結果、抽出精度への影響が懸念される。このように、従来の特定「状況」の可視化におけるノイズリダクションでは、抽出精度、情報の信ぴょう性、人手によるフィルタ生成のための時間と労力（コスト）の課題が存在していた。

一方、特定地域の状況可視化におけるノイズリダクションにおいて、位置情報付きツイートを上手く活用できれば、被災地において、被災直後からのメディアでは報道されにくい現地の声を知る有用な手がかりとなり得る。具体的には、各ツイートに付与されている位置情報を利用し、事前に、広範囲で取得・変換（緯度・経度から大字・町丁目レベルに）しておく方法が考えられる。事前に変換しておくことで、地域名による地域の抽出操作が、より容易になるからである。しかし、このような方法で抽出されたツイートにおいても、精度に問題が生じる場合がある。たとえば、bot やツイート連携アプリなどの可視化したい状況に直接関連しないツイートまでも抽出してしまうからである。したがって、特定地域の状況可視化を行う上でも、可視化したい状況以外のツイート（ノイズ）の除去処理が欠かせない。

そのため、特定地域の状況可視化には、多様な状況を捉えるために、フィルタ機能において、ノイズをリダクションしながらも、より多くの正解データを抽出する、つまり、再現率を重視したアプローチが考えられる。そして、再現率を重視する上では、フィルタ生成において、事前に、より多くのノイズツイートを集め、そこから、ノイズツイートだけに該当する条件設定が重要となる。しかし、従来手法では、より多くのノイズツイートの識別・収集とノイズツイートだけに該当する条件設定においては、事前の人手による処理（フィルタ生成）が行われており、少なからず時間と労力（コスト）の課題が存在していた。

そこで、本研究では、特定状況の可視化、および、特定地域状況を可視化するための２つのノイズリダクション手法について、抽出精度、情報の信ぴょう性、人手によるフィルタ生成のための時間と労力（コスト）、といった３つの課題を解決するために、従来手法と異なるアプローチに取り組んだ。具体的には、特定状況可視化のためのノイズリダクションでは、気象現象における降雨状況を用いて、主に、従来の自然言語処理だけの手法に位置情報をベースとした近距離法を加えて、抽出精度、情報の信ぴょう性の課題解決を試みた。また、特定地域の状況可視化のためのノイズリダクションでは、地震や集中豪雨により被災した熊本県、茨城県の状況（災害状況）を対象に、そこから投稿されたツイートの抽出時（ノイズリダクション時）に、機械的に、名詞の形態素 n-gram のフィルタ生成を行うことで、人手によるフィルタ生成のための時間と労力（コスト）の課題解決を試みた。

その結果として、特定状況可視化のためのノイズリダクション手法では、次の３点が得られた。１点目は、都市部などのツイート数が多く見込まれる地域においては、抽出の目的にそぐわないノイズツイートが多く混入する可能性があること。２点目は、本手法は、従来の自然言語処理だけの手法と比較して、適合率での有意差が認められ、また、抽出数において一定量のツイートが得られること。さらに、降雨タイプ別検証においても、本手法での抽出は、従来の自然言語処理だけの手法と比較して、高い抽出精度（適合率）が得られ、その抽出数もある程度見込めること。３点目は、本手法では近距離法を含んでおり、また、評価システムを用いた客観的検証による高い抽出精度（適合率）からも、従来の自然言語処理だけの手法と比べて、より情報の信ぴょう性の課題に対応可能であること。

これにより、研究目的で挙げた、特定状況可視化のためのノイズリダクションでは、特定状況の可視化において、本手法（組み合わせ法）を用いてノイズリダクションを行う場合、従来の自然言語処理だけの手法と比べて、情報の信ぴょう性の解決と抽出精度が向上できるといった仮説に対して、都市部などのツイート数が多く見込まれる地域においては、その有効性が明らかになった。このことにより、位置情報付きツイートを用いて状況可視化する場合、都市部などのツイート数が多く見込まれる地域においては、従来の自然言語処理だけの手法と比べて、本手法のノイズリダクションは有用であることを示唆するものと考える。

また、特定地域の状況可視化のためのノイズリダクションの結果として、次の３点が得られた。１点目は、抽出単位が県レベルの地震においては、抽出範囲も広いためノイズ混入の割合も高く（未フィルタの場合の適合率:0.05～0.08 であるため）、一定程度の、フィルタ適用後の適合率向上と F 尺度を維持しながら、形態素 4-gram において、再現率が高い結果（頻度閾値５:0.95～1.00、頻度閾値３:0.91～0.96）となること。さらに、頻度閾値に着目すると、頻度数５において、再現率がより高い結果となること。２点目は、抽出単位が市レベルにおいては、ツイート数が少ない可能性があり、ノイズ混入の割合も、県レベルでの抽出と比べて、それほど高くはないが（未フィルタの場合の適合率:0.09～0.53）、処理するツイート数が 110 件程度以上であれば、4-gram、頻度閾値３もしくは５の条件で再現率 0.91 という高い結果となること。３点目は、処理するツイート数が極端に少ない状況（３０件程度）では、フィルタ効果が弱い結果となること。

これにより、研究目的で挙げた、特定地域の状況可視化のためのノイズリダクションでは、特定地域状況の可視化において、本手法の実行時に機械的に生成される名詞の形態素 n-gram のフィルタを用いてノイズリダクションを行う場合、従来の人手によるフィルタ生成の手法と比べてフィルタ生成のための時間と労力（コスト）の課題を解決し、また、高い再現率を維持した抽出ができるという仮説に対して、処理するツイート数を、少なくとも 110 件程度以上とした場合に、形態素名詞 4-gram、頻度閾値３もしくは５の条件において、その有効性が示唆さ

れた。このことにより、地震などより広域で抽出する被災地やツイート数が多く見込まれる首都圏などが被災地となる場合、従来の人手によるフィルタ生成の手法と比べて、本手法のノイズリダクションは有用であることを示唆するものと考える。その結果、本手法は現地の声を、より手間をかけずに知る手がかりの１つとして寄与するものと考える。

最後に、本研究は、気象･防災といった人間環境科学の分野と SNS の分析･活用という人間情報科学の研究分野を実践･実用的に結びつけ、真に人間科学を深める研究であると考える。

謝辞

本研究を行うにあたり、日頃からご指導ご鞭撻をいただきました西村昭治教授に、深く感謝を申し上げます。

また、主に、中間報告におきまして、大変有益なレビュー・アドバイスをいただきました、菊池英明教授、金群教授、尾澤重知准教授に、深く感謝を申し上げます。

参考・引用文献

≪引用文献≫

Diansheng Guo, Chao Chen(2014).Detecting Non-personal and Spam Users on Geo-tagged Twitter Network, Transactions in GIS, 18(3) 370-384.

Erik Ward, Kazushi Ikeda, Maike Erdmann, Masami Nakazawa, Gen Hattori, Chihiro Ono (2012) .Automatic query expansion and classification for television related tweet collection,「研究報告データベースシステム (DBS) 」,2012-DBS-155(19),1-8.

蛭田慎也・米澤拓郎・徳田英幸(2013).場所誘因型位置情報付き発言の検出と可視化.情報処理学会論文誌,54(2),710-720.

株式会社トライバルメディアハウス・株式会社クロス・マーケティング(2012).ソーシャルメディア白書 2012 翔泳社.

影澤秀明・廣井慧・奥矢淳・香取啓志・加藤朗・砂原秀樹(2014).Twitter を用いたセンシングシステムの提案と考察.「マルチメディア、分散協調とモバイルシンポジウム 2014 論文集」, 725-732.

河居寛樹・上野秀剛・伊原彰紀(2013).形態素 N-gram を用いた不具合修正完了ソースコードの特定.「情報通信学会技術研究報告.MSS システム数理と応用」,112(457), 57-62.

岸浩稔・中西航(2013).位置情報付き tweet による被害状況の逐次把握可能性の検討.「生産研究」, 65(4), 529-532.

高橋哲朗(2013).発言同期を用いたマイクロブログ著者の位置推定 情報処理学会研究報告.自然言語処理研究会報告,2013(17),1-7.

≪引用 URL≫

株式会社ソフテル (2013) 【php】twitter Streaming API の statuses/filter を試す.
https://www.softel.co.jp/blogs/tech/archives/3929 (2018-09-30)

「熊本日日新聞」2016 年 7 月 21 日 「ライオン逃げた」デマ投稿の男逮捕 全国初
<https://this.kiji.is/128650230205220348?c=92619697908483575> (2018-10-04)

Onozka(2012) MeCab と NLTK を使って最瀬語と共起関係を出力する - Men talking over coffee with smoking Ark Royal.
<http://d.hatena.ne.jp/r_onodr/20120928/1348806618> (2018-09-30)

総務省消防庁 平成 22 版消防白書 局地的大雨や集中豪雨に備えて
<http://www.fdma.go.jp/html/hakusho/h22/h22/html/1-5c-2_kakomi08.html> (2018-09-30)

すがやみつる (2009) こんにちは統計学 : Python による χ 2 乗検定・t 検定・U 検定・分散分析・多重比較・相関係数の計算.
<http://www.m-sugaya.jp/python/index.html>.

yosiopp (2014) 簡易的な逆ジオコーディング
<http://yosiopp.net/archives/72> (2018-09-30)

≪参考文献≫

荒川豊・田頭茂明・福田晃(2010).Twitter 分析に基づく位置依存文字列の抽出.情報処理学会研究報告.MBL,[モバイルコンピューティングとユビキタス通信研究会研究報告],2010(10),1-6.

Fujita, H. (2013). Geo-tagged Twitter collection and visualization system. Cartography and
Geographic Information Science, 40(3), 183-191. doi:10.1080/15230406.2013.800272

服部充典.位置情報付きツイートから事象（自然現象・異常気象）・災害を可視化する手法の開発,早稲田大学,2014,修士論文.

服部充典・西村昭治(2018).位置情報付きツイートから状況可視化のためのノイズリダクション手法の検討.地理情報システム学会誌(GIS-理論と応用),26(1),1-11.

井上和也・戸田圭一・市川温・多田彰秀(1999).1999 年福岡市における都市型水害について.京都大学防災研究所年報,(43),307-323.

加藤宏紀・荒牧英治・宮部真衣・吉田稔・佐藤一誠・中川裕志(2012).ソーシャルメディアからの地域固有表現の抽出(地域情報&ソーシヤルメデイア,第 4 回集合知シンポジウム).電子情報通信学会技術研究報告.NLC,言語理解とコミュニケーション,112(367),29-34.

凍田和美・菊池達哉・吉山尚裕・柴田雄企・高橋雅也・竹中真希子・青木栄二(2011).地域住民の“信頼”と“人間関係”を基盤にした地域防災 SNS の開発.情報処理学会研究報告.CSEC,[コンピュータセキュリティ],2011(16),1-6.

毛利隆夫・田中英彦(1995).記憶に基づく推論による天気予測.人工知能学会誌,10(5),798-805.

津口裕茂・加藤輝之(2014).集中豪雨事例の客観的な抽出とその特性・特徴に関する統計解析.天気,61(6),455-469.

山田和貴・斉藤裕樹(2010).マイクロブログサービスの位置情報タグと発言コンテキスト解析を用いた行動推定システムの設計.情報処理学会研究報告. データベース・システム研究会報告,2010(21),1-6.

山守一徳(2010).災害時用掲示板の開設.三重大学教育学部研究紀要,61,13-19.

≪参考 URL≫

赤澤康幸 F-尺度(F-measure)について
<http://www.cse.kyoto-su.ac.jp/~g0846020/keywords/f-measure.html> (2018-09-30)

centos mecab に wikipedia はてなキーワード 辞書登録 | まとめーたー | gihyo.jp … 技術評論社 <http://matomater.com/16376/> (2018-09-30)

Code Samples - Google Maps JavaScript API v3 - Google Developers
<https://developers.google.com/maps/documentation/javascript/examples/>
(2018-09-30)
Coloreminder - Html & Css Color Code <http://coloreminder.com> (2018-09-30)

藤崎祥見・渡部徹太郎・林田敦 第10回　MongoDBでの集計処理：MongoDBでゆるふわDB
体験｜gihyo.jp … 技術評論社 <http://gihyo.jp/dev/serial/01/mongodb/0010>
(2018-09-30)

藤崎祥見・渡部徹太郎・林田敦 第11回　MongoDBのバックアップ：MongoDBでゆるふわ
DB体験｜gihyo.jp … 技術評論社 <http://gihyo.jp/dev/serial/01/mongodb/0011>
(2018-09-30)

緯度経度から住所を取得するPHPプログラムを作ってみる《大人の自由研究》
<http://ukkey3.blog33.fc2.com/blog-entry-329.html> (2018-09-30)

株式会社LIG（リグ）ビッグデータを分析できる！可視化を利用したウェブサイトまとめ｜
株式会社LIG <http://liginc.co.jp/web/service/other-service/90986> (2018-09-30)

Koma新知識: MongoDB基本使用 <http://komanew.blogspot.jp/2013/10/mongodb.html>
(2018-09-30)

MacにmongoDBをインストールする。そしてちょっと使ってみる。 - nigoblog
<http://nigohiroki.hatenablog.com/entry/2013/01/05/234631> (2018-09-30)

macでmongoDBのインストールから起動まで - dackdive's blog
<http://dackdive.hateblo.jp/entry/2014/02/09/020251> (2018-09-30)

松下敦郎 PythonとStreaming APIなどを使ってツイッターをだら見する - 松下小屋
<http://red.ribbon.to/~misdreavus/z/pythontwitter/pythontwitter.html> (2018-09-30)

mecabにwikipediaのタイトルリストを追加する ｜ ミラボ
<http://log.miraoto.com/2012/11/703/> (2018-09-30)

mongodb/mapreduceとAggregation Frameworkを試してみた - kanetann's blog
<http://kanetann.hatenablog.com/entry/2013/07/02/190229> (2018-09-30)

MongoDBに一括でデータを登録する(Bulk Inserts) - Symfoware
<http://symfoware.blog68.fc2.com/blog-entry-303.html> (2018-09-30)

MongodbのDB変更 - kiita's blog
<http://kiita312.hatenablog.com/entry/2013/03/03/130935> (2018-09-30)

MongoDBのシェルの操作方法メモ - Qiita
<http://qiita.com/yuiseki/items/1c656d6bab0307e1510c> (2018-09-30)

MongoDBをPythonで操作する(PyMongo使用) - Symfoware
<http://symfoware.blog68.fc2.com/blog-entry-302.html> (2018-09-30)

Nakamura, K. MongoDB Index 基本 - /var/log/kozy4324
<http://kozy4324.github.io/blog/2012/06/19/mongodb-index/> (2014-12-21)

西谷圭介 MongoDB のインストールと基本操作 (1/3) ：CodeZine
<http://codezine.jp/article/detail/6982> (2018-09-30)

python で形態素解析を行う(MeCab) - Anyaneko
<http://alotofwe.github.io/blog/2014/02/03/pythondexing-tai-su-jie-xi-woxing-u-mecab/> (2014-12-21)

Python で Mecab を使うための準備 - otukutun の日記
<http://otukutun.hatenablog.com/entry/2013/07/05/001537> (2018-09-30)

総務省消防庁 平成 26 版消防白書
<http://www.fdma.go.jp/html/hakusho/h26/h26/pdf/h26_all.pdf> (2018-09-30)

Streaming API から URL を収集してアレコレ
<http://blog.gen-zoh.net/?p=45> (2014-12-21)

Sugawara, Y. 【保存版】TwitterAPI1.1 REST API 全項目解説 | DX.univ
<http://dx.24-7.co.jp/twitterapi1-1-rest-api/> (2014-12-21)

常川真央 情報推薦アルゴリズムの効果をどう評価するのか - 図書館情報学を学ぶ
<http://d.hatena.ne.jp/kunimiya/20081223/p1> (2018-09-30)

tweepy で streaming を使う - 備忘録
<http://monowasure78.hatenablog.com/entry/2013/11/26/tweepy で streaming を使う> (2018-09-30)

Twitter, Inc. Streaming message types | Twitter Developers
<https://dev.twitter.com/streaming/overview/messages-types> (2018-09-30)

Twitter, Inc. Tweets | Twitter Developers
<https://dev.twitter.com/overview/api/tweets> (2018-09-30)

tweepy の Streaming API - 程よくネガティブ、適度にポジティブ
<http://d.hatena.ne.jp/vent_et_neige/20120409/1333956955> (2018-09-30)

Twitter のストリーミング API を試してみた @ ともの技術メモ
<http://tomono.eleho.net/2013/09/03/4505/> (2018-09-30)

Twitter の特徴 | メリットデメリット.com
<http://xn--9ckaldc9ld2ee.com/web/twitter.html> (2018-09-30)

Twitter の TL を全部 MongoDB にぶち込んでニヤニヤする - 凹み Tips
<http://tips.hecomi.com/entry/20120908/1347094725> (2018-09-30)

Twitter Streaming API についてのメモ - console.lealog();
<http://lealog.hateblo.jp/entry/2013/03/10/100845> (2018-09-30)

Twitter Streaming API を使う 1 - DoboWiki
<http://wiki.dobon.net/index.php?.NET%A5%D7%A5%ED%A5%B0%A5%E9%A5%DF%A5%F3%A5%B0%B8%A6%B5%E6%2F96> (2018-09-30)

渡部徹太郎 初心者向け MongoDB のキホン！
<http://www.slideshare.net/tetsutarowatanabe/mongo-db-32210761> (2018-09-30)

矢吹太朗 Streaming API で取得したつぶやきの処理方法 | 配電盤
<http://blog.unfindable.net/archives/4302> (2018-09-30)

矢吹太朗 Streaming API で大量のつぶやきをリアルタイムに保存する方法 (Python 編) | 配電盤 <http://blog.unfindable.net/archives/4257> (2018-09-30)

Yamamoto, Y. 第 3 回 Twitter API 勉強会 - ストリーミング API #twtr_hack
<http://www.slideshare.net/yusukey/3twitter-api-api> (2018-09-30)

付録

各ツールのソースコード

各番号は表１、表２、表３のツール一覧に対応。

表１ - No.1 geoTest_99y.py

```
#!/usr/bin/python
# coding: UTF-8
import sys

argvs = sys.argv

#f = open('exportTwitter.py', 'w') #  書き込みモードで開く
f = open('exportTwitter.py', 'w') #  書き込みモードで開く

str = """#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pymongo import Connection
import re
import sys

argvs = sys.argv

class GeoTest:
	def __init__(self):
		### MongDB へのアクセス
		#コネクション作成
		self.con = Connection('localhost', 27017)
		#コネクションから tweet データベースを取得
		self.db = self.con.tweety

	def __del__(self):
		self.con.disconnect()

	def getGeoData(self):
"""

f.write(str)

tempxx = " ¥"$gte¥":¥"%s¥",¥"$lt¥":¥"%s¥" " % (argvs[1],argvs[2])
tempyy = '{"created_at": {' + tempxx + '},"geo.coordinates":{"$ne":
None}},{"user.id":1,"geo":1,"lang":1,"text":1,"created_at":1}'

str = "		for data in self.db.tweet_2.find(" + "%s" % tempyy + ",timeout=False):"

f.write(str)

str = """
			p = re.compile("(.+)¥[(.+),(.+)¥]")
			m = p.match(str(data['geo']))
			lati = str(m.group(2)).strip()
			longi = str(m.group(3)).strip()

			print data['user']['id'],

			print data['lang'].encode('utf-8'),
			print data['created_at'].encode('utf-8'),
			print lati.encode('utf-8'),
			print longi.encode('utf-8'),

			#1 行のつぶやきにするため、改行取り除く
			temp = data['text']
			temp = temp.replace('¥¥n',' ')
			print temp.encode('utf-8')
```

```
if __name__ == "__main__":

    gt = GeoTest()
    gt.getGeoData()"""

f.write(str) # 引数の文字列をファイルに書き込む
f.close() # ファイルを閉じる
```

