一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
TL2016-10,NLC2016-10(2016-06)

Twitterから抽出された不具合情報の可視化と分析

諸正　梨乃† 　栗原　光平†† 　嶋田　和孝†††

† 九州工業大学 情報工学部 知能情報工学科
†† 九州工業大学 大学院情報工学府 情報科学専攻
††† 九州工業大学 大学院情報工学研究院 知能情報工学研究系
〒 820–8502 福岡県飯塚市川津 680-4
E-mail: †††shimada@pluto.ai.kyutech.ac.jp

あらまし　製品の不具合に関する情報を収集することは，事故や社会的損失を未然に防ぐためにも重要である．本研究では，Twitter を情報源とした不具合情報抽出の枠組みとその結果を分析するための可視化手法について報告する．不具合抽出としては少数のシードからブートストラップ的に不具合表現を含む文を不具合文として抽出する．次に，その抽出された不具合文に対して，クラスタリングを行い，得られたクラスタ情報を基に，3 段階の可視化を行う．可視化された結果を分析し，本データに対する可視化の有効性について検証する．

キーワード　不具合情報, Twitter, 抽出, 可視化

Visualization of trouble information extracted from Twitter

Rino MOROMASA†, Kohei KURIHARA††, and Kazutaka SHIMADA†††

† School of Computer Science and Systems Engineering
†† Graduate School of Computer Science and Systems Engineering
††† Department of Artificial Intelligence
Kyushu Institute of Technology,
680-4 Kawazu Iizuka Fukuoka, 820-8502 Japan
E-mail: †††shimada@pluto.ai.kyutech.ac.jp

Abstract　The World Wide Web contains a huge number of online documents that are easily accessible. One of the important information for business companies is trouble information of a product as the risk management. In this paper, we report a method for visualization of trouble information extracted from Twitter by using a bootstrapping approach. First, we apply the tweets with trouble information into a clustering method. Then, we visualize the clustered tweets by using a three-stage approach. Finally, we analyze the trouble information with the visualization method, and then discuss the results.

Key words　Trouble information, Twitter, Extraction, Visualization

1. はじめに

自動車のリコールなどに代表されるように，製品の不具合は大きな社会的損失に結びつき，時には重大な事故等につながることもある．メーカーや企業は不具合の発生を防ぐため，過去の不具合事例等の情報を製品製造に取り入れ，信頼性の向上に活用している [3]．安全な製品の開発を支援するためにも，不具合に関する情報を多く収集することは重要である．

製品の不具合情報収集に関連する研究として，新聞を対象に交通事故の記事から事故の原因となる表現や関連情報を抽出する研究 [6] や，不具合事例文から製品・部品を示す語を抽出する研究 [5] などが行われている．しかしながら，新聞などの一般メディアを対象とした場合，不具合の発生から記事として公開されるまでの時差の問題や一般メディアには出現しない不具合事例が多く存在する可能性があるなどの問題がある．また，その他の情報源として，公的組織が独自に不具合情報を収集し，不具合事例集として情報を保持している場合がある．それらを用いれば不具合について詳細な情報を得ることができるものの，公的組織の詳細な調査に基づき作成・公開されているものであることから，データ数に限りがあり，追加収集も困難であるという問題がある．

本研究ではそれらの欠点を補うために，個人が自由に情報

を発信することができる CGM (Consumer Generated Media) に着目し，その中でも情報源に Twitter を用いた情報抽出手法を試みる．Kurihara and Shimada [2] は Twitter(注1) を情報源とし，自動で製品の不具合について述べられた文 (以下，不具合文) の抽出を行っている．しかしながら，Kurihara らの研究では製品の不具合文の抽出のみを目的としており，抽出された不具合文から得られた情報の有効な活用法については述べられていない．抽出された不具合情報は，どのような不具合の症状があるのかといった特徴や，どのような不具合が起こりやすいのかといった傾向を把握することで初めて情報が有効に活用されたといえる．一方で，抽出される不具合情報は膨大なため，テキストそのものを人手で分析することには一般に大きなコストが掛かり，分析を支援するようなシステムが望まれる．分析を支援する手法として，要約や可視化といった様々な手法が挙げられる．