表 1 - No.2 exportTwitter.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pymongo import Connection
import re
import sys

argvs = sys.argv

class GeoTest:
    def __init__(self):
        ### MongDB へのアクセス
        #コネクション作成
        self.con = Connection('localhost', 27017)
        #コネクションから tweet データベースを取得
        self.db = self.con.tweety

    def __del__(self):
        self.con.disconnect()

    def getGeoData(self):
            for data in self.db.tweet_2.find({"created_at": { "$gte":"2014-07-01T00:00+9:00","$lt":"2014-07-
31T23:59+9:00" },"geo.coordinates":{"$ne":
None}},{"user.id":1,"geo":1,"lang":1,"text":1,"created_at":1},timeout=False):
                p = re.compile("(.+)¥[(.+),(.+)¥]")
                m = p.match(str(data['geo']))
                lati = str(m.group(2)).strip()
                longi = str(m.group(3)).strip()

                print data['user']['id'],

                print data['lang'].encode('utf-8'),
                print data['created_at'].encode('utf-8'),
                print lati.encode('utf-8'),
                print longi.encode('utf-8'),

                #1 行のつぶやきにするため、改行取り除く
                temp = data['text']
                temp = temp.replace('¥n',' ')
                print temp.encode('utf-8')

if __name__ == "__main__":

    gt = GeoTest()
    gt.getGeoData()
```

表 1 - No.4 returnNumber.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import math

for line in sys.stdin:
    temp = int(line) * 2 / 3
    #print line
    print temp, int(line)-temp
```

表 1 - No.7 export_3.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import ReverseGeocoding

rb = ReverseGeocoding.ReverseGeocoding()

for line in sys.stdin:
	items = line.split(" ",5)
	#print items

	lati = items[3].strip()
	longi = items[4].strip()
	tempx = rb.getAddress(lati,longi)
	if tempx != None:
		print items[0].strip(),
		print items[1].strip(),
		print items[2].strip(),
		print tempx.encode('utf-8'),
		print items[5].strip()
```

表 1 - No.8 ReverseGeocoding.py

```
#!/usr/bin/env python
# coding: utf-8
import os
import sqlite3
import unicodecsv
import math

class ReverseGeocoding:
    def __init__(self):
        # データベースに接続
        db_filename = 'address.sqlite3'
        self.conn = sqlite3.connect(db_filename)
        self.D = {}

    def getAddress(self,latitude,longitude):
        # カーサー
        self.cur = self.conn.cursor()

        lati = latitude[0:4]
        longi = longitude[0:5]

        # 1 行ずつ読み込む
        sql = "select * from convert where latitude like '" + lati + "%'" + " and longitude like '" + longi + "%' order by
latitude,longitude"
        # print sql
        self.cur.execute(sql)
        self.temp = []
        for row in self.cur:
            # 元々の緯度・経度情報にマッチしたテーブル上の緯度・経度について、それぞれ、その差分を取り、その後絶対値
にして、
            # 最終的に、緯度・経度のそれぞれの差分を合計する
            tempx = math.fabs(float(latitude) - float(row[0])) + math.fabs(float(longitude) - float(row[1]))
            self.tempxx = str(tempx) + " " + str(row[0]) + " " + str(row[1]) + " " + row[2]
            self.temp.append(self.tempxx)
            #print self.tempxx
        self.cur.close()
        #print self.temp

        #上記で合計した差分の一番小さい住所を return する
        #レコードすべて return する
        #return min(self.temp)

        #住所のみ return する
        x = None
        if len(self.temp) > 0:
            x = min(self.temp).split()[3]

        return x
```

```
    def __del__(self):
        self.conn.close()


if __name__ == "__main__":

    rg = ReverseGeocoding()

    latitude = "34.78585089"
    longitude = "135.52950444"

    #lati = round(float(latitude), 3)
    #longi = round(float(longitude), 3)

    print "******* " + rg.getAddress("34.78585089", "135.52950444")
    print "******* " + rg.getAddress("35.4708417", "139.4265385")
    print "******* " + rg.getAddress("38.28448738", "140.89024897")
```

表 1 - No.9 loadConvertionTable.py

```
#!/usr/bin/env python
# coding: utf-8
import os
import sqlite3
import unicodecsv

# データベースに接続
db_filename = 'address.sqlite3'
conn = sqlite3.connect(db_filename)

# 逆ジオコーディング用の CSV ファイルを読み込む
csvfile = "all_utf8.csv"
reader = unicodecsv.reader(file(csvfile, 'r'))

# データベースに登録する
for row in reader:

    sql = "insert into convert values(" + row[6] + "," + row[7] + "," + """ + row[1]+row[3]+row[5] + """ + ")"
    print sql
    conn.execute(sql)

    #print sql
    #sql

conn.commit()
conn.close()
```

表 1 - No.10 makeNoiseCorpus_xxxxx_V2.sh

```
#!/usr/bin/env bash
#東京都
echo "削除&作成  2014_6_rain-noise_Tokyo.data"
rm 2014_6_rain-noise_Tokyo.data
#
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都西多摩郡  西多摩郡% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都八王子市  八王子市% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都世田谷区  世田谷区% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都千代田区  千代田区% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都江戸川区  江戸川区% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都大田区  大田区% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都青梅市  青梅市% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都練馬区  練馬区% 2014_6_rain-noise_Tokyo.data
./makeNoiseCorpus_V2.sh 2014_6_tokyo_rain.data  東京都府中市  府中市% 2014_6_rain-noise_Tokyo.data
#
cat 2014_6_rain-noise_Tokyo.data | cut -d" " -f3-20 > temp_2014_6_rain_noise_Tokyo.data
```

表 1 - No.11 makeNoiseCorpus_V2.sh

```
#!/usr/bin/env bash

#echo $1
#echo $2
#echo $3
#echo $4

echo "$1 $2 $3"
cat $1 | ggrep -E $2 | python exportNoiseComment_V2.py $3 n >> $4
```

表 1 - No.12 exportNoiseComment_V2.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import sqlite3
import datetime
import rainCheck

argvs = sys.argv

for line in sys.stdin:
    items = line.split("T",2)

    #D1 = items[0].strip()
    D1t = items[0].split(" ",2)
    D1 = D1t[2].strip()
    D2 = items[1].split(":",2)

    dd1 = int(D2[0])+1
    if dd1 == 24:
        # 23:00 + 1:00 = 24:00 となった場合
        # 日付を変更する操作を実施
        # 文字列の日付データを、datetime 型に変換する
        dd3 = datetime.timedelta(days=1) + datetime.datetime.strptime(D1, '%Y-%m-%d')
        D1 = dd3.strftime('%Y-%m-%d')

        # さらに、24:00 ではなく、00:00 に変更する
        dd1 = 0

    temp = "%sT%02d:00:00" % (D1,dd1)
    time = "'" + temp + "'"
    #print time,
    rc = rainCheck.rainCheck()
    if argvs[2] <> "r":
        if rc.getRainCheck(argvs[1], time) == "×":
            sys.stdout.write(line)
    else:
        if rc.getRainCheck(argvs[1], time) == "◎":
            sys.stdout.write(line)
```

表 1 - No.13 makeFilter_V3.sh

```
#!/usr/bin/env bash

## "雨"で始まる形態素３gram の組は、頻度 3 以上を抽出
python getTrend_co_freqz.py tri $1 n | cut -d" " -f2-4 | ggrep -E "^雨" | ggrep -E -v "[0-9] * *" | ggrep -E -v "* [0-9] *" |
ggrep -E -v "* * [0-9]" | ggrep -E -v "[０-９] * *" | ggrep -E -v "* [０-９] *" | ggrep -E -v "* * [０-９]" | ggrep -v "(" | ggrep -v
")" | sort > zzz.data

## 振り返り
## "明日、今日、昨日、今朝、先日、夕方"で始まる形態素３gram の組は、頻度 1 以上を抽出


python getTrend_co_freqz.py tri 1 n | cut -d" " -f2-4 | ggrep -E "雨" | ggrep -E "^明日 | ^昨日 | ^今朝 | ^先日 | ^夕方 | ^昨晩 | ^
深夜 | ^明け方" | ggrep -E -v "[0-9] * *" | ggrep -E -v "* [0-9] *" | ggrep -E -v "* * [0-9]" | ggrep -E -v "[０-９] * *" | ggrep -E -
v "* [０-９] *" | ggrep -E -v "* * [０-９]" | ggrep -v "(" | ggrep -v ")" | sort >> zzz.data

## 気象・ニュース ストップワードを指定
```

```
echo "ggrep -v -E '時点|降水確率|天気|予報|警報|注意|雨量|ニュース|news|速報'" > qqq

cat zzz.data | python generatCard_tri_zz.py > ggg
```

表 1 - No.14 getTrend_co_freqz.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import nltk
from nltk.util import ngrams
import sys
import mecab_tokenizer

argvs = sys.argv
num = int(argvs[2])

raw = open('temp.data').read()

tokens = mecab_tokenizer.tokenize(unicode(raw,'utf-8'))

if argvs[1] == "uni":
    #words = raw.split();
    unigrams = nltk.bigrams(tokens)
    fd = nltk.FreqDist(unigrams)
    for w in fd:
        #print fd[w]
        #print w[0],w[1],w[2],fd[w]
        if fd[w] >= num:
            print fd[w],w[0]

if argvs[1] == "bi":
    #words = raw.split();
    bigrams = nltk.bigrams(tokens)
    fd = nltk.FreqDist(bigrams)
    for w in fd:
        #print fd[w]
        #print w[0],w[1],w[2],fd[w]
        if fd[w] >= num:
            print fd[w],w[0],w[1]

elif argvs[1] == "tri":
    #words = raw.split();
    trigrams = nltk.trigrams(tokens)
    fd = nltk.FreqDist(trigrams)
    for w in fd:
        #print fd[w]
        #print w[0],w[1],w[2],fd[w]
        if fd[w] >= num:
            print fd[w],w[0],w[1],w[2]

elif argvs[1] == "four":
    #words = raw.split();
    fourgrams = nltk.ngrams(tokens,4)
    fd = nltk.FreqDist(fourgrams)
    for w in fd:
        #print fd[w]
        #print w[0],w[1],w[2],fd[w]
        if fd[w] >= num:
            print fd[w],w[0],w[1],w[2],w[3]
```

表 1 - No.15 mecab_tokenizer.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import mecab_library
import pickle

# tweets = pickle.load(open("cluster2_tweets.pickle","r"))

def splitStr(str, num):
    l = []
    for i in range(num):
        l.append(str[i::num])
```

```
    l = ["".join(i) for i in zip(*l)]
    rem = len(str) % num   # zip で捨てられた余り
    if rem:
        l.append(str[-rem:])
    return l

def classifyPos(words, pos_list=["名詞","形容詞","動詞","助動詞","副詞","形容動詞"]):

    #stop_words = [u"0",u"1",u"2",u"3",u"4",u"5",u"6",u"7",u"8",u"9",u"0",u"RT",u"rt",u"http",u"https",u"gt",u"lt",u"です",u"
ます",u"いる",u"する"]
    stop_words = [u"RT",u"rt",u"http",u"https",u"gt",u"lt",u"://",u".",u":",u"~",u"/",u"#",u"(",u")",u"（",u"）",u"〜"]



    texts = []
    for token in words.tokens:
        if unicode(token.surface, 'utf-8') in stop_words: continue
        #elif len(unicode(token.surface, 'utf-8')) == 1: continue # 1 文字はスルー
        elif token.pos in pos_list:
            if token.basic == "*":
                texts.append(token.surface)
            elif token.pos == "感動詞": continue
            elif token.pos_detail1 == "非自立": continue
            else:
                texts.append(token.surface)
    return texts

def tokenize(text, pos_list=["名詞", "形容詞"]):
    sent = text.encode('utf-8')
    if len(sent) > 2000000:
        sents = splitStr(sent, 2000000)
        words = mecab_library.Tokens(sents[0])
        texts = classifyPos(words, pos_list)
        for i in range(1,len(sents)):
            w = mecab_library.Tokens(sents[i])
            texts += classifyPos(w)
            print len(texts)
        return texts

    else:
        words = mecab_library.Tokens(sent)
        texts = classifyPos(words)
        return texts
```

表 1 - No.16 mecab_library.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import MeCab

#mecab = MeCab.Tagger('-Ochasen')
mecab = MeCab.Tagger()

class Tokens(object):
    """text の形態素情報を保持"""
    def __init__(self, text):
        self.text = text
        #print mecab.parse(text)
        node = mecab.parseToNode(text)
        self.tokens = []
        while node:
            self.tokens.append(Token(node.surface, *node.feature.split(',')))
            node = node.next


class Token(object):
    """形態素情報"""
    def __init__(self, surface, *args):
        # Mecab Format
        # 表層形¥t 品詞,品詞細分類 1,品詞細分類 2,品詞細分類 3,活用形,活用型,原形,読み,発音
        self.surface = surface   # 表層形
        try:
            self.pos = args[0]   # 品詞
```

```
                self.pos_detail1 = args[1]   # 品詞細分類^Z1
                self.pos_detail2 = args[2]   # 品詞細分類^Z2
                self.pos_detail3 = args[3]   # 品詞細分類^Z3
                self.verb_form = args[4]      # 活用形
                self.verb_type = args[5]      # 活用型
                self.basic = args[6]      # 原型
                self.reading = args[7]   # 読み
                self.pronunciation = args[8]      # 発音
                self.type = True # 全ての要素が格納できたとき
            except IndexError:
                self.type = False # 全ての要素が格納できなかったとき
```

表 1 - No.17 generatCard_tri_zz.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys

write = sys.stdout.write

c = 1

#write('egrep -v "'),

write('ggrep -v -E "'),
for line in sys.stdin:
    if c != 1:
        write('|')
    c = c + 1
    items = line.split()
    write(items[0]),
    write(".*"),
    write(items[1]),
    write(".*"),
    write(items[2]),

write("")
#write('')
```

表 1 - No.18 checkAlgorithm_V2_1_x.sh

```
#!/usr/bin/env bash

### フィルタ適用前
#全数カウント（西多摩郡、八王子市、世田谷区、千代田区、江戸川区、大田区、練馬区、青梅市、府中市）

#cat 2014_6_tokyo_rain.data | ggrep -E $1 | wc | cut -d" " -f1-8 > cnt;
#cat corpus_test.data | ggrep -E $1 | wc | cut -d" " -f1-8 > cnt;
cat corpus_test.data | wc | cut -d" " -f1-8 > cnt;


#正解数カウント
#cat 2014_6_tokyo_rain.data | ggrep -E $1 | python rainProc_V2.py $2% | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;
#cat corpus_test.data | ggrep -E $1 | python rainProc_V2.py $2% | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;
cat corpus_test.data | python rainProc_V2x.py | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;

###フィルタ適用後
temp="cat ggg";
temp2="cat qqq";


#cat 2014_6_tokyo_rain.data | ggrep -E $1 | `eval ${temp}` | python rainProc_V2.py $2% > temp_a;
#cat corpus_test.data | ggrep -E $1 | `eval ${temp}` | python rainProc_V2.py $2% > temp_a;
cat corpus_test.data | `eval ${temp}` | `eval ${temp2}` | python rainProc_V2x.py > temp_a;

#抽出数カウント
cat temp_a | wc | cut -d" " -f1-8 >> cnt;

#抽出後の正解数カウント
cat temp_a | ggrep -E "◎" | wc | cut -d" " -f1-8 >> cnt;

python evaluateFilter_V2.py $2
```

表 1 - No.19 rainProc_V2x.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import sqlite3
import datetime
import rainCheck

argvs = sys.argv


for line in sys.stdin:
	#print line
	if line.find('東京都西多摩郡') > -1 :
		temp_a = "西多摩郡%"
	elif line.find('東京都八王子市') > -1 :
		temp_a = "八王子市%"
	elif line.find('東京都世田谷区') > -1 :
		temp_a = "世田谷区%"
	elif line.find('東京都千代田区') > -1 :
		temp_a = "千代田区%"
	elif line.find('東京都江戸川区') > -1 :
		temp_a = "江戸川区%"
	elif line.find('東京都大田区') > -1 :
		temp_a = "大田区%"
	elif line.find('東京都青梅市') > -1 :
		temp_a = "青梅市%"
	elif line.find('東京都練馬区') > -1 :
		temp_a = "練馬区%"
	elif line.find('東京都府中市') > -1 :
		temp_a = "府中市%"
	else:
		temp_a = ""

	#print temp_a
	items = line.split("T",2)

	#D1 = items[0].strip()
	D1t = items[0].split(" ",2)
	D1 = D1t[2].strip()
	D2 = items[1].split(":",2)

	dd1 = int(D2[0])+1
	if dd1 == 24:
		# 23:00 + 1:00 = 24:00 となった場合
		# 日付を変更する操作を実施
		# 文字列の日付データを、datetime 型に変換する
		dd3 = datetime.timedelta(days=1) + datetime.datetime.strptime(D1, '%Y-%m-%d')
		D1 = dd3.strftime('%Y-%m-%d')

		# さらに、24:00 ではなく、00:00 に変更する
		dd1 = 0

	temp = "%sT%02d:00:00" % (D1,dd1)
	time = """ + temp + """
	print time,
	rc = rainCheck.rainCheck()
	#print rc.getRainCheck(argvs[1], time)
	print rc.getRainCheck(temp_a, time)
```

表 1 - No.20 evaluateFilter_V2.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import math
argvs = sys.argv
write = sys.stdout.write
flag = "N"

f = open("cnt")
```

```
lines2 = f.readlines() # 1 行毎にファイル終端まで全て読む(改行文字も含まれる)

#print lines2

write(argvs[1])
write(",")

t1 = lines2[0].strip()
write(t1)
write(",")
if t1 == "0":
	flag = "Y"

t2 = lines2[1].strip()
write(t2)
write(",")
if t2 == "0":
	flag = "Y"

t3 = lines2[2].strip()
write(t3)
write(",")
if t3 == "0":
	flag = "Y"

t4 = lines2[3].strip()
#t4_t1 = lines2[3].strip()
#t4_t2 = lines2[4].strip()
#t4 = int(t4_t1) - int(t4_t2)

write(str(t4))
write(",")

if flag == "N":
	t5 = round(float(t2) / float(t1),3)
	write(str(t5))
else:
	write("-")
write(",")

#適合率（正解率）
t6 = 0
if flag == "N":
	t6 = round(float(t4) / float(t3),3)
	write(str(t6))
else:
	write("-")
write(",")

#再現率（抽出後の正解数 / 全正解数）
t7 = 0
if flag == "N":
	t7 = round(float(t4) / float(t2),3)
	write(str(t7))
else:
	write("-")
write(",")

#F 値　( 2 * 適合率 * 再現率 / (適合率 + 再現率) )
if (t6 + t7) == 0:
	flag = "Y"

if flag == "N":
	t8 = round(2 * float(t6) * float(t7) / (float(t6) + float(t7)) ,3)
	write(str(t8))
else:
	write("-")


print
```

表 1 - No.21 makeDistanceEvaluationCard_1.py

```
#!/usr/bin/python
# coding: UTF-8
import sys

write = sys.stdout.write
argvs = sys.argv

#allLines = open('date.data').read()
#f = open('distanceCard.data', 'w') # 書き込みモードで開く

f = open('date.data', 'r')

#ファイルをすべて読み込む
list = f.readlines()

for line in list:
    for i in xrange(24):
        #両端の空白行削除, ゼロパディング:zfill
        for j in xrange(6):
            write(line.strip() + "T" + str(i).zfill(2) + ":" + str(j) ),
            print ""

f.close() # ファイルを閉じる
```

表 1 - No.22 makeDistanceEvaluation_1.py

```
#!/usr/bin/python
# coding: UTF-8
import sys

write = sys.stdout.write
argvs = sys.argv

print "#!/usr/bin/env bash"

for line in sys.stdin:
    print './getEvent_xx.sh',argvs[1],"",
    write(line.strip() + ""),
    print ' "雨"',"2","3000","2",'|', 'python export_3.py | ggrep -E ',argvs[2]
```

表 1 - No.23 t_xxxxxxx.sh

```
#!/usr/bin/env bash
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T00:0" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T00:1" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T00:2" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T00:3" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T00:4" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T00:5" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T01:0" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T01:1" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T01:2" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T01:3" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T01:4" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T01:5" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
```

```
./getEvent_xx.sh corpus_test_tmp_10.data "2014-06-01T02:0" "雨" "2" "3000" "2" | python export_3.py | ggrep -E　東京都大田区
（以降略）
```

表 1 - No.24 getEvent_xx.sh

```
#!/usr/bin/env bash
cat $1 | ggrep -E "$2" | ggrep -E "$3" | sort | uniq | cut -d' ' -f1 | uniq -c | sort | python ../omitID.py $4 > omitData.out;
if [ -s omitData.out ]; then #  ファイルサイズが 0 より大きいとき
	cat $1 | ggrep -E "$2" | ggrep -E "$3" | eval `cat omitData.out` | python ../testGetDistance_3xx.py $5 $6 | sort -k2,3 | uniq | sort -k3,3;
else #  ファイルサイズが 0 のとき
	cat $1 | ggrep -E "$2" | ggrep -E "$3" | python ../testGetDistance_3xx.py $5 $6 | sort -k2,3 | uniq | sort -k3,3;
fi
```

表 1 - No.25 omitID.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys

argvs = sys.argv
write = sys.stdout.write
data = []
i = 0
j = 0

for line in sys.stdin:
   data.append(line.strip())

lc = len(data)

#print lc

#write('ggrep -v -E "')

while lc > 0:
   #print i
   tempx = data[i].split(" ",1) #print items
   if int(tempx[0]) >= int(argvs[1]):
      if j == 0:
         write('ggrep -v -E "')
         j = j + 1
      write(tempx[1].strip())
      if (lc-1) > 0:
         write("|")

   i = i + 1
   lc = lc - 1

if j != 0:
   print("")
```

表 1 - No.26 testGetDistance_3xx.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import datetime
import commands
import geoDistance

argvs = sys.argv

data = []
data_1 = []
data_f = []

i = 0
c = 0
f = "N"

gd = geoDistance.GeoDistance()
```

```
for line in sys.stdin:
   data.append(line)

lc = len(data)
lcx = lc


while lc > 1:
   #print i
   items_c = data[i].split(" ",5) #print items
   #print items_c[1],items_c[2]
   data_f.append(data[i])

   c = i

   while lcx > 1:
      c = c + 1
      #print c
      items_n = data[c].split(" ",5) #print items

      #print items_c[2],items_c[1],items_n[2],items_n[1]
      #print items_c[4],items_c[3]

      if int(argvs[1]) > gd.getGeoDistance(items_c[4],items_c[3],items_n[4],items_n[3]):
         data_f.append(data[c])

      if len(data_f) >= int(argvs[2]):
            for temp in data_f:
               print temp,
            data_f = []

      lcx = lcx - 1

   i = i + 1
   lc = lc - 1
   lcx = lc
   data_f = []

"""



   lati = items[1].strip()
   longi = items[2].strip()
   tempx = rb.getAddress(lati,longi) if tempx != None:
   print items[0].strip(),
   print tempx.encode('utf-8'), print items[3].strip()

lon1 = 139.764885
lat1 = 35.68136
lon2 = 139.810461
lat2 = 36.709499

data = [[lon1,lat1],[lon2,lat2]]
data.append([149.764885,35.68136])

print data
print data[0][0], data[0][1]
print data[1][0], data[1][1]
print data[2][0], data[2][1]


gd = geoDistance.GeoDistance()

print gd.getGeoDistance(data[0][0],data[0][1],data[1][0],data[1][1])
print gd.getGeoDistance(data[1][0],data[1][1],data[2][0],data[2][1])

#print gd.getGeoDistance(data[0][0],data[1][1],)
"""
```

表 1 - No.27 checkAlgorithm_V2_2_x.sh

```
#!/usr/bin/env bash

### フィルタ適用前
#全数カウント（西多摩郡、八王子市、世田谷区、千代田区、江戸川区、大田区、練馬区、青梅市、府中市）

#cat corpus_test.data | ggrep -E $1 | wc | cut -d" " -f1-8 > cnt;
cat corpus_test.data | wc | cut -d" " -f1-8 > cnt;


#正解数カウント
#cat corpus_test.data | ggrep -E $1 | python rainProc_V2.py $2% | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;
cat corpus_test.data | python rainProc_V2x.py | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;


###近距離フィルタ適用後

#./xxxxx.sh | python rainProc_V2.py $2% > temp_a;
#cat $3 | python rainProc_V2.py $2% > temp_a;
cat $1 | python rainProc_V2x.py > temp_a;

#抽出数カウント
cat temp_a | wc | cut -d" " -f1-8 >> cnt;

#抽出後の正解数カウント
cat temp_a | ggrep -E "◎" | wc | cut -d" " -f1-8 >> cnt;

python evaluateFilter_V2.py $2
```

表 1 - No.28 checkAlgorithm_V2_3_x.sh

```
#!/usr/bin/env bash

### フィルタ適用前
#全数カウント（西多摩郡、八王子市、世田谷区、千代田区、江戸川区、大田区、練馬区、青梅市、府中市）

#cat corpus_test.data | ggrep -E $1 | wc | cut -d" " -f1-8 > cnt;
cat corpus_test.data | wc | cut -d" " -f1-8 > cnt;


#正解数カウント
#cat corpus_test.data | ggrep -E $1 | python rainProc_V2.py $2% | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;
cat corpus_test.data | python rainProc_V2x.py | ggrep "◎" | wc | cut -d" " -f1-8   >> cnt;

temp="cat ggg";
temp2="cat qqq";

###近距離フィルタ適用
###ノイズフィルタ適用
#./xxxxx.sh | ggrep -E $1 | `eval ${temp}` | python rainProc_V2.py $2% > temp_a;
#cat $3 | ggrep -E $1 | `eval ${temp}` | python rainProc_V2.py $2% > temp_a;
cat $1 | `eval ${temp}` | `eval ${temp2}` | python rainProc_V2x.py > temp_a;

#抽出数カウント
cat temp_a | wc | cut -d" " -f1-8 >> cnt;

#抽出後の正解数カウント
cat temp_a | ggrep -E "◎" | wc | cut -d" " -f1-8 >> cnt;

python evaluateFilter_V2.py $2
```

表 1 - No.29 loadMLData.py

```
#!/usr/bin/env python
# coding: utf-8
import os
import sqlite3
import unicodecsv

# データベースに接続
db_filename = 'ml_rain_address.sqlite3'
conn = sqlite3.connect(db_filename)
```

```
# 2015/6 月分　CSV ファイルを読み込む
#csvfile = "降水量_6 月_hour_utf8.csv"

# 2014/6 月分　CSV ファイルを読み込む
#csvfile = "2014_6_data_f.csv"

# 2014/12 月分　CSV ファイルを読み込む
csvfile = "2014_12_data_f.csv"

reader = unicodecsv.reader(file(csvfile, 'r'))

c = 0
i = 0
x = 0

row_x = ""
sql = ""

# データベースに登録する

for row in reader:
    if c == 0:
        row_x = row
        c = c + 1
    else:

        # 観測地点名
        m_name = row[x]

        for temp in row_x:

            if i > 0:

                # 観測地点名
#                 print name,

                # 最大降水量
                x = x + 1
                # row[x]
                amount = row[x]

                # 日付 temp

                sql = 'insert into ml_t_station values("%s", "%s", "%s")'    % (m_name, amount, temp)
                print sql
                conn.execute(sql)

            i = i + 1

        i = 0
        x = 0
    c = c + 1

conn.commit()
conn.close()
```

表 1 - No.31 generateMapzz_xx.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys

argvs = sys.argv
i_flag = argvs[1]

print

if i_flag == "1":
    icon = 'http://chart.apis.google.com/chart?chst=d_map_spin&chld=0.25||04122f|13||'
elif i_flag == "2":
    icon = 'http://chart.apis.google.com/chart?chst=d_map_spin&chld=0.25||0d368c|13||'
elif i_flag == "3":
```

```
        icon = 'http://chart.apis.google.com/chart?chst=d_map_spin&chld=0.25||165be9|13||'
elif i_flag == "4":
        icon = 'http://chart.apis.google.com/chart?chst=d_map_spin&chld=0.25||739cf2|13||'
else:
        icon = 'http://chart.apis.google.com/chart?chst=d_map_spin&chld=0.25||d0defb|13||'


for line in sys.stdin:
        items = line.split(" ",6)
        lati = items[3]
        longi = items[4]
        data = items[5]

        print "var m_latlng1 = new google.maps.LatLng(",
        print lati,
        print ",",
        print longi,
        print ")"

        print "var marker1 = new google.maps.Marker({"
        print " position: m_latlng1,"
        print " animation: google.maps.Animation.DROP,"

        print " map: map,"

        print " icon: '%s'" %icon
        print "});"
```

表 2 - No.1 omitStopWord.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import re

def format_text(text,parm):
	'''
	MeCab に入れる前のツイートの整形方法例
	'''

	text=re.sub(r'https?://[¥w/:%#¥$&¥?¥(¥)~¥.=¥+¥-…]+', "", text)
	text=re.sub('RT', "", text)
	#text=re.sub(r'[!-~]', "", text)#半角記号,数字,英字
	text=re.sub(r'[：-@]', "", text)#全角記号
	text=re.sub('¥n', " ", text)#改行文字
	text=re.sub(parm, " ", text)#word

	return text


argvs = sys.argv

# 空白行を出力しないようにするため、print でなく、write を使用
write = sys.stdout.write

for line in sys.stdin:
    print(format_text(line,argvs[1]))
```

表 2 - No.3 omitCard_2.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
import re

argvs = sys.argv



# 空白行を出力しないようにするため、print でなく、write を使用
write = sys.stdout.write

print("#!/usr/bin/env bash")
print("")

c = 1

#除去カード作成 !で除去
write("gawk -F" ' '!($5$6$7$8$9$10$11$12$13$14$15$16$17$18$19$20$21$22$23$24$25$26$27$28$29$30 ~ /"),