そこで，今回は可視化に注目する．視覚的にわかりやすい可視化を行うことは，人手による分析を行う際にもユーザへの負担がより少なくなる．情報の可視化に関する研究は盛んに行われており，具体的には検索作業の負担の軽減，特定の情報パターンの検出支援 [10] といった様々な目的で可視化手法が考えられてきた．情報の可視化を行うことで膨大な量の情報全てに目を通すことなく，重要な情報だけに集中することができる．

本研究では，Kurihara らの手法（以降，先行研究と呼ぶ）によって得られた不具合情報に対して視覚的にわかりやすい可視化を行い，人手による分析を支援するシステムを作成する．具体的には，3 段階に分けて可視化を行うことで直感的にわかりやすく，かつ具体的な製品の不具合の症状を把握していく．3 段階には，それぞれどういった不具合の症状があるのか，その不具合はどのような状況で起きているのか，その状況を示す不具合文をそれぞれ把握できるように分ける．提案手法としてまず，先行研究によって抽出された膨大な不具合文に対して，クラスタリングを行う．そしてクラスタ内の単語に重み付けを行い不具合の症状を表す単語を獲得し，獲得した単語を基に可視化を行う．

本稿は大きく分けて 2 つのパートで構成される．1 つめは不具合情報抽出についてである．不具合情報抽出は，前述の先行研究 [2] に基づいて実行される．2 つめは可視化であり，先行研究で得られた不具合文について，その可視化手法と可視化によって得られた分析結果などを論じる．

2. 不具合情報抽出

2.1 概　　要

多様な不具合文を抽出するためには，未知の不具合表現を自動で獲得する必要がある．不具合表現は，一般的に不具合文に多く出現し，1 つの不具合文中に複数出現することもある．よって，不具合文中から未知の不具合表現をブートストラップ的に獲得することで，より多様な不具合文の自動抽出が可能となると考えられる．不具合表現「壊れた」を基に抽出した不具合文

(注1)：https://twitter.com/

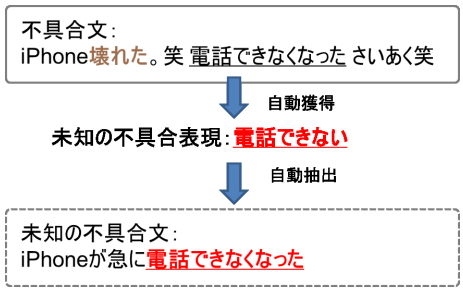

図 1　不具合文から未知の不具合表現を獲得する例

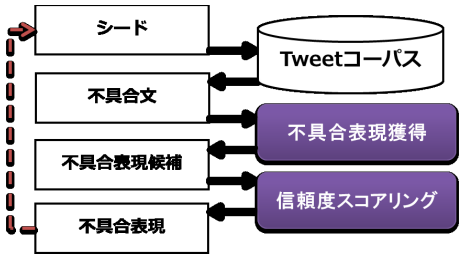

図 2　ブートストラップ法の処理の流れ

から，未知の不具合表現を獲得する例を図 1 に示す．

例のようにして得られた未知の不具合表現を用いて，再度不具合文の抽出を行う．これを繰り返していくことで不具合表現を自動で獲得しつつ，不具合文を抽出する．

2.2 ブートストラップ法の処理の流れ

提案手法で行うブートストラップ法の処理の流れを図 2 に示す．まず初めに，極めて強い不具合らしさを持つような不具合表現を人手で与え，これをシードとして 1 回目の不具合文抽出を行う．次に，抽出された不具合文から新たな不具合表現を獲得し，それらについて信頼度のスコアリングを行う．最後に，スコアをもとに信頼度の高い不具合表現のみを選出し，それらを次のステップの入力（シード）として与え，再び不具合文を抽出する．この流れを繰り返していくことで，最初に与えた少数の不具合表現から，多様な不具合文を自動で抽出することができる．この処理を一定の条件下で繰り返す．

2.3 不具合表現の獲得

不具合文からの不具合表現の獲得は，品詞のパターンマッチを用いたルールベースの手法により行う．獲得ルールは，Twitter 上の不具合情報を人手で分析して得られた 3 つの特徴を基にして作成した．それぞれの特徴とそれにもとづいた不具合表現獲得ルールについて，以下で詳しく述べる．

2.3.1 特定の副詞に着目したルール

「勝手に」や「突然」などといった，予期せぬ出来事や意図せぬ動作などに関連する副詞は，不具合文にも出現しやすく，不具合と強い関連性を持つと考えられる．また，そのような特徴的な記述に着目した問題発見手法も提案されている [4]．あらかじめ不具合に関連の強いと思われる副詞を人手で設定しておき，それらと一緒に出現した動詞や名詞のまとまりを不具合表現として獲得する．今回は「勝手に」,「急に」,「突然」,「いきなり」を利用する．副詞に着目したルールで不具合表現を獲得する例を図 3 に示す．

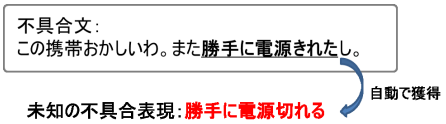

図 3　副詞に着目した不具合表現獲得の例

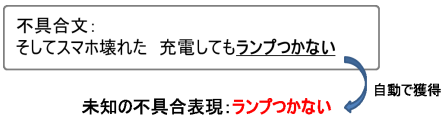

図 4　動詞の未然形に着目した不具合表現獲得の例

2.3.2　動詞の未然形に着目したルール

製品の不具合で最も多く見られるのが,「動かない」や「起動しない」などといった，動詞の未然形に「～ない」という否定の助動詞が連続する表現である．この事実に基づき，不具合文中に出現する動詞の未然形と否定の助動詞の組み合わせを不具合表現として獲得する．動詞の未然形に着目して不具合表現を獲得する例を図 4 に示す．

2.3.3　特定のネガティブな語に着目したルール

製品の異常や不具合の症状などの記述には,「悪い」や「破れる」などといった，ネガティブな単語が出現しやすい．この傾向にもとづき，不具合文中のネガティブな語の付近に出現する語を不具合表現として獲得する．ただし，全てのネガティブな語が不具合に関連するわけではなく，不具合とは無関係なネガティブな単語も多く存在する．そこで，今回は既存の評価表現辞書などは使わず，特に不具合に関連の強いと思われるネガティブな語をあらかじめ人手で与え，それらを利用して不具合表現を獲得する．今回は「悪い」,「おかしい」,「故障」,「異常」,「やばい」をネガティブな語とした．

2.4　信頼度のスコアリング

ブートストラップ法では，シードにより得られた出力を次の入力に用いるという特性上，一度適合率の低い結果が出力の中に含まれてしまうと連鎖的に適合率が低下してしまうという問題があり，意味ドリフトと呼ばれている [1]．これを解決するためには，毎回の出力の精度をできる限り高く保ち，適合率の低いシードを次の入力に含まないようにすることが重要となる．

そこで，提案手法では獲得した不具合表現について信頼度のスコアリングを行い，より不具合らしさが強いと思われるものだけを次の入力として利用する．スコアリングの方法として，まず不具合文に出現する単語（動詞・名詞・形容詞）についてそれぞれに信頼度を計算し，それらの値を用いて不具合表現の信頼度を算出する．ここで，不具合文に多く出現し，なおかつ不具合の対象となる製品名の近くに出現しやすい単語は，不具合らしさが強いと仮定し，ある単語 w の信頼度を以下の式で計算する．

$$score(w) = \sum_{i \in I} \frac{1}{dist(i)} \qquad (1)$$

ここで I は単語 w が出現する文の集合であり，$dist(i)$ は不具合の対象となる製品名と，単語 w 間の距離である．なお，不具合対象となる製品名については，あらかじめ人手で設定してあるものとする．