for line in sys.stdin:
	if c != 1:
		write('|')
	c = c + 1

	s = line.split(" ")
	write(s[1].strip()),

write("/)"")
```

表 2 - No.4 run.sh

```
#!/usr/bin/env bash

gawk -F' ' '!($5$6$7$8$9$10$11$12$13$14$15$16$17$18$19$20$21$22$23$24$25$26$27$28$29$30 ~ /震源.*熊本県.*熊本.*地方|地方.*さ.*10.*km|頃.*熊本県.*熊本.*地方|10.*km.*EARTH.*Quake|さ.*10.*km.*EARTH|熊本県.*熊本.*地方.*地震|詳細.*2016.*04.*16|推定.*詳細.*2016.*04|熊本県.*熊本.*地方.*N3|熊本県.*熊本.*地方.*さ|地方.*地震.*最大.*震度|熊本.*地方.*N3.*2|震源.*さ.*10.*km|00.*2016.*04.*16|分.*頃.*熊本県.*熊本|熊本.*地方.*さ.*10|地震.*発生.*震源.*さ|km.*2016.*04.*16|熊本.*地方.*地震.*最大|報.*熊本県.*熊本.*地方|2016.*04.*16.*01|2016.*04.*16.*04|37.*00.*2016.*04|頃.*熊本県.*阿蘇.*地方|熊本県.*熊本.*地方.*10|熊本県.*北西.*部.*2016|部.*2016.*04.*16|北西.*部.*2016.*04|10.*km.*地震.*規模|報.*震源.*熊本県.*熊本|発生.*震源.*さ.*10|分.*頃.*熊本県.*阿蘇|saigai.*jishin.*EARTH.*Quake|分.*気象庁.*発表.*16 日|10.*km.*2016.*04|さ.*10.*km.*2016|2016.*04.*16.*03|km.*EARTH.*Quake.*緊急地震速報|2016.*04.*16.*08|km.*地震.*規模.*マグニチュード|震源.*熊本県.*阿蘇.*地方|2016.*04.*16.*14|2016.*04.*16.*16|JST 株式会社.*日本.*熊本県.*熊本市|震源地.*熊本県.*熊本.*地方|N3.*2.*8.*E|地方.*N3.*2.*8|2.*8.*E.*130|熊本県.*阿蘇.*地方.*N3|熊本県.*阿蘇.*地方.*さ|4.*推定.*詳細.*2016|最大.*震度.*4.*推定|震度.*4.*推定.*詳細|地震.*最大.*震度.*4|8.*熊本県.*熊本.*地方|33.*0.*N.*131|NW.*KUMAMOTO.*PREF.*32|熊本県.*北西.*部.*NW|部.*NW.*KUMAMOTO.*PREF|北西.*部.*NW.*KUMAMOTO|2016.*04.*16.*09|N.*130.*8.*E|地震.*津波.*心配.*地震|32.*8.*N.*130|さ.*10.*km.*地震|推定.*地震.*津波.*心配|地震.*規模.*マグニチュード.*3|3.*推定.*詳細.*2016|N3.*2.*7.*E|頃.*発生.*G.*気象庁|最大.*震度.*3.*推定|震度.*3.*推定.*詳細|地震.*最大.*震度.*3|地方.*N3.*2.*7|分.*頃.*発生.*G|130.*8.*10.*km|2016.*04.*16.*11|E.*130.*8.*10|130.*8.*熊本県.*熊本|3.*さ.*10.*km|熊本県.*熊本.*地方.*エムスリー株式会社|周辺.*地震.*発生.*模様|地方.*周辺.*地震.*発生|2016.*04.*16.*15|N.*130.*7.*E|阿蘇.*地方.*さ.*10|2016.*04.*16.*07|時刻.*2016.*04.*16|発生.*時刻.*2016.*04|2.*7.*E.*130|jishin.*EARTH.*Quake.*eqjp|阿蘇.*地方.*N3.*3|2016.*04.*16.*13|熊本.*地方.*10.*4|熊本県.*阿蘇.*地方.*地震|4.*さ.*10.*km|0.*緊急地震速報.*予報.*ID|最大.*震度.*5.*弱|0.*0.*緊急地震速報.*予報|8.*E.*130.*8|32.*7.*N.*130|頃.*発生.*震源地.*熊本県|報.*2016.*4.*16|E.*10.*km.*地震|N.*131.*1.*E|さ.*10.*km.*MAP|熊本県.*熊本.*地方.*32|分.*熊本県.*熊本.*地方|JST 株式会社.*震源地.*日本.*熊本県|分.*震源.*熊本県.*熊本|さ.*10.*km.*エムスリー株式会社|震源.*地下.*10.*km|地震.*津波.*心配.*jishin|3.*0.*0.*緊急地震速報|4.*0.*0.*緊急地震速報|km.*saigai.*jishin.*EARTH|2016.*04.*16.*05|2016.*04.*16.*21|5.*さ.*10.*km|JST 株式会社.*日本.*熊本県.*益城町|熊本.*地方.*北緯.*32|熊本県.*熊本.*地方.*北緯|地震.*地震.*津波.*心配|度.*さ.*10.*km|2016.*04.*16.*10|8.*N.*130.*8|04.*16.*01.*44|熊本県.*北東部.*NE.*KUMAMOTO|震源地.*熊本県.*阿蘇.*地方|北東部.*NE.*KUMAMOTO.*PREF|10.*km.*saigai.*jishin|地震.*最大.*震度.*5|気象庁.*発表.*熊本県.*熊本|地震.*発生.*最大.*震度 3|発表.*熊本県.*熊本.*地方|10.*km.*最大.*予測|2016.*04.*16.*17|2016.*04.*16.*18|さ.*10.*km.*最大|7.*N.*130.*7|JST 株式会社.*日本.*熊本県.*宇城市|LT.*震源.*GT.*北緯|さ.*10.*km.*saigai|さ.*10.*km.*規模|地方.*2016.*04.*16|04.*16.*09.*48|10.*km.*jishin.*EARTH|10.*km.*M.*5|32.*8.*130.*8|8.*130.*8.*熊本県|km.*jishin.*EARTH.*Quake|さ.*10.*km.*jishin|熊本県.*熊本.*地方.*M4|心配.*LT.*震源.*GT|推定.*津波.*心配.*LT|津波.*心配.*LT.*震源|7.*E.*130.*7|阿蘇.*地方.*地震.*最大|弱.*推定.*詳細.*2016|度.*東経.*130.*7|東経.*130.*7.*度|報.*熊本県.*阿蘇.*地方|10.*km.*未満.*saigai|2016.*04.*16.*20|km.*未満.*saigai.*jishin|最大.*震度 3.*気象庁.*発表|震度 3.*気象庁.*発表.*熊本県|推定.*地震.*発生.*震源|未満.*saigai.*jishin.*EARTH|0.*N.*131.*1|04.*16.*08.*20|10.*km.*推定.*規模|N.*131.*2.*E|NE.*KUMAMOTO.*PREF.*33|さ.*10.*km.*推定|地震.*04.*月.*16 日|地方.*震源.*さ.*10|発生.*震源.*さ.*推定|7.*熊本県.*熊本.*地方|EARTH.*Quake.*地震.*強震モニタ|km.*EARTH.*Quake.*地震|KUMAMOTO.*PREF.*33.*0|PREF.*33.*0.*N|強震モニタ.*地震.*発生.*時刻|熊本県.*阿蘇.*地方.*10|地震.*発生.*時刻.*2016|4.*37.*00.*2016|N3.*2.*6.*E|熊本県.*北東部.*2016.*04|最大.*震度.*4.*37|震度.*4.*37.*00|地方.*N3.*2.*6|発生.*震源地.*熊本県.*熊本|北東部.*2016.*04.*16|04.*16.*14.*27|04.*16.*16.*01|16.*09.*48.*32|N3.*3.*E.*131|阿蘇.*地方.*周辺.*地震|熊本.*地方.*震源.*さ|熊本県.*阿蘇.*地方.*周辺|熊本県.*熊本.*地方.*震源|地方.*N3.*3.*E|0.*N.*131.*2|10.*km.*地震.*津波|16.*16.*01.*59|5.*弱.*推定.*詳細|km.*地震.*津波.*心配|緊急地震速報.*最終.*報.*熊本県|震度.*5.*弱.*推定|地下.*10.*km.*地震|発生.*最大.*予測.*震度 3|2.*SA.*1015.*Y|EARTH.*Quake.*緊急地震速報.*1|Quake.*緊急地震速報.*1.*報|気象庁.*発表.*お待ち.*eqjp|最大.*震度 3.*震源.*地下|最大.*予測.*震度 3.*G|最大.*予測.*震度 4.*G|情報.*気象庁.*発表.*お待ち|心配.*地震.*jishin.*災害|正確.*情報.*気象庁.*発表|地震.*規模.*マグニチュード.*4|地震.*発生.*模様.*K|地震.*発生.*模様.*正確|津波.*心配.*地震.*jishin|発生.*最大.*震度 3.*震源|発生.*最大.*予測.*震度 4|発生.*模様.*K.*NET|発生.*模様.*正確.*情報|模様.*K.*NET.*強震モニタ|模様.*正確.*情報.*気象庁|予測.*震度 3.*G.*Y|予測.*震度 4.*G.*Y|130.*7.*10.*km|32.*7.*度.*東経|7.*度.*東経.*130|E.*130.*7.*10|N3.*3.*0.*E|震源.*GT.*北緯.*32|震源.*さ.*推定.*10|震度 3.*震源.*地下.*10|地方.*N3.*3.*0|度.*東経.*130.*8|東経.*130.*8.*度|分.*震源.*熊本県.*阿蘇|北緯.*32.*7.*度|130.*7.*熊本県.*熊本|さ.*推定.*10.*km|緊急地震速報.*3.*報.*熊本県|熊本.*地方.*周辺.*地震|熊本県.*熊本.*地方.*周辺|心配.*震度.*情報.*16 日|地震.*津波.*心配.*震度|地方.*地震.*震源.*さ|津波.*心配.*震度.*情報|04.*16.*03.*03|2016.*04.*15.*23|33.*度.*東経.*131|7.*さ.*10.*km|PREF.*32.*8.*N|ころ.*熊本県.*熊本.*地方|熊本.*地方.*10.*3|熊本.*地方.*10.*5|熊本.*地方.*地震.*震源|最大.*震度.*3.*緊急地震速報|最大.*震度 3.*地震.*地震|震度 3.*地震.*地震.*津波|地震.*震源.*さ.*10|分.*ころ.*熊本県.*熊本|北緯.*33.*度.*東経|0.*さ.*10.*km|2016.*04.*16.*02|8.*さ.*10.*km|Kumamoto.*jishin.*熊本.*地震|KUMAMOTO.*PREF.*32.*8|M4.*3.*さ.*10|さ.*10.*km.*未満|緊急地震速報.*1.*報.*熊本県|熊本.*揺れ.*計測.*震度|頃.*熊本.*揺れ.*計測|分.*頃.*熊本.*揺れ|報.*震源.*熊本県.*阿蘇|2.*6.*E.*130|3.*0.*E.*131|36.*00.*2016.*04|8.*10.*km.*M4|EARTH.*Quake.*eqjp.*確定|eqjp.*確定.*情報.*16 日|JST 株式会社.*日本.*熊本県.*阿蘇市|Quake.*eqjp.*確定.*情報|さ.*10.*km.*緊急地震速報|熊本県.*阿蘇.*地方.*M4|熊本県.*阿蘇.*地方.*エムスリー株式会社|熊本県.*阿蘇.*地方.*北緯|推定.*10.*km.*未満|地震.*発生.*予想.*最大|地震速報.*2016.*04.*16|発生.*予想.*最大.*震度|04.*16.*01.*25|04.*16.*18.*25|119.*8K.*g.*さ|16.*01.*44.*07|2016.*04.*16.*12|2016.*04.*16.*9|3.*報.*震源.*熊本県|6.*TNT.*119.*8K|9.*さ.*10.*km|M2.*6.*TNT.*119|TNT.*119.*8K.*g|さ.*10.*km.*マグニチュード|時.*57.*分.*頃|地震.*発生.*最大.*震度 4|地方.*地震.*発生.*最大|分.*熊本県.*阿蘇.*地方|予想.*発生.*時刻.*2016|揺れ.*計測.*震度.*2|04.*16.*04.*51|10.*km.*M4.*3|3.*TNT.*42.*5|9.*熊本県.*熊本.*地方|EARTH.*Quake.*緊急地震速報.*警報|緊急.*作業.*中.*震度|作業.*中.*震度.*分離|時刻.*別.*地震.*発生|心配.*地震.*時刻.*別|地震.*時刻.*別.*地震|地震.*発生.*緊急.*作業|中.*震度.*分離.*ため|津波.*心配.*地震.*時刻|発生.*緊急.*作業.*中|別.*地震.*発生.*緊急|0.*1.*緊急地震速報.*警報|04.*16.*08.*02|04.*16.*13.*15|1.*報.*震源.*熊本県|16.*01.*25.*05|16.*03.*03.*10|32.*8.*度.*東経|8.*度.*東
```

経.*130|km.*推定.*規模.*エムスリー株式会社|km.*地震.*規模.*エムスリー株式会社|N3.*2.*9.*E|阿蘇.*地方.*北緯.*33|熊本県.*熊本.*地方.*震度 3|地方.*N3.*2.*9|付近.*M2.*6.*TNT|報.*最終.*震源.*熊本県|北緯.*32.*8.*度|1.*報.*熊本県.*熊本|2.*報.*震源.*熊本県|km.*M4.*3.*最大|Oita.*jishin.*大分.*地震|緊急地震速報.*5.*報.*熊本県|熊本.*地方.*2016.*04|熊本県.*熊本.*地方.*2016|熊本県.*熊本.*地方.*推定|頃.*大分.*揺れ.*計測|最終.*報.*熊本県.*熊本|大分.*揺れ.*計測.*震度|地震.*jishin.*災害.*saigai|度.*東経.*131.*1|東経.*131.*1.*度|付近.*震度.*Mw.*4|分.*頃.*大分.*揺れ|04.*16.*05.*52|10.*km.*規模.*M4|130.*7.*度.*さ|3.*37.*00.*2016|3.*報.*熊本県.*熊本|4.*TNT.*60.*0|4.*報.*震源.*熊本県|42.*5.*kg.*さ|60.*0.*kg.*さ|M2.*3.*TNT.*42|M2.*4.*TNT.*60|TNT.*42.*5.*kg|TNT.*60.*0.*kg|緊急地震速報.*2.*報.*熊本県|緊急地震速報.*4.*報.*熊本県|最大.*震度.*3.*37|最大.*震度.*5.*強|最大.*震度 4.*気象庁.*発表|最大.*予測.*震度 5 弱.*G|震度.*3.*37.*00|震度 4.*気象庁.*発表.*熊本県|発生.*最大.*予測.*震度 5 弱|予測.*震度 5 弱.*G.*Y|04.*16.*21.*05|09.*48.*32.*発生|1.*報.*2016.*4|3.*地震.*発生.*震源|5.*TNT.*84.*8K|5.*弱.*0.*1|57.*分.*頃.*熊本県|6.*さ.*10.*km|8.*TNT.*239.*0|84.*8K.*g.*さ|jishin.*EARTH.*Quake.*緊急地震速報|jishin.*EARTH.*Quake.*地震情報|JST 株式会社.*日本.*熊本県.*嘉島町|km.*上越市.*影響.*もの|M2.*5.*TNT.*84|M4.*3.*最大.*震度|TNT.*84.*8K.*g|阿蘇.*地方.*33.*0|熊本県.*阿蘇.*地方.*33|熊本県.*天草.*芦北.*震度 1|熊本県.*南部.*2016.*04|頃.*地震.*震源地.*熊本県|最大.*震度.*4.*緊急地震速報|上越市.*影響.*もの.*予想|地方.*32.*8.*N|地方.*33.*0.*N|南部.*2016.*04.*16|付近.*M2.*5.*TNT|分.*頃.*地震.*震源地|04.*16.*01.*45|04.*16.*20.*44|10.*km.*震度.*不明|130.*8.*E.*10|16.*01.*45.*55|3.*熊本県.*熊本.*地方|32.*7.*130.*7|5.*4.*さ.*10|5.*報.*熊本県.*熊本|7.*130.*7.*熊本県|7.*E.*130.*8|EARTH.*Quake.*37.*00|JST 株式会社.*日本.*熊本県.*御船町|M.*5.*3.*さ|M.*5.*4.*さ|M4.*4.*さ.*10|Quake.*37.*00.*2016|さ.*10.*km.*M4|気象庁.*発表.*熊本県.*阿蘇|弱.*0.*1.*緊急地震速報|地方.*10.*4.*3|度.*震度 3.*熊本県.*熊本|発生.*M4.*3.*さ|発表.*熊本県.*阿蘇.*地方|0.*TNT.*476.*9|04.*16.*04.*23|04.*16.*21.*44|130.*9.*熊本県.*熊本|131.*1.*10.*km|16.*01.*59.*発生|2.*TNT.*30.*1|2016.*04.*16.*0|2016.*04.*16.*5|2016.*04.*16.*8|5.*3.*さ.*10|5.*報.*震源.*熊本県|8.*10.*km.*エムスリー株式会社|E.*131.*1.*10|eqjp.*2.*SA.*1015|お待ち.*eqjp.*2.*SA|熊本.*地方.*32.*8|熊本.*地方.*震度 3.*エムスリー株式会社|推定.*諏訪地域.*揺れ.*観測|発生.*震源地.*熊本県.*阿蘇|発表.*お待ち.*eqjp.*2|10.*km.*MAP.*緊急地震速報|10.*km.*震度.*3|130.*9.*10.*km|3.*E.*131.*1|7.*TNT.*169.*2|8.*TNT.*7.*6|9.*TNT.*337.*6|E.*130.*9.*10|JST 株式会社.*日本.*熊本県.*菊陽町|km.*最大.*予測.*震度 3|阿蘇.*地方.*震源.*さ|熊本.*地方.*地震.*発生|熊本県.*阿蘇.*地方.*震源|震度.*情報.*16 日.*04|震度 4.*G.*Y.*緊急地震速報|諏訪地域.*揺れ.*観測.*震度|地震.*津波.*心配.*地震情報|付近.*M2.*3.*TNT|付近.*震度.*M.*W3|04.*16.*04.*05|04.*16.*11.*02|1.*緊急地震速報.*警報.*ID|130.*8.*度.*さ|16.*08.*20.*41|16.*11.*02.*51|3.*最大.*震度.*4|8.*10.*km.*M|AQUA.*REAL.*G.*Y|EARTH.*Quake.*2016.*04|EARTH.*Quake.*緊急地震速報.*3|km.*最大.*予測.*震度 4|M4.*5.*さ.*10|Quake.*2016.*04.*16|Quake.*緊急地震速報.*3.*報|エムスリー株式会社.*8.*さ.*10|阿蘇.*地方.*2016.*04|観測.*地域.*詳細.*気象庁|熊本.*地方.*32.*7|熊本県.*阿蘇.*地方.*2016|地域.*詳細.*気象庁.*情報|地方.*32.*7.*N|地方.*北緯.*33.*度|付近.*M2.*4.*TNT|04.*16.*04.*09|04.*16.*04.*15|04.*16.*07.*23|10.*km.*M4.*5|10.*km.*エムスリー株式会社.*5|10.*km.*エムスリー株式会社.*8|10.*km.*未満.*jishin|131.*1.*度.*さ|169.*2.*kg.*さ|2016.*04.*16.*06|2016.*04.*16.*6|4.*熊本県.*熊本.*地方|5.*熊本県.*熊本.*地方|8.*E.*10.*km|E.*5.*0.*km|EARTH.*Quake.*緊急地震速報.*予報|km.*未満.*jishin.*EARTH|M2.*7.*TNT.*169|N.*130.*9.*E|Quake.*緊急地震速報.*予報.*ID|TNT.*169.*2.*kg|最終.*震源.*熊本県.*熊本|最大.*震度.*震度 3.*jishin|最大.*震度 4.*震源.*地下|最大.*震度 4.*地震.*地震|震度.*震度 3.*jishin.*EARTH|震度 3.*jishin.*EARTH.*Quake|震度 4.*地震.*地震.*津波|震度 5 弱.*G.*Y.*緊急地震速報|速報.*LV.*1.*16 日|地震.*津波.*心配.*16 日|地方.*北緯.*32.*7|地方.*北緯.*32.*8|発生.*G.*気象庁.*地震情報|発生.*最大.*震度 4.*震源|未満.*jishin.*EARTH.*Quake|予想.*最大.*震度.*震度 3|0.*km.*2016.*04|04.*16.*03.*55|10.*km.*震度.*4|131.*0.*10.*km|16.*13.*15.*28|2.*報.*熊本県.*熊本|2016.*04.*16.*1|239.*0.*kg.*さ|32.*6.*N.*130|32.*9.*度.*東経|337.*6.*kg.*さ|4.*報.*熊本県.*熊本|7.*10.*km.*M4|8.*地震.*発生.*震源|E.*130.*8.*最大|E.*131.*0.*10|km.*EARTH.*Quake.*37|M2.*8.*TNT.*239|M2.*9.*TNT.*337|M4.*0.*さ.*10|TNT.*239.*0.*kg|TNT.*337.*6.*kg|エムスリー株式会社.*5.*さ.*10|熊本県.*熊本.*震度 1.*熊本県|震度.*不明.*気象庁.*震源|震度 4.*震源.*地下.*10|速報.*LV.*3.*16 日|地下.*10.*km.*未満|発生.*M.*5.*3|不明.*気象庁.*震源.*情報|付近.*M2.*7.*TNT|北緯.*32.*9.*度|揺れ.*計測.*震度.*3|01.*44.*07.*発生|04.*16.*14.*03|04.*16.*15.*38|04.*月.*16 日.*01|04.*月.*16 日.*03|1.*TNT.*21.*3K|1.*TNT.*673.*6|10.*km.*エムスリー株式会社.*9|130.*7.*E.*10|16.*04.*05.*49|16.*04.*23.*18|16.*05.*52.*51|16.*07.*23.*54|16.*14.*27.*01|18.*24 時.*%.*天気|2.*9.*E.*130|2.*熊本県.*阿蘇.*地方|21.*3K.*g.*さ|3.*E.*131.*2|30.*1.*kg.*さ|4.*TNT.*1.*9|4.*最大.*震度.*4|5.*TNT.*2.*7|5.*地震.*発生.*震源|7.*E.*10.*km|7.*度.*さ.*10|E.*130.*7.*最大|E.*131.*1.*最大|EARTH.*Quake.*緊急地震速報.*最終|jishin.*大分.*地震.*04|km.*震度.*不明.*気象庁|KUMAMOTO.*PREF.*32.*7|m.*2016.*04.*16|M2.*1.*TNT.*21|M2.*2.*TNT.*30|PREF.*32.*7.*N|SA.*1015.*Y.*M4|TNT.*21.*3K.*g|TNT.*30.*1.*kg|エムスリー株式会社.*3.*地震.*発生|エムスリー株式会社.*6.*地震.*発生|エムスリー株式会社.*7.*地震.*発生|エムスリー株式会社.*8.*地震.*発生|さ.*5.*0.*km|阿蘇.*地方.*10.*4|熊本.*地震.*04.*月|熊本県.*熊本.*震度 2.*熊本県|月.*16 日.*01.*時|月.*16 日.*03.*時|時.*16.*分.*頃|大津波警報.*津波警報.*津波注意報.*発表|大分.*地震.*04.*月|津波警報.*津波注意報.*発表.*中|津波警報.*等.*大津波警報.*津波警報|等.*大津波警報.*津波警報.*津波注意報|発生.*M.*5.*4|発生.*M4.*5.*さ|付近.*M2.*9.*TNT|揺れ.*計測.*震度.*1|0.*131.*2.*熊本県|0.*E.*131.*2|04.*16.*01.*30|04.*16.*02.*04|04.*16.*10.*38|10.*4.*3.*4|10.*km.*エムスリー株式会社.*7|10.*km.*未満.*地震|131.*1.*E.*10|131.*2.*10.*km|131.*2.*熊本県.*阿蘇|16.*04.*09.*26|16.*15.*38.*38|3.*4.*0.*0|3.*推定.*地震.*津波|33.*0.*131.*2|4.*3.*4.*0|4.*4.*0.*0|5.*最大.*震度.*4|6.*E.*130.*7|7.*10.*km.*エムスリー株式会社|E.*0.*1.*km|E.*130.*7.*推定|E.*131.*2.*10|GT.*北緯.*32.*7|jishin.*熊本.*地震.*04|km.*M.*5.*4|km.*M4.*2.*AQUA|km.*M4.*5.*最大|km.*未満.*地震.*津波|M4.*2.*AQUA.*REAL|阿蘇.*地方.*N3.*2|観測.*震度.*観測.*地域|気象庁.*発表.*16 日.*04|熊本.*地方.*推定.*震度 3|時.*48.*分.*頃|推定.*震度 3.*熊本県.*熊本|地震.*震源地.*熊本県.*熊本|地方.*さ.*20K.*m|度.*さ.*推定.*規模|度.*東経.*131.*2|東経.*131.*2.*度|発生.*M4.*4.*さ|発表.*16 日.*04.*時|未満.*地震.*津波.*心配|揺れ.*観測.*震度.*観測|℃.*0.*6.*時|0.*131.*1.*熊本県|0.*E.*131.*1|04.*16.*08.*08|1.*E.*10.*km|1.*熊本県.*阿蘇.*地方|10.*4.*4.*4|10.*4.*5.*4|10.*km.*M4.*4|10.*km.*エムスリー株式会社.*4|10.*km.*マグニチュード.*3|131.*1.*熊本県.*阿蘇|16.*21.*05.*06|2016.*04.*15.*22|2016.*04.*16.*7|3.*10.*km.*震度|32.*9.*N.*130|32.*発

生.*M.*5|33.*0.*131.*1|4.*4.*4.*0|4.*5.*4.*0|48.*32.*発生.*M|5.*4.*0.*0|6.*地震.*発生.*震源|8.*E.*130.*9|8.*度.*
さ.*10|9.*TNT.*10.*7|AQUA.*REAL.*2016.*04|G.*Y.*37.*00|km.*M.*5.*3|km.*M4.*4.*最大|km.*エムスリー株式会
社.*9.*最大|M.*5.*3.*熊本県|M.*5.*3.*最大|M4.*4.*最大.*震度|M4.*5.*最大.*震度|REAL.*2016.*04.*16|SA.*1015.*Y.*エ
ムスリー株式会社|Y.*37.*00.*2016|エムスリー株式会社.*7.*さ.*10|エムスリー株式会社.*9.*地震.*発生|ころ.*熊本県.*阿
蘇.*地方|阿蘇.*地方.*地震.*震源|緊急地震速報.*1.*報.*エムスリー株式会社|熊本.*地方.*さ.*エムスリー株式会社|熊本県.*阿
蘇.*地方.*震度 3|時.*15.*分.*頃|時.*34.*分.*頃|時.*44.*分.*頃|震源.*さ.*20K.*m|震度.*観測.*地域.*詳細|震度 1.*熊本県.*
天草.*芦北|震度 1.*福岡県.*筑後.*震度 1|地方.*10.*4.*4|地方.*10.*4.*5|地方.*推定.*震度 3.*エムスリー株式会社|津波.*心
配.*jishin.*地震情報|付近.*M2.*2.*TNT|付近.*M2.*8.*TNT|分.*ころ.*熊本県.*阿蘇|0.*熊本県.*熊本.*地方|01.*45.*55.*発
生|01.*59.*発生.*M|04.*16.*07.*09|04.*16.*16.*02|04.*16.*17.*40|1.*度.*さ.*10|10.*km.*エムスリー株式会
社.*6|130.*7.*度.*震度
3|16.*04.*51.*24|16.*07.*09.*51|16.*08.*08.*51|16.*21.*44.*39|2.*TNT.*951.*5|2016.*04.*16.*3|5.*3.*最大.*震度
|5.*9.*さ.*10|5.*強.*推定.*詳細|59.*発生.*M.*5|7.*6.*kg.*さ|7.*地震.*発生.*震源|7.*度.*震度 3.*熊本県
|9.*E.*130.*9|AQUA.*HYPO.*G.*Y|EARTH.*Quake.*eqjp.*地震情報|G.*Y.*緊急地震速報.*最終|G.*気象庁.*地震情報.*4
月 16 日|jishin.*八.*代.*地震|JST 株式会社.*日本.*熊本県.*大津町|km.*エムスリー株式会社.*5.*最大|km.*エムスリー株式会
社.*8.*最大|M1.*8.*TNT.*7|M4.*5.*地震.*発生|TNT.*7.*6.*kg|Y.*緊急地震速報.*最終.*報|Yatsushiro.*jishin.*八.*代|さ.*
地震.*規模.*マグニチュード|確定.*情報.*16 日.*4|強.*推定.*詳細.*2016|熊本県.*熊本.*地方.*M|熊本県.*微小.*地震速報.*熊
本県|頃.*八.*代.*揺れ|最大.*予測.*震度 3.*1|時.*23.*分.*頃|時.*26.*分.*頃|弱.*37.*00.*2016|情報.*16 日.*4.*時|震源.*さ.*
地震.*規模|震源.*破獄.*地震.*規模|震度.*5.*強.*推定|震度.*情報.*16 日.*03|震度.*分離.*ため.*情報|震度 3.*エムスリー株式
会社.*3.*地震|代.*揺れ.*計測.*震度|地震.*震源.*破獄.*地震|地震情報.*4 月 16 日.*04.*時|地方.*震源.*さ.*地震|地方.*地震.*
震源.*破獄|八.*代.*揺れ.*計測|付近.*M2.*1.*TNT|分.*頃.*八.*代|分離.*ため.*情報.*震度|予測.*震度 3.*1.*報|%.*12.*18.*時
|02.*JST 株式会社.*日本.*熊本県|04.*16.*03.*34|04.*16.*04.*24|10.*7.*kg.*さ|12.*時.*%.*12|131.*1.*E.*0|15.*1.*kg.*さ
|16.*10.*38.*53|16.*18.*25.*10|2.*さ.*10.*km|2.0.*TNT.*15.*1|28.*JST 株式会社.*日本.*熊本県
|3.*TNT.*1.*3|32.*7.*130.*8|476.*9.*kg.*さ|5.*3.*熊本県.*熊本|6.*12.*時.*%|6.*報.*震源.*熊本県|7.*130.*8.*熊本県
|7.*N.*130.*8|8.*N.*130.*7|9.*km.*2016.*04|9.*度.*東経.*130|G.*Y.*2016.*04|GT.*北緯.*33.*度|JST 株式会社.*日本.*熊
本県.*産山村|km.*エムスリー株式会社.*7.*最大|M.*2.0.*TNT.*15|M1.*9.*TNT.*10|M4.*1.*地震.*発生