以上のようにして求められた単語ごとの信頼度をもとに，不具合表現の信頼度を算出する．不具合表現の信頼度は，含まれる単語の持つスコアの平均で求められる．よって，不具合らしさの強い単語が多く出現する不具合表現ほど，より高い信頼度が得られる．

不具合表現の信頼度が求まったあと，全ての不具合表現をスコアが高い順にソーティングし，上位 N% のものだけを有効なものとして次回の入力に利用する．この N の値は人手で事前に設定しておく．N の値が厳しければ厳しいほど高い適合率が期待できるが，反対に次のターンの入力として用いる不具合表現の数は減少するため，再現率は低くなる．なお，反復の際には前回獲得した不具合表現の情報を保持しておき，上位 N% の中に新規に獲得した不具合表現が含まれていなかった場合はそこで反復を終了する．

2.5　不具合文の抽出

文中に不具合表現が出現しているかどうかを判定し，出現していれば不具合文として抽出する．ただし，より事実性の高い文だけを抽出するために，ノイズ除去のためのルールをあらかじめ人手で設定しておき，不要な文の抽出を防ぐ．ルールの例としては，不具合表現の後ろに「言っていた」「聞いた」などの伝聞に関する言い回しや,「気がした」のような事実性のない言い回しが出現した場合には抽出しない，などがある．

2.6　不具合文抽出の結果

不具合対象となる製品には携帯電話・スマートフォンを設定し，データセットとして携帯・スマートフォンに関連するツイート 10 万件を用いた (注2)．最初に人手で与えるシードには次の 7 種類の語を用いた．

シード

壊れる，おかしい，異常，故障，破損，フリーズ，バグ

以上のシードとツイート群についてブートストラップによる不具合文抽出を実行し，得られた結果の正しさを人手で確認し，適合率によって評価した．今回は 5 回目で反復時に新規の不具合表現が獲得されなくなり，2.4 節の条件を満たしたため，繰り返し処理が停止した．

結果としては 3,249 ツイートが不具合文として抽出され，その結果を人手で判定したところ，2,623 ツイートが不具合文であった．すなわち，精度は 80.7%であった．以下は獲得された不具合文の例である．

不具合文

・充電切れるわケータイ熱くなっちゃって充電できないわ勝手に電源切れるわ最悪です
・てか携帯画面真っ暗になって電池パック抜いて電源入れようとしても電源つかないんだけど

一方で，精度から分かるように，20%は誤りであった．誤りの要因はいくつか存在するが，典型的な例としては以下のような

(注2)：具体的には，スマートフォン，携帯電話およびそれらを示す単語が出現している文を収集した

ものがある．

誤抽出の例

・iPhone の充電ケーブル壊れた（；Д｀）充電できない（；Д｀）
・携帯の充電器がおかしいうまく充電されない

これらは，不具合情報であるが，今回対象とした「携帯電話・スマートフォン」そのものの不具合ではないため，今回の実験では誤りに含まれている．より詳細な結果については，Kurihara and Shimada [2] を参照して欲しい．