|TNT.*10.*7.*kg|TNT.*15.*1.*kg|TNT.*476.*9.*kg|Y.*2016.*04.*16|エムスリー株式会社.*0.*TNT.*476|エムスリー株式
会社.*4.*地震.*発生|さ.*9..*9.*km|ため.*情報.*震度.*震度|マグニチュード.*3.*3.*推定|阿蘇.*地方.*10.*3|宇城市.*震度.*情
報.*16 日|気象庁.*発表.*16 日.*03|規模.*マグニチュード.*3.*3|強震モニタ.*2016.*04.*16|熊本県.*阿蘇.*震度 2.*熊本県|熊
本県.*熊本.*地方.*震度 4|時.*%.*12.*18|時.*03.*分.*頃|時.*18.*分.*頃|情報.*震度.*震度.*情報|震源.*GT.*北緯.*33|震
度.*3.*1.*2|震度.*震度.*情報.*16 日|震度 3.*熊本県.*熊本.*震度 1|推定.*最大.*震度.*4|地方.*10.*5.*3|発表.*16 日.*03.*時|
報.*32.*8.*N|報.*M.*5.*3|0.*0.*緊急地震速報.*1|0.*3.*0.*0|0.*緊急地震速報.*1.*報|0.*微小.*地震速報.*熊本県
|04.*16.*11.*38|1.*3.*トン.*さ|1.*熊本県.*熊本.*地方|10.*4.*0.*3|10.*km.*エムスリー株式会社.*3|10.*km.*緊急地震速
報.*緊急地震速報|130.*7.*最大.*震度 3|16.*03.*55.*53|20160416082045.*2016.*04.*16|32.*8.*130.*9|36.*JST 株式会社.*
日本.*熊本県|4.*0.*3.*0|44.*分.*頃.*熊本県|5.*10.*km.*震度|5.*最大.*震度.*3|5.*微小.*地震速報.*熊本県|673.*6.*kg.*さ
|8.*130.*9.*熊本県|8.*推定.*地震.*津波|8K.*g.*さ.*9.|9.*地震.*発生.*震源|EARTH.*Quake.*地震情報.*4 月 16 日
|jishin.*EARTH.*Quake.*16 日|jishin.*EARTH.*Quake.*2016|jishin.*EARTH.*Quake.*熊本県|JST 株式会社.*日本.*熊本
県.*八代市|km.*推定.*規模.*M4|M.*5.*9.*さ|M2.*5.*微小.*地震速報|M4.*1.*AQUA.*REAL|M4.*3.*地震.*発生
|ND.*20160416082045.*2016.*04|TNT.*1.*3.*トン|TNT.*673.*6.*kg|エムスリー株式会社.*1.*TNT.*673|エムスリー株式
会社.*3.*TNT.*1|エムスリー株式会社.*5.*最大.*震度|エムスリー株式会社.*6.*さ.*10|マグニチュード.*3.*4.*推定|マグニ
チュード.*3.*8.*推定|規模.*マグニチュード.*3.*4|規模.*マグニチュード.*3.*8|緊急地震速報.*6.*報.*熊本県|緊急地震速
報.*8.*報.*熊本県|緊急地震速報.*予報.*ID.*20160416082045|熊本.*震度 2.*熊本県.*阿蘇|熊本.*地方.*エムスリー株式会
社.*5|熊本.*地方.*エムスリー株式会社.*9|熊本県.*熊本県熊本市中央区.*区.*熊本県熊本市東区|熊本県熊本市南区.*区.*熊本県
熊本市北区.*区|最大.*震度.*6.*弱|最大.*予測.*震度 4.*最終|時.*25.*分.*頃|時.*46.*分.*頃|震度 3.*1.*報.*32|震度 3.*エムス
リー株式会社.*7.*地震|震度 3.*熊本県.*阿蘇.*震度 2|推定.*最大.*震度.*3|推定.*震度 3.*熊本県.*阿蘇|地方.*10.*4.*0|地
方.*32.*9.*N|地方.*北緯.*32.*9|発生.*G.*気象庁.*16 日|付近.*M.*2.0.*TNT|付近.*M1.*9.*TNT|付近.*エムスリー株式会
社.*0.*TNT|付近.*エムスリー株式会社.*3.*TNT|報.*32.*6.*N|報.*32.*7.*N|予測.*震度 4.*最終.*報|℃.*注意報.*乾燥.*熊本
県熊本市|0.*0.*緊急地震速報.*3|0.*km.*エムスリー株式会社.*8|0.*緊急地震速報.*3.*報|0.*熊本県.*阿蘇.*地方|01.*25.*05.*
熊本県|01.*59.*32.*8|03.*03.*10.*発生|04.*16.*03.*26|04.*16.*04.*18|07.*JST 株式会社.*日本.*熊本県
|10.*3.*7.*3|10.*3.*8.*3|10.*km.*M4.*0|10.*km.*マグニチュード.*4|10.*km.*規模.*M|10.*km.*規模.*エムスリー株式会
社|12.*時.*48.*分|131.*1.*最大.*震度 3|15.*分.*頃.*熊本県
|16.*01.*44.*09|16.*01.*59.*32|16.*04.*24.*26|16.*08.*02.*47|16.*17.*40.*20|2.*7.*kg.*さ
|2.*9.*E.*131|20160416014412.*2016.*04.*16|20160416160204.*2016.*04.*16|2016 年.*4 月 16 日.*4.*時|20K.*m.*地震.*
規模|22.*JST 株式会社.*日本.*熊本県|3.*3.*0.*0|3.*7.*3.*0|3.*8.*3.*0|3.*最大.*震度.*3|32.*6.*度.*東経
|32.*9.*130.*9|4.*最大.*震度.*5|43.*JST 株式会社.*日本.*熊本県|5.*4.*最大.*震度|5.*4K.*g.*さ|5.*弱.*37.*00|5.*推定.*地
震.*津波|6.*度.*東経.*130|7.*3.*0.*0|7.*TNT.*5.*4K|7.*最大.*震度.*3|8.*3.*0.*0|8.*最大.*震度.*3|8.*推定.*地震.*発生
|8.*微小.*地震速報.*熊本県|9.*10.*km.*M|9.*130.*9.*熊本県|9.*N.*130.*9|EARTH.*Quake.*eqjp.*気象庁|G.*Y.*熊本県.*
北西|JST 株式会社.*日本.*熊本県.*甲佐町|km.*M.*5.*5|km.*エムスリー株式会社.*6.*最大|M.*5.*4.*熊本県|M.*5.*4.*最大
|M1.*5.*TNT.*2|M1.*7.*TNT.*5|M2.*8.*微小.*地震速報|M4.*0.*AQUA.*REAL|M4.*2.*
さ.*10|ND.*20160416014412.*2016.*04|ND.*20160416160204.*2016.*04|NE.*KUMAMOTO.*PREF.*32|TNT.*2.*7.*kg|
TNT.*5.*4K.*g|Y.*熊本県.*北西.*部|エムスリー株式会社.*0.*微小.*地震速報|エムスリー株式会社.*7.*最大.*震度|エムス
リー株式会社.*8.*最大.*震度|エムスリー株式会社.*9.*さ.*10|さ.*10.*9.*km|さ.*8.*6.*km|さ.*推定.*規模.*エムスリー株式会
社|マグニチュード.*3.*6.*推定|もの.*予想.*強震モニタ.*地震|阿蘇.*地方.*10.*5|影響.*もの.*予想.*強震モニタ|益城町.*惣
領.*付近.*震度|確定.*情報.*16 日.*3|乾燥.*熊本県熊本市.*kumamoto.*熊本|気象庁.*震源.*情報.*04|規模.*マグニチュー
ド.*3.*6|緊急地震速報.*1.*報.*M4|緊急地震速報.*3.*報.*M4|緊急地震速報.*3.*報.*エムスリー株式会社|熊本.*地方.*エムス
リー株式会社.*4|熊本.*地方.*エムスリー株式会社.*8|熊本県.*阿蘇.*地方.*推定|熊本県.*益城町.*惣領.*付近|熊本県.*熊本市.*
城南町藤山.*付近|熊本県.*産山村.*山鹿.*付近|熊本県.*産山村.*田尻.*付近|最終.*報.*熊本県.*阿蘇|時.*02.*分.*頃|時.*05.*
分.*頃|時.*10.*分.*頃|情報.*16 日.*14.*時|情報.*16 日.*3.*時|心配.*地震.*緊急地震速報.*発表|震源地.*日本.*熊本県.*益城町
|震度.*5.*弱.*37|震度.*情報.*16 日.*21|震度 3.*エムスリー株式会社.*6.*地震|震度 3.*熊本県.*熊本.*震度 2|震度 4.*最終.*

報.*32|推定.*震度 4.*熊本県.*熊本|地震情報.*2016 年.*4 月 16 日.*4|地震情報.*4 月 16 日.*07.*時|地方.*10.*3.*8|地方.*10.*5.*4|注意報.*乾燥.*熊本県熊本市.*kumamoto|津波.*心配.*地震.*緊急地震速報|津波.*心配.*地震情報.*4 月 16 日|日本.*熊本県.*益城町.*惣領|日本.*熊本県.*熊本市.*城南町藤山|日本.*熊本県.*産山村.*山鹿|日本.*熊本県.*産山村.*田尻|発生.*M4.*0.*さ|発生.*エムスリー株式会社.*7.*さ|発生.*エムスリー株式会社.*8.*さ|付近.*M1.*8.*TNT|付近.*エムスリー株式会社.*1.*TNT|報.*M.*5.*4|北緯.*32.*6.*度|予想.*強震モニタ.*地震.*発生|0.*0.*緊急地震速報.*5|0.*10.*km.*M4|0.*kg.*さ.*10|0.*km.*エムスリー株式会社.*9|0.*緊急地震速報.*5.*報|0.*最大.*震度.*3|03.*03.*10.*熊本県|03.*時.*03.*分|04.*16.*04.*00|04.*16.*07.*57|04.*16.*09.*16|04.*16.*11.*03|04.*16.*15.*48|04.*月.*16 日.*02|04.*月.*16 日.*09|05.*熊本県.*北西.*部|07.*09.*51.*発生|07.*発生.*M.*5|08.*20.*41.*発生|09.*JST 株式会社.*日本.*熊本県|09.*時.*16.*分|1.*9.*kg.*さ|1.*E.*0.*1|1.*kg.*さ.*10|1.*km.*M.*5|1.*さ.*10.*km|1.*地震.*発生.*震源|1.*報.*32.*6|10.*5.*3.*5|10.*熊本県.*北西.*部|10.*時.*26.*分|11.*02.*51.*発生|130.*7.*推定.*エムスリー株式会社|130.*8.*最大.*震度 3|131.*0.*熊本県.*阿蘇|131.*1.*E.*5|131.*2.*E.*10|131.*2.*度.*さ|14.*27.*01.*発生|16.*09.*16.*28|16.*14.*03.*57|16.*分.*頃.*熊本県|17.*時.*40.*分|18.*分.*頃.*熊本県|20.*時.*44.*分|2016.*04.*16.*22|2016.*04.*16.*23|2016.*4.*16.*4|23.*分.*頃.*熊本県|25.*05.*熊本県.*北西|3.*3.*推定.*地震|3.*5.*弱.*0|3.*9.*3.*0|3.*最大.*震度.*5|3.*地震.*津波.*心配|32.*6.*130.*8|32.*9.*N.*131|32.*発生.*最大.*予測|4.*地震.*発生.*予想|44.*07.*発生.*M|48.*32.*発生.*最大|48.*分.*頃.*熊本県|5.*0.*km.*M4|5.*3.*5.*弱|5.*4.*熊本県.*熊本|5.*強.*0.*1|5.*地震.*津波.*心配|55.*JST 株式会社.*日本.*熊本県|59.*32.*8.*130|6.*130.*8.*熊本県|6.*E.*130.*8|6.*報.*熊本県.*熊本|7.*最大.*震度 3.*エムスリー株式会社|8.*6.*km.*MAP|8.*報.*熊本県.*熊本|9.*3.*0.*0|9.*最大.*震度.*3|E.*130.*8.*推定|EARTH.*Quake.*熊本県.*北西|EARTH.*Quake.*微小.*地震速報|eqjp.*気象庁.*情報.*16 日|K.*NET.*強震モニタ.*速報|km.*M4.*0.*最大|km.*M4.*1.*AQUA|km.*MAP.*緊急地震速報.*予報|km.*エムスリー株式会社.*8.*AQUA|km.*エムスリー株式会社.*9.*AQUA|M.*5.*5.*さ|M.*5.*8.*地震|m.*地震.*規模.*マグニチュード|M1.*4.*TNT.*1|M4.*0.*最大.*震度|MAP.*緊急地震速報.*予報.*ID|NET.*強震モニタ.*速報.*LV|Quake.*eqjp.*気象庁.*情報|Quake.*熊本県.*北西.*部|Quake.*微小.*地震速報.*熊本県|TNT.*1.*9.*kg|エムスリー株式会社.*8.*推定.*地震|エムスリー株式会社.*9.*最大.*震度|さ.*9..*1.*km|マグニチュード.*3.*9.*推定|マグニチュード.*4.*3.*推定|阿蘇.*地方.*震度 3.*エムスリー株式会社|芦北.*震度 1.*福岡県.*筑後|気象庁.*情報.*16 日.*04|気象庁.*発表.*16 日.*01|規模.*マグニチュード.*3.*9|規模.*マグニチュード.*4.*3|緊急地震速報.*警報.*ID.*20160416014412|熊本.*地方.*M.*5|熊本.*地方.*M4.*3|熊本.*地方.*エムスリー株式会社.*7|熊本.*地方.*震度 4.*M4|熊本県.*阿蘇.*震度 1.*熊本県|熊本県.*阿蘇市.*山田.*付近|熊本県.*熊本.*震度 3.*熊本県|月.*16 日.*02.*時|月.*16 日.*09.*時|最高.*21.*℃.*最低|最大.*震度.*震度 4.*jishin|産山村.*山鹿.*付近.*震度|時.*32.*分.*頃|時.*55.*分.*頃|情報.*16 日.*04.*時|震源地.*日本.*熊本県.*熊本市|震源地.*日本.*熊本県.*産山村|震度.*5.*弱.*緊急地震速報|震度.*震度 4.*jishin.*EARTH|震度.*分離.*ため.*情|震度 3.*G.*Y.*緊急地震速報|震度 3.*熊本県.*阿蘇.*熊本県|震度 4.*jishin.*EARTH.*Quake|震度 4.*M4.*5.*地震|推定.*エムスリー株式会社.*8.*推定|速報.*LV.*4.*16 日|地震.*jishin.*災害.*saiga|地震.*津波.*心配.*気象庁|地震.*微小.*地震速報.*熊本県|天草.*芦北.*震度 1.*熊本県|天草.*芦北.*震度 1.*福岡県|度.*震度 3.*熊本県.*阿蘇|曇.*一.*時.*雨|日本.*熊本県.*阿蘇市.*山田|日本.*熊本県.*益城町.*付近|発生.*エムスリー株式会社.*9.*さ|発生.*最大.*震度 3.*エムスリー株式会社|発生.*震源.*さ.*20K|発表.*16 日.*01.*時|予想.*最大.*震度.*震度 4|℃.*降水確率.*40.*%|0.*0.*地震速報.*2016|0.*1.*km.*M4|0.*6.*時.*100|0.*地震速報.*2016.*04|01.*59.*発生.*最大|04.*16.*01.*29|04.*16.*05.*33|04.*月.*16 日.*04|07.*時.*01.*分|08.*20.*41.*32|09.*48.*32.*32|1.*10.*km.*M4|10.*3.*6.*3|10.*3.*9.*3|10.*5.*4.*5|10.*JST 株式会社.*日本.*熊本県|10.*km.*M4.*1|10.*発生.*M.*5|100.*6.*12.*時|13.*時.*15.*分|130.*7.*E.*5|130.*8.*E.*9.|130.*8.*度.*震度 3|14.*時.*03.*分|14.*時.*27.*分|15.*38.*38.*発生|15.*時.*48.*分|16.*01.*29.*10|16.*02.*04.*10|16.*04.*18.*14|16.*07.*57.*21|16.*11.*38.*24|16.*14.*03.*56|16 日.*03.*時.*03|17 日.*日.*雨.*のち|2.*6.*E.*131|2.*地震.*発生.*震源|2.*報.*2016.*4|20.*41.*32.*7|20.*41.*発生.*M4|2016.*04.*16.*4|20160416094838.*2016.*04.*16|20 日.*水.*曇.*時|21 日.*木.*曇.*一|23.*JST 株式会社.*日本.*熊本県|24 時.*%.*天気.*c|26.*分.*頃.*熊本県|3.*6.*3.*0|30.*JST 株式会社.*日本.*熊本県|32.*32.*8.*130|34.*分.*頃.*熊本県|35.*JST 株式会社.*日本.*熊本県|4.*3.*3.*0|4.*5.*弱.*0|4.*報.*2016.*4|40.*JST 株式会社.*日本.*熊本県|41.*32.*7.*130|46.*JST 株式会社.*日本.*熊本県|476.*9.*g.*さ|48.*32.*32.*8|5.*4.*5.*弱|5.*5.*さ.*10|51.*JST 株式会社.*震源地.*日本|51.*発生.*最大.*予測|52.*JST 株式会社.*日本.*熊本県|53.*JST 株式会社.*日本.*熊本県|58.*JST 株式会社.*日本.*熊本県|59.*発生.*最大.*予測|6.*3.*0.*0|6.*M2.*6.*熊本県|6.*N.*130.*7|6.*熊本県.*北西部.*さ|6.*最大.*震度.*3|6.*時.*100.*6|6.*推定.*地震.*津波|6.*地震.*津波.*心配|7.*km.*M2.*5|7.*M2.*7.*熊本県|7.*熊本県.*北西部.*さ|8.*N.*130.*9|8.*最大.*震度 3.*エムスリー株式会社|8.*地震.*津波.*心配|8.*地震.*発生.*最大|8.*度.*震度 3.*熊本県|8.*報.*震源.*熊本県|E.*131.*2.*最大|E.*131.*2.*推定|E.*2016.*04.*16|E.*6.*0.*km|E.*7.*0.*km|EARTH.*Quake.*1.*報|G.*Y.*緊急地震速報.*3|G.*Y.*緊急地震速報.*警報|G.*Y.*緊急地震速報.*予報|G.*Y.*速報.*LV|GT.*北緯.*32.*8|jishin.*EARTH.*Quake.*微小|JST 株式会社.*日本.*熊本県.*西原村|km.*4 月 16 日.*1.*時|km.*4 月 16 日.*4.*時|km.*M4.*0.*AQUA|km.*エムスリー株式会社.*4.*最大|M1.*0.*TNT.*476|M2.*6.*M2.*6|M2.*7.*M2.*7|M4.*3.*AQUA.*REAL|ND.*20160416094838.*2016.*04|Quake.*1.*報.*2016|TNT.*476.*9.*g|Y.*M4.*2.*さ|Y.*緊急地震速報.*3.*報|Y.*緊急地震速報.*予報.*ID|エムスリー株式会社.*5.*地震.*発生|エムスリー株式会社.*6.*最大.*震度|さ.*10.*6.*km|さ.*3.*0.*km|さ.*9..*3K.*m|阿蘇.*震度 2.*熊本県.*熊本|阿蘇.*地方.*M.*5|阿蘇.*地方.*エムスリー株式会社.*6|阿蘇.*地方.*さ.*20K|阿蘇.*地方.*地震.*発生|雨.*のち.*晴.*最高|気象庁.*震源.*情報.*03|規模.*エムスリー株式会社.*9.*推定|宮崎県.*北部.*山.*沿い|強震モニタ.*緊急地震速報.*予報.*ID|緊急地震速報.*警報.*ID.*20160416094838|緊急地震速報.*警報.*ID.*20160416160204|熊本.*震度 1.*熊本県.*天草|熊本.*地方.*32.*9|熊本県.*阿蘇.*熊本県.*熊本|熊本県.*阿蘇.*地方.*M|熊本県.*宇城市.*震度.*情報|熊本県.*益城町.*小池.*付近|熊本県.*熊本市.*富合町木原.*付近|熊本県.*天草.*芦北.*震度 2|熊本県.*天草.*芦北.*震度 3|熊本県.*天草.*芦北.*地方|月.*16 日.*04.*時|降水確率.*40.*%.*天気|最終.*震源.*熊本県.*阿蘇|最終.*報.*32.*7|最大.*速度.*0.*4|最大.*予測.*震度 3.*最終|最大.*予測.*震度 4.*1|最大.*予測.*震度 5 強.*G|時.*100.*6.*12|時.*24.*分.*頃|時.*27.*分.*頃|時.*45.*分.*頃|時.*47.*分.*頃|時.*49.*分.*頃|情報.*16 日.*22.*時|心配.*jishin.*地震情報.*4 月 16 日|心配.*気象庁.*情報.*16 日|震度.*3.*2.*2|震度.*情報.*16 日.*05|震度.*情報.*16 日.*07|震度 2.*熊本県.*阿蘇.*震度 1|震度 2.*熊本県.*熊本県熊本市中央区.*区|震度 3.*熊本県.*熊本.*地震情報|震度 4.*M4.*3.*地震|水.*曇.*時.*晴|地震.*震源地.*熊本県.*阿蘇|地震.*震度 2.*熊本県.*阿蘇市|地震.*震度 2.*熊本県.*熊本県熊本市中央区|地震.*震度 3.*熊本県.*熊本県熊本市西区|地方.*10.*3.*6|地方.*10.*3.*9|地方.*推定.*震度 4.*M4|津波.*心配.*気象庁.*情報|津波注意報.*発表.*中.*地震|店.*in.*熊本市.*熊本県|度.*東経.*130.*9|東経.*130.*9.*度|日.*雨.*のち.*晴|日本.*熊本県.*益城町.*小池|日本.*熊本県.*熊本市.*富合町木原|発生.*M.*5.*5|発生.*M.*5.*9|発生.*エムスリー株式会社.*5.*さ|発生.*エムスリー株式会社.*6.*さ|発生.*最大.*震度 4.*M4|発生.*最大.*予測.*震度 5 強|付近.*M1.*0.*TNT|付

近.*M1.*5.*TNT|付近.*M1.*7.*TNT|報.*33.*N.*131|報.*M4.*3.*熊本県|木.*曇.*一.*時|予測.*震度 3.*最終.*報|予測.*震度 4.*1.*報|予測.*震度 5 強.*G.*Y|0.*km.*M4.*2|0.*km.*MAP.*1200|0.*度.*東経.*131|01.*25.*05.*発生|01.*29.*10.*熊本県|01.*44.*07.*32|01.*44.*09.*発生|01.*時.*46.*分|02.*51.*32.*8|02.*51.*発生.*M4|02.*時.*57.*分|03.*10.*熊本県.*北東部|04.*09.*26.*発生|04.*16.*03.*16|04.*月.*16 日.*07|04.*時.*18.*分|05.*52.*51.*発生|06.*JST 株式会社.*日本.*熊本県|07.*23.*54.*発生|07.*32.*8.*130|07.*時.*23.*分|07.*発生.*最大.*予測|08.*08.*51.*発生|1.*10.*km.*震度|1.*9.*トン.*さ|1.*E.*5.*0|1.*最大.*震度 3.*エムスリー株式会社|1.*度.*震度 3.*熊本県
|10.*0.*km.*MAP|10.*2.*km.*MAP|10.*3.*5.*3|10.*3K.*m.*MAP|10.*4.*3.*3|10.*6.*km.*MAP|10.*km.*M4.*2|10.*km.*M4.*7|10.*km.*M4.*8|10.*km.*MAP.*2016|10.*km.*MAP.*37|10.*km.*緊急地震速報.*警報|10.*熊本県.*北東部.*NE|10.*分.*頃.*熊本県|1015.*Y.*M4.*2|11.*02.*51.*32|11.*時.*49.*分|12.*JST 株式会社.*日本.*熊本県|13.*15.*28.*発生|130.*8.*E.*緊急地震速報|131.*1.*度.*震度 3|16.*03.*16.*39|16.*05.*33.*46|16.*08.*02.*48|16.*15.*48.*12|16 日.*01.*時.*46|18.*JST 株式会社.*日本.*熊本県|2.*10.*km.*エムスリー株式会社|2.*19 日.*火.*晴
|2.*E.*10.*km|2.*km.*2016.*04|20.*熊本県.*北西.*部
|2016.*04.*16.*19|20160416045129.*2016.*04.*16|20160416080251.*2016.*04.*16|20160416110256.*2016.*04.*16|20160416142706.*2016.*04.*16|22.*時.*53.*分|29.*10.*熊本県.*北西|3.*5.*3.*0|3.*緊急地震速報.*4.*報|3.*熊本県.*阿蘇.*地方|3.*地震.*発生.*最大|3.*報.*熊本県.*阿蘇|3.*予想.*発生.*時刻|31.*JST 株式会社.*日本.*熊本県|32.*6.*130.*7|32.*JST 株式会社.*日本.*熊本県|33.*0.*度.*東経|38.*JST 株式会社.*日本.*熊本県|39.*JST 株式会社.*日本.*熊本県|3 分.*頃.*熊本県.*阿蘇|4.*0.*1.*緊急地震速報|4.*3.*さ.*10|4.*地震.*発生.*最大|42.*JST 株式会社.*日本.*熊本県|44.*07.*32.*8|44.*07.*発生.*最大|44.*JST 株式会社.*日本.*熊本県|45.*55.*発生.*M6|46.*分.*頃.*熊本県|48.*JST 株式会社.*日本.*熊本県
|5.*0.*km.*MAP|5.*0.*km.*エムスリー株式会社|5.*0.*さ.*10|5.*3.*0.*0|5.*4.*トン.*さ|5.*8.*AQUA.*REAL|5.*推定.*地震.*発生|5.*分.*頃.*熊本県|5.*報.*2016.*4|51.*32.*8.*130|51.*JST 株式会社.*日本.*熊本県|56.*JST 株式会社.*日本.*熊本県|57.*JST 株式会社.*日本.*熊本県|6.*130.*7.*熊本県|6.*N.*130.*8|6.*熊本県.*熊本.*地方|6.*弱.*推定.*詳細|6.*地震.*発生.*最大|7.*6.*トン.*さ|7.*TNT.*5.*4|7.*最大.*震度.*4|7.*地震.*津波.*心配|7.*地震.*発生.*最大|7.*度.*さ.*推定|7.*報.*震源.*熊本県|8.*10.*km.*震度|8K.*g.*さ.*11|9.*N.*131.*1|9.*推定.*地震.*津波|9.*推定.*津波.*心配|9.*地震.*津波.*心配
|9..*1.*km.*MAP|9..*9.*km.*MAP|951.*5.*g.*さ|E.*131.*1.*推定
|E.*8.*0.*km|E.*8.*4K.*m|E.*9..*6.*km|EARTH.*Quake.*強震モニタ.*地震|G.*Y.*1.*報|G.*Y.*アニメ.*熊本県|G.*Y.*地震速報.*2016|JST 株式会社.*日本.*熊本県.*南
|km.*EARTH.*Quake.*2016|km.*M.*5.*2|km.*M.*5.*8|km.*M2.*6.*M2|km.*M2.*7.*M2|km.*M4.*2.*最大
|km.*M4.*3.*AQUA|km.*M4.*7.*最大|km.*MAP.*2016.*04|km.*MAP.*37.*00|km.*エムスリー株式会社.*0.*微小|km.*エムスリー株式会社.*4.*16 日|km.*規模.*M.*5|km.*規模.*M4.*3|km.*震度.*4.*1|M.*5.*8.*AQUA|m.*M2.*5.*微小
|M.*W3.*5.*さ|M1.*2.*TNT.*951|M4.*3.*10.*km|M4.*4.*熊本県.*熊本|M4.*7.*さ.*10|M4.*7.*最大.*震度
|MAP.*2016.*04.*16|MAP.*37.*00.*2016|Mw.*4.*3.*さ
|ND.*20160416045129.*2016.*04|ND.*20160416080251.*2016.*04|ND.*20160416110256.*2016.*04|ND.*20160416142706.*2016.*04|Quake.*強震モニタ.*地震.*発生|Quake.*緊急地震速報.*警報.*ID|TNT.*1.*9.*トン|TNT.*5.*4.*トン
|TNT.*7.*6.*トン|TNT.*951.*5.*g|W3.*5.*さ.*10|Y.*1.*報.*2016|Y.*地震速報.*2016.*04|エムスリー株式会社.*4.*TNT.*1|エムスリー株式会社.*6.*地震.*津波|エムスリー株式会社.*7.*TNT.*5|エムスリー株式会社.*8.*TNT.*7|さ.*10.*0.*km|さ.*10.*2.*km|さ.*10.*3K.*m|さ.*10.*7.*km|さ.*11.*1.*km|さ.*20K.*m.*地震|さ.*8.*4K.*m|さ.*8.*7.*km|さ.*9..*7.*km|マグニチュード.*3.*5.*推定|マグニチュード.*4.*5.*推定|阿蘇.*熊本県.*天草.*芦北|阿蘇.*地方.*M4.*3|阿蘇.*地方.*M4.*5|芦北.*震度 1.*熊本県.*球磨|確定.*情報.*16 日.*5|気象庁.*情報.*16 日.*03|気象庁.*発表.*16 日.*05|気象庁.*発表.*16 日.*07|規模.*M4.*1.*推定|規模.*M4.*3.*推定|規模.*M4.*5.*推定|規模.*エムスリー株式会社.*4.*推定|規模.*エムスリー株式会社.*6.*推定|規模.*マグニチュード.*3.*5|規模.*マグニチュード.*4.*5|球磨.*熊本県.*天草.*芦北|強.*0.*1.*緊急地震速報|緊急地震速報.*予報.*ID.*20160416045129|緊急地震速報.*予報.*ID.*20160416080251|緊急地震速報.*予報.*ID.*20160416110256|熊本.*震度 1.*熊本県.*阿蘇|熊本.*震度 3.*熊本県.*阿蘇|熊本.*地方.*32.*6|熊本.*地方.*エムスリー株式会社.*3|熊本.*地方.*エムスリー株式会社.*6|熊本.*地方.*さ.*20K|熊本.*地方.*さ.*M4|熊本県.*阿蘇.*熊本県.*天草|熊本県.*阿蘇.*地方.*震度 4|熊本県.*宇城市.*松橋町古保山.*付近|熊本県.*菊陽町.*辛川.*付近|熊本県.*球磨.*熊本県.*天草|熊本県.*北西部.*さ.*9.|熊本県熊本市中央区.*区.*熊本県熊本市東区.*熊本県熊本市西区|熊本県熊本市東区.*熊本県熊本市西区.*熊本県熊本市南区.*区|月.*16 日.*07.*時|最高.*25.*℃.*最低|最大.*震度 3.*エムスリー株式会社.*3|時.*06.*分.*頃|時.*20.*分.*頃|時.*30.*分.*頃|時.*3 分.*頃.*熊本県|時.*40.*分.*頃|時.*5.*分.*頃|時.*53.*分.*頃|時.*54.*分.*頃|詳細.*気象庁.*情報.*地震情報|情報.*16 日.*03.*時|情報.*16 日.*21.*時|情報.*16 日.*5.*時|心配.*地震情報.*2016 年.*4 月 16 日|震度.*3.*緊急地震速報.*4|震度.*4.*1.*3|震度.*6.*弱.*推定|震度.*M.*W3.*5|震度.*Mw.*4.*3|震度.*情報.*16 日.*01|震度 2.*熊本県.*天草.*芦北|震度 3.*エムスリー株式会社.*8.*地震|震度 4.*エムスリー株式会社.*9.*地震|震度 4.*熊本県.*熊本.*震度 3|代.*地震.*04.*月|地震.*最大.*震度.*6|地震情報.*4 月 16 日.*03.*時|地方.*10.*3.*5|地方.*10.*3.*7|地方.*32.*6.*N|地方.*M4.*3.*10|地方.*震度 3.*エムスリー株式会社.*3|地方.*北緯.*32.*6|地方.*北緯.*33.*0|津波.*心配.*地震情報.*2016 年|度.*さ.*20K.*m|曇.*時.*晴.*最高|南阿蘇村.*震度.*情報.*16 日|日本.*熊本県.*阿蘇市.*付近|日本.*熊本県.*宇城市.*松橋町古保山|日本.*熊本県.*菊陽町.*辛川|日本.*熊本県.*南.*区役所|破獄.*地震.*規模.*エムスリー株式会社|八.*代.*地震.*04|発表.*16 日.*05.*時|発表.*16 日.*07.*時|発表.*震度.*情報.*16 日|付近.*M1.*4.*TNT|付近.*エムスリー株式会社.*4.*TNT|付近.*エムスリー株式会社.*7.*TNT|付近.*エムスリー株式会社.*8.*TNT|報.*M4.*4.*熊本県|北緯.*33.*0.*度|揺れ.*計測.*震度.*4|℃.*最低.*12.*℃|0.*1.*km.*M|0.*10.*km.*M|0.*5.*TNT.*84|0.*8.*TNT.*239|0.*km.*M.*5|0.*km.*エムスリー株式会社.*6|0.*km.*エムスリー株式会社.*7|01.*45.*55.*32|01.*JST 株式会社.*日本.*熊本県|03.*10.*発生.*M|03.*時.*55.*分|04.*05.*49.*発生|04.*16.*01.*33|04.*16.*03.*32|04.*16.*05.*53|04.*16.*16.*54|04.*23.*18.*発生|04.*JST 株式会社.*日本.*熊本県|04.*月.*16 日.*05|04.*時.*05.*分|04.*時.*51.*分|07.*23.*54.*32|07.*時.*42.*分|08.*51.*32.*9|08.*時.*20.*分|1.*℃.*注意報.*乾燥|1.*10.*km.*M|1.*16 日.*03.*時|1.*16 日.*04.*時|1.*km.*2016.*04|1.*Kumamoto.*jishin.*熊本|1.*地震.*発生.*最大|1.*報.*32.*8|10.*38.*53.*発生|10.*4.*7.*4|10.*9.*km.*MAP|10.*時.*55.*分|10.*発生.*最大.*予測|11.*JST 株式会社.*日本.*熊本県|11.*時.*29.*分|11.*時.*38.*分|13.*0.*km.*MAP|13.*1.*℃.*注意報
|130.*6.*10.*km|130.*7.*E.*3|130.*8.*最大.*震度 4|131.*1.*熊本県.*熊本|131.*2.*最大.*震度 3|14.*JST 株式会社.*日本.*熊本県|14.*時.*46.*分|14.*時.*58.*分|15.*時.*15.*分
|16.*01.*30.*50|16.*01.*33.*20|16.*03.*26.*55|16.*04.*15.*01|16.*08.*20.*42|16.*14.*27.*03|16.*時.*57.*分|16 日.*04.*時.*05|16 日.*08.*時.*20|16 日.*12.*時.*48|16 日.*14.*時.*03|2.*10.*km.*M4|2.*km.*M2.*6|2.*熊本県.*北西部.*さ|2.*最大.*速度.*0|2.*報.*熊本県.*阿蘇
|2016.*4.*16.*2|2016.*4.*16.*3|20160416055256.*2016.*04.*16|20160416131533.*2016.*04.*16|20160416182513.*2016.

*04.*16|20160416210511.*2016.*04.*16|2016 年.*4 月 16 日.*3.*時|21.*05.*06.*発生|21.*44.*39.*発生|22.*時.*10.*分|22.*時.*25.*分|23.*54.*32.*8|239.*0.*g.*さ|24.*JST 株式会社.*日本.*熊本県|24.*分.*頃.*熊本県|25.*分.*頃.*熊本県|26.*33.*0.*131|26.*発生.*最大.*予測|27.*JST 株式会社.*日本.*熊本県|27.*分.*頃.*熊本県|29.*JST 株式会社.*日本.*熊本県|3.*0.*km.*4 月 16 日|3.*36.*00.*2016|3.*6.*推定.*地震|3.*8.*トン.*さ|3.*8.*推定.*地震|3.*9.*推定.*地震|3.*Kumamoto.*jishin.*熊本|3.*推定.*最大.*震度|32.*9.*131.*0|32.*分.*頃.*熊本県|33.*分.*頃.*熊本県|34.*JST 株式会社.*日本.*熊本県|3K.*m.*2016.*04|3K.*m.*M2.*6|4.*3.*推定.*地震|4.*7.*4.*0|4.*緊急地震速報.*最終.*報|4.*地震.*津波.*心配|4.*地震.*発生.*震源|4.*報.*熊本県.*阿蘇|45.*55.*32.*9|45.*分.*頃.*熊本県|49.*JST 株式会社.*日本.*熊本県|49.*分.*頃.*熊本県|4K.*m.*M2.*5|4K.*m.*M2.*7|4K.*m.*M2.*8|5.*0.*km.*4 月 16 日|5.*5.*AQUA.*REAL|5.*Oita.*jishin.*大分|5.*TNT.*84.*8|5.*地震.*発生.*予想|50.*JST 株式会社.*日本.*熊本県|51.*発生.*M4.*4|53.*発生.*最大.*予測|55.*発生.*最大.*予測|55.*分.*頃.*熊本県|57.*分.*震源.*熊本県|59.*JST 株式会社.*日本.*熊本県|6.*0.*km.*エムスリー株式会社|6.*E.*131.*0|6.*km.*2016.*04|6.*km.*M2.*8|6.*TNT.*3.*8|6.*推定.*津波.*心配|6.*分.*頃.*熊本県|7.*推定.*津波.*心配|7.*報.*M.*5|8.*4K.*m.*MAP|8.*7.*km.*MAP|84.*8.*g.*さ|9.*10.*km.*M4|9.*10.*km.*震度|9.*131.*0.*熊本県|9.*E.*131.*0|9.*km.*M2.*6|9.*地震.*発生.*最大|9..*3K.*m.*MAP|9..*7.*km.*MAP|9..*8K.*m.*MAP|951.*5.*kg.*さ|AQUA.*REAL.*M.*5|AQUA.*REAL.*M4.*2|E.*10.*4K.*m|E.*11.*0.*km|E.*13.*0.*km|E.*130.*6.*10|E.*9..*0.*km|E.*9..*9.*km|EARTH.*Quake.*確定.*情報|EARTH.*Quake.*地震情報.*2016 年|G.*Y.*熊本県.*熊本|G.*気象庁.*微小.*地震速報|GT.*北緯.*32.*9|jishin.*EARTH.*Quake.*確定|jishin.*微小.*地震速報.*熊本県|JST 株式会社.*日本.*熊本県.*宇土市|K.*NET.*強震モニタ.*2016|K.*NET.*強震モニタ.*緊急地震速報|km.*M.*5.*1|km.*M2.*8.*微小|km.*M4.*8.*最大|km.*エムスリー株式会社.*3.*最大|km.*緊急地震速報.*緊急地震速報.*警報|km.*震度.*3.*2|KUMAMOTO.*PREF.*32.*9|M.*0.*5.*TNT|M.*0.*8.*TNT|M.*5.*2.*さ|M.*5.*5.*AQUA|m.*M4.*2.*AQUA|m.*saigai.*jishin.*EARTH|M2.*6.*熊本県.*北西部|M4.*3.*熊本県.*阿蘇|M4.*5.*10.*km|M4.*8.*さ.*10|N.*130.*6.*E|N.*130.*739.*E|ND.*20160416055256.*2016.*04|ND.*20160416131533.*2016.*04|ND.*20160416182513.*2016.*04|ND.*20160416210511.*2016.*04|NET.*強震モニタ.*2016.*04|PREF.*32.*9.*N|Quake.*eqjp.*地震情報.*16 日|Quake.*eqjp.*地震情報.*4 月 16 日|Quake.*確定.*情報.*16 日|Quake.*地震.*強震モニタ.*2016|Quake.*地震情報.*2016 年.*4 月 16 日|REAL.*G.*Y.*2016|SOUTHERN.*KUMAMOTO.*PREF.*32|TNT.*239.*0.*g|TNT.*3.*8.*トン|TNT.*84.*8.*g|TNT.*951.*5.*kg|Y.*アニメ.*熊本県.*北西|Y.*熊本県.*熊本.*地方|アニメ.*熊本県.*北西.*部|エムスリー株式会社.*2.*TNT.*951|エムスリー株式会社.*3.*10.*km|エムスリー株式会社.*4.*さ.*10|エムスリー株式会社.*4.*地震.*津波|エムスリー株式会社.*6.*TNT.*3|エムスリー株式会社.*7.*地震.*津波|エムスリー株式会社.*8.*AQUA.*CMT|エムスリー株式会社.*8.*熊本県.*熊本|エムスリー株式会社.*9.*10.*km|エムスリー株式会社.*9.*AQUA.*CMT|エムスリー株式会社.*9.*推定.*津波|エムスリー株式会社.*9.*地震.*津波|さ.*13.*0.*km|さ.*20K.*m.*M4|さ.*4.*0.*km|さ.*6.*0.*km|さ.*7.*7.*km|さ.*8.*0.*km|さ.*9..*6.*km|さ.*9..*8K.*m|マグニチュード.*3.*7.*推定|マグニチュード.*4.*0.*推定|阿蘇.*震度 1.*熊本県.*天草|阿蘇.*地方.*M4.*0|阿蘇.*地方.*M4.*1|雨.*高 2.*4℃.*最低|嘉島町.*震度.*情報.*16 日|気象庁.*情報.*16 日.*07|気象庁.*情報.*16 日.*14|気象庁.*発表.*16 日.*09|気象庁.*発表.*16 日.*14|気象庁.*微小.*地震速報.*熊本県|規模.*エムスリー株式会社.*4.*地震|規模.*エムスリー株式会社.*5.*推定|規模.*エムスリー株式会社.*9.*地震|規模.*マグニチュード.*3.*7|規模.*マグニチュード.*4.*0|宮崎県.*北部.*平野.*部|緊急地震速報.*7.*報.*熊本県|緊急地震速報.*警報.*ID.*20160416142706|緊急地震速報.*予報.*ID.*20160416055256|緊急地震速報.*予報.*ID.*20160416131533|緊急地震速報.*予報.*ID.*20160416153813|緊急地震速報.*予報.*ID.*20160416182513|緊急地震速報.*予報.*ID.*20160416210511|区.*震度.*情報.*16 日|熊本.*確定.*情報.*16 日|熊本.*地方.*10.*6|熊本.*地方.*M4.*5|熊本.*微小.*地震速報.*熊本県|熊本県.*宇城市.*松橋町萩尾.*付近|熊本県.*益城町.*福原.*付近|熊本県.*熊本.*確定.*情報|熊本県.*熊本.*地震情報.*16 日|熊本県.*熊本市.*戸島町.*付近|熊本県.*熊本市.*城南町塚原.*付近|熊本県.*南部.*SOUTHERN.*KUMAMOTO|熊本県.*美里町.*木早川内.*付近|月.*16 日.*05.*時|最高.*22.*℃.*最低|最終.*報.*32.*8|最終.*報.*33.*N|最大.*震度.*3.*36|最大.*震度 3.*エムスリー株式会社.*7|最大.*震度 3.*エムスリー株式会社.*8|最大.*震度 4.*M4.*3|最大.*速度.*0.*3|最低.*12.*℃.*降水確率|産山村.*震度.*情報.*16 日|時.*09.*分.*頃|時.*33.*分.*頃|時.*37.*分.*頃|時.*38.*分.*頃|時.*51.*分.*頃|時.*52.*分.*頃|時.*56.*分.*頃|時.*57.*分.*震源|時.*58.*分.*頃|時.*6.*分.*頃|情報.*16 日.*07.*時|情報.*16 日.*10.*時|情報.*16 日.*13.*時|心配.*jishin.*微小.*地震速報|震源地.*日本.*熊本県.*美里町|震度.*3.*36.*00|震度.*4.*緊急地震速報.*最終|震度 1.*宮崎県.*北部.*平野|震度 1.*大分県.*中部.*大分県|震度 1.*長崎県.*島原半島.*震度 1|震度 3.*G.*Y.*37|震度 3.*G.*Y.*地震速報|震度 3.*エムスリー株式会社.*4.*地震|震度 3.*エムスリー株式会社.*8.*さ|震度 4.*1.*報.*32|推定.*規模.*エムスリー株式会社.*4|推定.*規模.*エムスリー株式会社.*9|推定.*震度 3.*エムスリー株式会社.*8|惣領.*付近.*震度.*Mw|大分県.*西部.*震度 1.*大分県|第 9.*報.*最終.*地震|地震.*規模.*エムスリー株式会社.*9|地震.*規模.*マグニチュード.*5|地震.*強震モニタ.*2016.*04|地震.*震度 3.*熊本県.*阿蘇市|地震.*震度 3.*熊本県.*宇城市|地震.*震度 3.*熊本県.*産山村|地震.*震度 3.*熊本県.*南阿蘇村|地震情報.*2016 年.*4 月 16 日.*3|地震情報.*4 月 16 日.*14.*時|地方.*10.*4.*7|地方.*M4.*3.*さ|地方.*M4.*4.*さ|地方.*M4.*5.*10|地方.*エムスリー株式会社.*3.*10|地方.*エムスリー株式会社.*4.*地震|地方.*エムスリー株式会社.*5.*さ|地方.*エムスリー株式会社.*6.*地震|地方.*エムスリー株式会社.*8.*地震|地方.*エムスリー株式会社.*9.*10|地方.*震度 3.*エムスリー株式会社.*7|地方.*推定.*震度 3.*M4|長崎県.*島原半島.*震度 1.*宮崎県|津波.*心配.*jishin.*微小|度.*震度 4.*熊本県.*熊本|南部.*SOUTHERN.*KUMAMOTO.*PREF|日本.*熊本県.*宇城市.*松橋町萩尾|日本.*熊本県.*益城町.*福原|日本.*熊本県.*熊本市.*戸島町|日本.*熊本県.*熊本市.*城南町塚原|日本.*熊本県.*御船町.*役場|日本.*熊本県.*美里町.*木早川内|発生.*G.*気象庁.*気象庁|発生.*G.*気象庁.*微小|発表.*16 日.*09.*時|発表.*16 日.*14.*時|美里町.*木早川内.*付近.*震度|付近.*エムスリー株式会社.*2.*TNT|付近.*エムスリー株式会社.*6.*TNT|福岡県.*筑後.*震度 1.*長崎県|報.*M4.*5.*熊本県|報.*エムスリー株式会社.*8.*熊本県|報.*最終.*地震.*16 日|木早川内.*付近.*震度.*M|℃.*最低.*9.*℃|0.*4.*Oita.*jishin|0.*kg.*さ.*9.|0.*km.*M4.*0|0.*推定.*地震.*津波|0.*地震.*発生.*震源|0.*分.*頃.*熊本県|00.*JST 株式会社.*日本.*熊本県|01.*33.*20.*熊本県|02.*分.*頃.*熊本県|03.*03.*10.*33|03.*10.*33.*0|03.*55.*53.*発生|03.*時.*16.*分|03.*時.*34.*分|03.*時.*57.*分|04.*05.*49.*32|04.*16.*01.*46|04.*16.*02.*34|04.*16.*06.*11|04.*16.*07.*24|04.*16.*08.*09|04.*16.*14.*04|04.*16.*15.*39|04.*16.*20.*30|04.*23.*18.*33|04.*51.*24.*32|04.*月.*16 日.*13|04.*時.*00.*分|04.*時.*15.*分|04.*時.*23.*分|04.*時.*24.*分|04.*分.*気象庁.*発表|05.*06.*32.*8|05.*06.*発生.*M4|05.*49.*32.*8|05.*52.*51.*33|05.*頃.*熊本県.*熊本|05.*時.*37.*分|05.*分.*気象庁.*発表|06.*32.*8.*130|06.*時.*34.*分|06.*時.*40.*分|07.*時.*49.*分|07.*時.*57.*分|08.*02.*47.*発生|08.*08.*51.*32|08.*51.*発生.*M4|08.*時.*02.*分|08.*分.*気象庁.*発表|09.*51.*発生.*M4|09.*時.*53.*分|1.*10.*km.*エムスリー株式会社|1.*16 日.*01.*時|1.*3K.*g.*さ|1.*推定.*地震.*発生|1.*地震.*津波.*心配|1.*報.*熊本県.*阿蘇|10.*5.*9.*5|10.*5.*km.*MAP|10.*7.*km.*MAP|10.*JST 株式会社.*震源地.*日本|10.*時.*38.*分|11.*1.*km.*MAP|11.*6.*km.*MAP|1193.*微小.*地震速報.*熊本県|12.*0.*km.*MAP|12.*18.*時.*20|12.*時.*59.*分

|13.*15.*28.*32|13.*時.*09.*分|130.*7.*E.*エムスリー株式会社|130.*7.*最大.*震度 4|130.*7.*推定.*M4|130.*7.*度.*震度 4|130.*8.*E.*2|130.*9.*度.*さ|131.*0.*熊本県.*熊本|131.*1.*最大.*震度 4|131.*1.*推定.*エムスリー株式会社|131.*2.*E.*0|131.*2.*度.*震度 3|14.*03.*57.*発生|14.*27.*01.*32|14.*時.*32.*分|14.*時.*47.*分|15.*28.*32.*9|15.*38.*38.*32|15.*JST 株式会社.*日本.*熊本県|15.*分.*震源.*熊本県|16.*03.*32.*41|16.*03.*55.*52|16.*04.*00.*41|16.*04.*15.*03|16.*04.*51.*25|16.*13.*15.*26|16.*18.*25.*12|16.*20.*44.*00|16 日.*02.*時.*57|16 日.*03.*時.*16|16 日.*04.*時.*18|16 日.*07.*時.*01|16 日.*07.*時.*23|16 日.*09.*時.*16|16 日.*10.*時.*26|16 日.*14.*時.*27|16 日.*15.*時.*48|16 日.*20.*時.*44|16 日.*22.*時.*06|16 日.*22.*時.*10|17.*JST 株式会社.*日本.*熊本県|17.*時.*05.*分|17.*時.*17.*分|18.*時.*20.*18|18.*時.*25.*分|2.*4.*0.*0|2.*7.*E.*131|2.*7.*トン.*さ|2.*8.*E.*131|2.*8.*最大.*速度|2.*AQUA.*REAL.*M4|2.*kg.*さ.*9.|2.*最大.*震度.*4|2.*推定.*地震.*発生|2.*度.*さ.*10|2.*度.*震度 3.*熊本県|20.*18.*24 時.*%|20.*JST 株式会社.*日本.*熊本県|20.*分.*気象庁.*発表|2016.*4.*16.*8|2016.*4.*16.*9|20160416014601.*2016.*04.*16|20160416072359.*2016.*04.*16|20160416080834.*2016.*04.*16|20160416140403.*2016.*04.*16|20160416153813.*2016.*04.*16|20160416204405.*2016.*04.*16|20160416214441.*2016.*04.*16|2016 年.*4 月 16 日.*21.*時|20K.*m.*M.*5|20K.*m.*saigai.*jishin|21.*05.*06.*32|22.*時.*06.*分|23.*18.*33.*0|23.*54.*発生.*M4|25.*JST 株式会社.*日本.*熊本県|28.*32.*9.*131|28.*JST 株式会社.*震源地.*日本|3.*16 日.*04.*時|3.*4.*推定.*地震|3.*5.*推定.*地震|3.*5.*地震.*発生|3.*TNT.*1.*3K|3.*緊急地震速報.*1.*報|3.*最大.*速度.*0|3.*推定.*諏訪地域.*揺れ|3.*報.*M.*5|32.*6.*131.*0|33.*1.*N.*131|33.*JST 株式会社.*日本.*熊本県|34.*頃.*熊本県.*熊本|38.*32.*6.*131|38.*38.*32.*6|38.*分.*頃.*熊本県|3K.*m.*M2.*5|4.*10.*km.*震度|4.*5.*推定.*地震|4.*Oita.*jishin.*大分|4.*最大.*震度 3.*地震|4.*推定.*地震.*津波|4.*推定.*津波.*心配|4.*報.*M.*5|45.*55.*発生.*最大|47.*分.*気象庁.*発表|47.*分.*頃.*熊本県|4K.*m.*2016.*04|4K.*m.*M.*5|4 月 16 日.*17.*時.*40|5.*16 日.*22.*時|5.*2.*AQUA.*REAL|5.*2.*さ.*10|5.*5.*熊本県.*熊本|5.*5.*最大.*震度|5.*8.*地震.*発生|5.*9.*5.*強|5.*9.*最大.*震度|5.*M2.*5.*熊本県|5.*熊本県.*北西部.*さ|5.*最大.*震度 4.*地震|5.*弱.*緊急地震速報.*警報|5.*推定.*最大.*震度|5.*推定.*諏訪地域.*揺れ|5.*地震.*発生.*最大|5.*予想.*発生.*時刻|51.*24.*32.*8|51.*32.*9.*130|51.*33.*0.*131|51.*発生.*M4.*3|51.*発生.*M4.*5|52.*51.*33.*0|52.*51.*発生.*エムスリー株式会社|52.*熊本県.*北東部.*NE|54.*32.*8.*130|55.*32.*9.*130|57.*頃.*発生.*震源地|58.*分.*気象庁.*発表|6.*131.*0.*熊本県|6.*7.*km.*MAP|6.*kg.*さ.*10|6.*km.*M2.*7|6.*推定.*地震.*発生|6.*報.*M.*5|6.*報.*最終.*震源|6.*報.*最終.*地震|673.*6.*g.*さ|7.*1.*さ.*10|7.*10.*km.*震度|7.*4.*0.*0|7.*4K.*m.*MAP|7.*E.*131.*1|7.*kg.*さ.*11|7.*最大.*震度 3.*地震|7.*推定.*地震.*発生|7.*度.*震度 4.*熊本県|7.*微小.*地震速報.*熊本県|7.*報.*熊本県.*熊本|7.*報.*最終.*震源|8.*E.*130.*7|8.*最大.*震度.*4|8.*最大.*速度.*0|860.*km.*上越市.*影響|8K.*g.*さ.*10|9.*5.*強.*0|9.*kg.*さ.*9.|9.*km.*M4.*2|9.*最大.*震度.*5|9.*度.*東経.*131|9.*分.*頃.*熊本県|AQUA.*REAL.*2.*SA|AQUA.*REAL.*強震モニタ.*地震|AQUA.*REAL.*速報.*LV|E.*10.*3K.*m|E.*10.*9.*km|E.*11.*1.*km|E.*20K.*m.*地震|E.*4.*0.*km|E.*7.*7.*km|E.*8.*6.*km|EARTH.*Quake.*04.*月|EARTH.*Quake.*4.*報|eqjp.*2016.*04.*16|G.*Y.*緊急地震速報.*4|G.*Y.*緊急地震速報.*6|g.*さ.*9..*8K|in.*熊本市.*熊本県.*微小|jishin.*EARTH.*Quake.*04|jishin.*EARTH.*Quake.*1|jishin.*EARTH.*Quake.*最大|jishin.*災害.*saigai.*16 日|kg.*さ.*10.*0|kg.*さ.*10.*2|kg.*さ.*10.*6|kg.*さ.*13.*0|kg.*さ.*8.*6|kg.*さ.*9..*1|km.*EARTH.*Quake.*4|km.*EARTH.*Quake.*強震モニタ|km.*EARTH.*Quake.*地震速報|km.*M.*5.*9|km.*M2.*7.*微小|km.*M4.*5.*AQUA|km.*エムスリー株式会社.*3.*16 日|km.*エムスリー株式会社.*5.*16 日|km.*エムスリー株式会社.*5.*AQUA|km.*エムスリー株式会社.*7.*AQUA|km.*エムスリー株式会社.*8.*16 日|km.*マグニチュード.*3.*5|km.*震度.*3.*1|km.*地震.*規模.*M4|LV.*1.*16 日.*01|LV.*1.*16 日.*04|M.*5.*0.*さ|M.*5.*2.*AQUA|M.*5.*3.*地震|M.*5.*5.*熊本県|M.*5.*5.*最大|M.*5.*9.*熊本県|M.*5.*9.*最大|M.*7.*1.*さ|m.*M4.*0.*AQUA|m.*M4.*1.*AQUA|M1.*1.*TNT.*673|M1.*3.*TNT.*1|M2.*5.*M2.*5|M2.*7.*熊本県.*北西部|M2.*7.*微小.*地震速報|M4.*0.*熊本県.*熊本|M4.*0.*地震.*発生|M4.*1.*10.*km|M4.*1.*地震.*津波|M4.*2.*AQUA.*CMT|M4.*2.*最大.*震度|M4.*2.*地震.*発生|M4.*5.*AQUA.*REAL|M4.*5.*地震.*津波|M4.*6.*さ.*10|M4.*8.*最大.*震度|M6.*3.*さ.*10|MAP.*1193.*微小.*地震速報|N.*131.*108.*E|ND.*20160416014601.*2016.*04|ND.*20160416072359.*2016.*04|ND.*20160416080834.*2016.*04|ND.*20160416140403.*2016.*04|ND.*20160416153813.*2016.*04|ND.*20160416204405.*2016.*04|ND.*20160416214441.*2016.*04|NET.*強震モニタ.*緊急地震速報.*予報|Quake.*04.*月.*16 日|Quake.*4.*報.*2016|Quake.*緊急地震速報.*最終.*報|Quake.*地震.*強震モニタ.*微小|REAL.*2.*SA.*1015|REAL.*強震モニタ.*地震.*発生|TNT.*1.*3K.*g|TNT.*2.*7.*トン|TNT.*673.*6.*g|Y.*緊急地震速報.*4.*報|Y.*緊急地震速報.*6.*報|エムスリー株式会社.*2.*地震.*発生|エムスリー株式会社.*4.*10.*km|エムスリー株式会社.*4.*最大.*震度 3|エムスリー株式会社.*4.*推定.*津波|エムスリー株式会社.*5.*10.*km|エムスリー株式会社.*5.*TNT.*2|エムスリー株式会社.*6.*推定.*地震|エムスリー株式会社.*6.*推定.*津波|エムスリー株式会社.*7.*10.*km|エムスリー株式会社.*7.*推定.*地震|エムスリー株式会社.*7.*推定.*津波|さ.*10.*1.*km|さ.*10.*4K.*m|さ.*10.*5.*km|さ.*10.*8K.*m|さ.*11.*0.*km|さ.*11.*6.*km|さ.*11.*9.*km|さ.*12.*0.*km|さ.*12.*1.*km|さ.*12.*7.*km|さ.*13.*2.*km|さ.*20K.*m.*saigai|さ.*3.*3K.*m|さ.*6.*7.*km|さ.*7.*4K.*m|さ.*8.*9.*km|さ.*9..*2.*km|さ.*9..*4K.*m|さ.*9..*5.*km|マグニチュード.*3.*5.*地震|マグニチュード.*4.*1.*推定|マグニチュード.*5.*8.*推定|阿蘇.*地方.*震度 4.*M4|阿蘇.*地方.*推定.*震度 3|一.*時.*雨.*高 2|確定.*情報.*16 日.*14|気象庁.*震源.*情報.*07|気象庁.*発表.*16 日.*17|気象庁.*発表.*16 日.*22|規模.*M.*5.*3|規模.*M.*5.*8|規模.*エムスリー株式会社.*6.*地震|規模.*エムスリー株式会社.*7.*推定|規模.*エムスリー株式会社.*7.*地震|規模.*マグニチュード.*4.*1|規模.*マグニチュード.*5.*8|菊陽町.*辛川.*付近.*M2|距離.*808.*km.*上越市|距離.*845.*km.*上越市|距離.*860.*km.*上越市|強震モニタ.*速報.*LV.*1|強震モニタ.*速報.*LV.*3|強震モニタ.*微小.*地震速報.*熊本県|緊急地震速報.*警報.*3.*報|緊急地震速報.*警報.*4.*報|緊急地震速報.*警報.*6.*報|緊急地震速報.*警報.*ID.*20160416014601|緊急地震速報.*発表.*震度.*情報|緊急地震速報.*予報.*ID.*20160416072359|緊急地震速報.*予報.*ID.*20160416080834|緊急地震速報.*予報.*ID.*20160416140403|緊急地震速報.*予報.*ID.*20160416204405|緊急地震速報.*予報.*ID.*20160416214441|区.*熊本県熊本市東区.*熊本県熊本市西区.*熊本県熊本市南区|区.*熊本県熊本市北区.*区.*玉名市|熊本.*震度 2.*熊本県.*天草|熊本.*地方.*M4.*2|熊本.*地方.*M4.*4|熊本.*地方.*さ.*M2|熊本.*地方.*推定.*震度 4|熊本県.*阿蘇.*熊本県.*球磨|熊本県.*阿蘇.*地方.*32|熊本県.*宇城市.*豊野町山崎.*付近|熊本県.*益城町.*赤井.*付近|熊本県.*嘉島町.*井寺.*付近|熊本県.*菊陽町.*戸次.*付近|熊本県.*熊本.*地方.*M2|熊本県.*熊本県.*北西.*部|熊本県.*熊本県熊本市西区.*熊本県熊本市南区.*区|熊本県.*熊本市.*城南町東阿高.*付近|熊本県.*御船町.*小坂.*付近|熊本県.*御船町.*役場.*西|熊本県.*産山村.*震度.*情報|熊本県.*北西部.*さ.*10|熊本県.*北西部.*さ.*8|熊本県.*北東部.*さ.*0|熊本県熊本市西区.*熊本県熊本市南区.*区.*熊本県熊本市北区|熊本市.*熊本県.*微小.*地震速報|熊本市.*城南町藤山.*付近.*M2|熊本市.*富合町木原.*付近.*震度|計測.*震度.*2.*8|警報.*4.*報.*M|警報.*6.*報.*M|月.*16 日.*13.*時|御船町.*役場.*西.*3K|頃.*S 波.*到達.*予定|頃.*震源