3. 可 視 化

3.1 概　　要

2. 節で述べた不具合文抽出手法では，10 万ツイートから約 80%の精度で約 3,200 件のツイートが不具合文として抽出された．抽出されたこの膨大な量の不具合文には，同じ不具合の症状について述べている文が複数存在している．またこれらの不具合文中には，不具合の症状と個人の意見や感想といった不具合の症状とは関係のない冗長な要素が混在している場合が多く見られる．これらは Twitter から抽出された不具合情報ならではの問題点であり，人手による分析が困難になる原因でもある．このような問題に対して，抽出された不具合文を不具合の症状ごとにまとめ，まとまった不具合の症状から不具合の症状を表す要素のみを獲得することで，人手による分析を支援することができる．さらに，その約 3200 件の中には 20 %程度の誤りが含まれている．分析をしやすい環境は抽出手法そのもののエラー分析にも有効であると考えられる．

そこで，本研究では可視化を行う前処理として，抽出された不具合文から代表的な不具合の症状を表す単語を獲得する．不具合の症状を表す代表的な単語を獲得することで，より人手による分析が容易になると考えられる．この処理にはクラスタリングを用いる．そして，獲得された単語をもとに 3 段階での可視化を行う．

3.2 不具合の症状を表す単語の獲得

効率的に可視化およびその結果の分析を行うには，不具合文が症状ごとに分類されていることが望ましい．そこで，不具合のそれぞれの症状はそれぞれに関係する特徴的な表層表現によって記述されていると仮定し，前節で得られた不具合文（ただし，誤りを含む）をクラスタリングすることで対応する．さらに，クラスタ内での代表語を選出することで，そのクラスタに属する不具合の概要がつかめることを期待する．

図 5 にスマートフォンに関する不具合文と不具合文から得られた症状を表す単語のイメージを示す．本研究では図 5 の左にあるような不具合文が 4 文あった場合，右のように不具合の症状をまとめ，不具合の症状を表す単語の獲得を目指す．

3.2.1 不具合文のクラスタリング

クラスタリング手法の一つとして K-means 法がある．K-means 法では，あらかじめクラスタ数 K を設定し，クラスタに属する事例から重心などを計算し，K 個のクラスタに事例を分割する．今回の流れでは，この K は得られた不具合文群中に存在する不具合の症状の数を意味する．しかしながら，前もって不具合の症状が何種類あるのか把握することは不可能であり，適切なクラスタ数を事前に設定することは困難である．

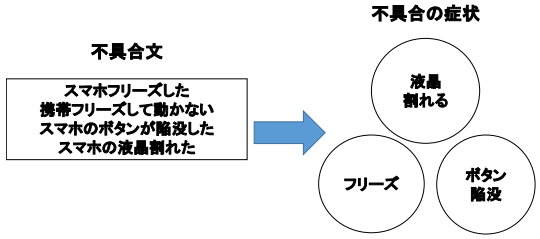

図 5　不具合文 (左) と得られる不具合の症状 (右)

そこで，本研究では Repeated bisection 法 [9] に基づくクラスタリングにより，この問題に対応する．Repeated bisection 法はデータ集合を繰り返し 2 分割し，収束条件として与えたある閾値を利用することで，K-means 法において最適なクラスタ数 K の値を求める手法である．実装にはクラスタリングツール bayon(注3) を用いる．

素性はシンプルな頻度ベースの Bag-of-words とする．形態素解析には MeCab(注4) を用い，追加辞書として mecab-ipadic-NEologd [7] を利用した．また，顔文字などの記号群が頻出し，クラスタリングやそれ以降の処理に悪影響を与えるため，この段階でいくつかのルールにより関連する記号群を削除している．具体的には 「＼ (」 や 「) ／」，w の連続などである．さらに，本手法で可視化する不具合文の特性として，2.6 節で使用したシード表現は基本的に頻出語になりやすいという問題点がある．そこで，事前に 2.6 節で利用した 7 つのシードはクラスタリングの素性空間や以降の処理には含めないようにする．

3.2.2 クラスタごとの不具合の症状の獲得

不具合文が不具合の症状ごとにまとまるようにクラスタリングを行っただけでは，それぞれのクラスタにどのような不具合が属しているのかを直感的に把握することはできない．ここで，各クラスタが正しく不具合の症状ごとにまとまっていると仮定すると，不具合の症状は，そのクラスタで頻出し，他のクラスタでは出現しにくい単語であると考えられる．そこで，本研究では，$tf \cdot idf$ の idf を文書単位ではなくクラスタ単位で考え，各クラスタの代表語を選出する．すなわち，1 つのクラスタを 1 文書とし，クラスタ内での単語 t の出現数を tf 値，特定のクラスタに出現する単語 t の値が高くなるような値を idf 値と定義する．

具体的には，tf 値，idf 値はそれぞれ式 (2)，式 (3) のように求められる．

$$tf = \frac{\text{クラスタでの単語 } t \text{ の出現回数}}{\text{クラスタ内の全単語数}} \tag{2}$$

$$idf = \log \frac{\text{全クラスタ数}}{\text{単語 } t \text{ を含むクラスタ数}} \tag{3}$$

各クラスタ内でこの $tf \cdot idf$ 値が最大となるものを各クラス

(注3)：https://code.google.com/archive/p/bayon/
(注4)：http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html

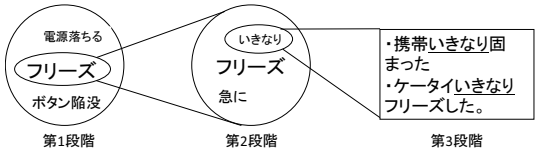

図 6　可視化の全体的な流れ

タの代表語とする．ただし，前節のクラスタリングの精度は100%ではないため，同じような症状が別のクラスタに属している場合もある．また，そもそも不具合を表しやすい単語や不具合文抽出で利用したルールに関係する要素（2.3.1 節や 2.3.3 節を参照）は代表語になりやすく，複数のクラスタが同じ代表語を持つ場合が存在する．そのような場合は，最も大きな値を持っている単語のみを代表語とし，それ以外のクラスタでは $tf \cdot idf$ の上位から他のクラスタと重複がなくなるまで代表語を探索し，選定する．

3.3　獲得された不具合の症状の可視化

本節では獲得された不具合の症状を表す単語を基に，可視化を行う手法について述べる．可視化の方針として，Shneiderman [8] らの提唱する「まず概観し，拡大や除去の後，要求に応じた詳細化を行う」という概念に基づき，3 段階に分けて可視化を行う．概観，拡大と除去，要求に応じた詳細化とは，それぞれ可視化のタスクである．概観ではデータ集合の全体像を把握する．拡大で注目すべき要素の拡大，除去で必要のない要素の除去を行う．そして要求に応じた詳細化では，特定の要素から詳細な情報を得る．これらのタスクをそれぞれの可視化の段階に用いる．

図 6 に可視化の全体的な流れの例を示す．第 1 段階では，不具合を象徴する単語を表示する．次に，第 2 段階では第 1 段階で選択された単語のクラスタ内の単語群を表示する．そして第 3 段階では，第 2 段階で選択された単語を含む不具合文を表示する．

実際には，第 1，第 2 段階ではワードクラウド (注5) と呼ばれる手法を用いて可視化を行う．ワードクラウドとは，解析対象とするテキスト内で使われている単語を視覚化する手法で，文章中の「ワード (単語)」を出現頻度や重要度によって大きさを変えて表示し，「クラウド (雲)」のような形で図示する．この手法は，近年 Twitter を情報源にユーザのつぶやきの特徴や傾向を直感的に把握できることから頻繁に利用されている．本研究のように膨大な量の単語を一目で理解できるようにする場合，ワードクラウドを用いるのが適切であるといえる．

第 1 段階：概観） 第 1 段階は概観の部分に相当し，クラスタごとの不具合を象徴する単語（代表語）を表示する．各単語は $tf \cdot idf$ 値に基づき，表示するときの大きさが決定される．収集された不具合文の全体像を視覚的にわかりやすく図示する．