```
地.*距離.*808|頃.*震源地.*距離.*845|頃.*震源地.*距離.*860|最大.*震度.*3.*4|最大.*震度 3.*エムスリー株式会社.*5|最大.*震度 3.*エムスリー株式会社.*6|最大.*震度 4.*M4.*5|最大.*速度.*0.*2|最大.*速度.*0.*5|産山村.*田尻.*付近.*M2|産山村.*田尻.*付近.*エムスリー株式会社|時.*0.*分.*頃|時.*00.*分.*頃|時.*01.*分.*頃|時.*04.*分.*気象庁|時.*05.*分.*気象庁|時.*08.*分.*気象庁|時.*11.*分.*頃|時.*15.*分.*震源|時.*20.*18.*24 時|時.*20.*分.*気象庁|時.*28.*分.*頃|時.*42.*分.*頃|時.*43.*分.*頃|時.*47.*分.*気象庁|時.*58.*分.*気象庁|時.*9.*分.*頃|時.*雨.*高 2.*4℃|心配.*jishin.*地震情報.*16 日|心配.*微小.*地震速報.*熊本県|震源地.*距離.*808.*km|震源地.*距離.*845.*km|震源地.*距離.*860.*km|震度.*2.*8.*最大|震度.*3.*4.*報|震度.*3.*緊急地震速報.*1|震度.*情報.*16 日.*15|震度.*分離.*ため.*こ|震度 1.*熊本県.*阿蘇.*熊本県|震度 2.*熊本県.*熊本.*震度 1|震度 2.*大分県.*西部.*震度 1|震度 2.*福岡県.*筑後.*震度 1|震度 3.*熊本県.*阿蘇.*地震情報|震度 3.*熊本県.*宇城市.*震度|震度 3.*熊本県.*熊本県熊本市西区.*区|震度 3.*最終.*報.*32|震度 3.*大分県.*西部.*震度 2|震度 4.*M4.*1.*地震|震度 4.*熊本県.*熊本.*震度 2|震度 4.*熊本県.*熊本県熊本市中央区.*区|推定.*エムスリー株式会社.*6.*推定|推定.*エムスリー株式会社.*7.*推定|推定.*規模.*M.*5|推定.*規模.*エムスリー株式会社.*6|推定.*規模.*エムスリー株式会社.*7|推定.*地震.*発生.*saigai|晴.*最高.*21.*℃|速度.*0.*4.*Oita|第 9.*報.*最終.*震源|地震.*2016.*04.*16|地震.*規模.*エムスリー株式会社.*4|地震.*規模.*エムスリー株式会社.*6|地震.*規模.*エムスリー株式会社.*7|地震.*強震モニタ.*微小.*地震速報|地震.*緊急地震速報.*発表.*震度|地震.*震度 4.*熊本県.*熊本県熊本市中央区|地震.*津波.*心配.*微小|地震.*発生.*saigai.*jishin|地震情報.*2016 年.*4 月 16 日.*21|地震情報.*4 月 16 日.*06.*時|地震情報.*4 月 16 日.*11.*時|地方.*10.*4.*2|地方.*10.*5.*9|地方.*M.*5.*8|地方.*M4.*1.*10|地方.*エムスリー株式会社.*4.*10|地方.*エムスリー株式会社.*5.*10|地方.*エムスリー株式会社.*7.*10|地方.*エムスリー株式会社.*7.*地震|地方.*震度 3.*エムスリー株式会社.*6|地方.*震度 4.*M4.*5|中.*地震.*緊急地震速報.*発表|長崎.*東.*90.*km|津波.*心配.*jishin.*気象庁|津波.*心配.*jishin.*熊本県|津波.*心配.*jishin.*最大|津波.*心配.*微小.*地震速報|天草.*芦北.*震度 3.*大分県|度.*震度 4.*熊本県.*震度 3|島原半島.*震度 1.*宮崎県.*北部|東.*90.*km.*付近|日本.*熊本県.*宇城市.*豊野町山崎|日本.*熊本県.*宇城市.*役所|日本.*熊本県.*益城町.*赤井|日本.*熊本県.*嘉島町.*井寺|日本.*熊本県.*菊陽町.*戸次|日本.*熊本県.*熊本市.*秋津町|日本.*熊本県.*熊本市.*城南町東阿高|日本.*熊本県.*御船町.*小坂|日本.*熊本県.*大津町.*付近|発生.*G.*気象庁.*最大|発生.*M4.*2.*さ|発生.*M4.*7.*さ|発生.*M4.*8.*さ|発生.*saigai.*jishin.*EARTH|発表.*16 日.*17.*時|発表.*16 日.*22.*時|発表.*お待ち.*eqjp.*熊本県|発表.*中.*地震.*緊急地震速報|付近.*M.*0.*5|付近.*M.*0.*8|付近.*M1.*1.*TNT|付近.*M1.*2.*TNT|付近.*M1.*3.*TNT|付近.*エムスリー株式会社.*5.*TNT|付近.*震度.*弱.*Mw|報.*M.*5.*5|報.*M.*5.*9|報.*エムスリー株式会社.*5.*熊本県|報.*エムスリー株式会社.*6.*熊本県|報.*エムスリー株式会社.*9.*熊本県|役場.*西.*3K.*m/)'
```

表 3 - No.1 test4yy.py

```
# -*- coding: utf-8 -*-

from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from pymongo import Connection
import time
import datetime
import calendar
import json

consumer_key = "***"
consumer_secret = "***"

access_token = "***"
access_token_secret = "***"

class StdOutListener(StreamListener):
    def __init__(self):
        super(StdOutListener,self).__init__()
        self.count = 0
        self.start_at = datetime.datetime.now()
        ### MongDB へのアクセス
        #コネクション作成
        self.con = Connection('localhost', 27017)
        #コネクションから tweet データベースを取得
        self.db = self.con.tweety

    def __del__(self):
        self.con.disconnect()

    def on_data(self, data):
        if data.startswith("{"):
          temp = json.loads(data)
          #print len(temp)

          #空（改行）メッセージを無視する
          if len(temp) == 1:
            #print "空"
            return True

          # created_at 日付フォーマット変換
```

```
            temp2 = calendar.timegm(time.strptime(temp["created_at"],"%a %b %d %H:%M:%S +0000 %Y"))
            temp3 = calendar.timegm(time.strptime(temp["user"]["created_at"],"%a %b %d %H:%M:%S +0000 %Y"))

            # JST に変換(ISO/W3C フォーマット)
            temp["created_at"] = time.strftime("%Y-%m-%dT%H:%M:%S+09:00",time.localtime(temp2))
            temp["user"]["created_at"] = time.strftime("%Y-%m-%dT%H:%M:%S+09:00",time.localtime(temp3))

            #print temp["created_at"]
            if (datetime.datetime.now() - self.start_at).seconds < 60:
               self.db.tweet_2.insert(temp)
               self.count += 1
            else:
               print "60 秒処理したので抜けます",
               print self.count,
               print datetime.datetime.now()
               # pymongo から終了する場合
               return False
            return True

    def on_error(self, status):
        print status
        return True    # keep stream alive

    def on_timeout(self):
        print 'Snoozing Zzzzzz'

if __name__ == '__main__':
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)

    stream = Stream(auth, l)
#     stream.filter(track = [keyword])#検索する場合
#     stream.sample()
    stream.filter(locations=[123,24,146.2,39.0,138.4,33.5,146.1,46.20])
```

表 3 - No.2 get_message_prod.sh

```
#!/usr/bin/env bash
while :
do
  echo "処理を開始しました"
  python test4yy.py
  echo "30 秒 Sleep します"
  sleep 30
done
```

表 3 - No.3 startup_prod.sh

```
#!/usr/bin/env bash
ulimit -n 2048
mongod --fork --dbpath /Volumes/HDS2-UT/data --logpath /Volumes/HDS2-UT/logs/mongod.log --rest
sleep 15
get_message_prod.sh >> out.log 2>> err.log < /dev/null &
sleep 15
system_manage.py
launchctl unload manage_tweet.plist
launchctl load manage_tweet.plist
```

表 3 - No.4 system_manage.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import smtplib
from email.MIMEText import MIMEText
from email.Utils import formatdate
from email.Header import Header

# メール送信
def sendmail(data):
```

```
    # gmail にログインするときのメールアドレス
    gmail_address = '***'
    # gmail にログインするときのパスワード
    gmail_passwd = '***'

    # gmail の SMTP サーバアドレス
    gmail_smtp_address = 'smtp.gmail.com'
    # gmail の SMTP サーバポート番号
    gmail_smtp_port = 587

    # 送信元メールアドレス
    from_address = '***'
    # 送信先メールアドレスのリスト
    to_address = ['***', ]

    # 日本語の処理
    # こちらを参考
    # http://blog.livedoor.jp/yawamen/archives/51566670.html
    # http://d.hatena.ne.jp/yutakikuchi/20110106/1294337077

    # 件名
    # 文字列を utf-8 に変換
    subject = u'★つぶやき取得テータス★'

    # メール本文
    # utf8 から unicode に変換
    body = data.decode('utf-8')
    #print body

    # unicode から iso-2022-jp に変換
    msg = MIMEText(body.encode('iso-2022-jp'), 'plain', 'iso-2022-jp')

    msg['From'] = from_address
    msg['To'] = ', '.join(to_address)
    msg['Date'] = formatdate()
    msg['Subject'] = Header(subject.encode('iso-2022-jp'), 'iso-2022-jp')

    smtpobj = smtplib.SMTP(gmail_smtp_address, gmail_smtp_port)
    smtpobj.ehlo()
    smtpobj.starttls() # こっから SSL
    smtpobj.ehlo()
    smtpobj.login(gmail_address, gmail_passwd) # ログイン
    smtpobj.sendmail(from_address, to_address, msg.as_string()) # 送信
    smtpobj.close()

# unix コマンドを実行
import commands

status_1 = commands.getoutput("mongo tweety --quiet --eval 'db.tweet_2.find().count();'")


#status = commands.getoutput("tail out.log")

status_2 = commands.getoutput("tail out.log")

# print status

#エラーがあればメール送信
#if error_detected:
sendmail(status_1 + "¥n" + status_2);
```