第 2 段階：拡大と除去） 第 2 段階は拡大と除去に相当する．第 1 段階で表示されている単語を選択すると，その単語のクラスタ内に出現する単語群を表示する．これにより，ある代表語

(注5)：http://timdream.org/wordcloud2.js

図 7　第 1 段階の出力（概観）．

に対して，どのような不具合が存在するかを容易に把握できる効果が期待される．例として，図 6 の第 1 段階では，「フリーズ」といった症状がみてとれるが，それがどのような状況で起こったのかまでを把握することはできない．しかし，第 2 段階からは「急にフリーズ」や「いきなりフリーズ」といった情報が得られ，どのような状況で起こったのか把握することができる．この第 2 段階では，各クラスタの $tf \cdot idf$ 値に基づき，最大上位 200 単語までを表示する．

第 3 段階：要求に応じた詳細化） 第 3 段階はオンデマンド詳細化の部分に相当している．このオンデマンド詳細化では，特定の単語から詳細な情報を得ることが目的である．第 2 段階で表示されている単語をさらに選択すると，その単語を含む不具合文を表示する．選択された単語は，不具合文中でどこにあるのかすぐに把握できるよう，色を変更する．

3.4　可視化による不具合情報の分析と議論

実際に，先行研究によって得られた不具合文（2.6 節でのスマートフォン・携帯に関する 3,249 ツイート）を用いて可視化した．その可視化の結果に基づき，提案手法の有効性を実際の分析を通して議論・検証する．

提案した手法ではまず，クラスタリングを実行する．Bayon に与えるパラメータ（閾値）は 5.0 とした．また，-idf オプションも利用した．実行の結果，34 個のクラスタが得られた．実際にはいくつかの閾値と Repearted bisection 法ではなく，いくつかの K の値による K-means 法でも実行したが，直感的にこのパラメータの設定が最も良いと感じたため，以降はこの結果について議論する．

3 段階に分けて行った可視化の結果をそれぞれ図 7，図 8，図 9 に示す (注6)．第 1 段階となる図 7 では不具合の全体像が見える．たとえば，この中に「充電器」が存在する．これは 2.6 節で述べた不具合情報抽出の失敗例が集まったものである．このように，提案手法は単に不具合情報の分析に役立つだけでなく，抽出手法のエラー分析としても機能する．

図 8 は，第 1 段階において「起動」を選択した際に表示された結果である．この画像を見れば，このクラスタは「勝手に再起動する」という不具合の集合であることがある程度推測可能であり，提案手法に基づく分析手法の有効性の一部が垣間見え

(注6)：第 1 段階と第 2 段階の文字の色および位置はランダムに決まり，実行するごとに異なるので大きな意味はない．

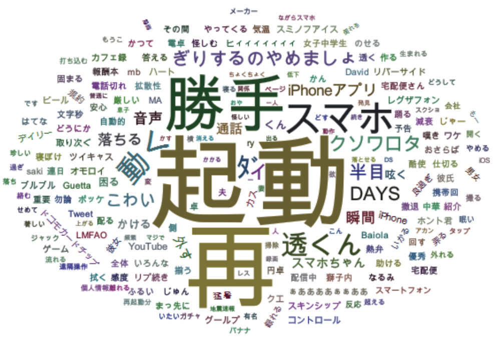

図 8　第 2 段階：第 1 段階において「起動」を選択した状態.

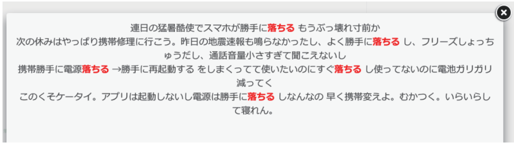

図 9　第 3 段階：第 2 段階において「落ちる」を選択した状態.

る．ここで，さらに単語をクリックすると第 3 段階に移行する．図 9 が「落ちる」が選択された場合の第 3 段階の出力である．この図 9 に表示される実際の事例を見ることで，どのような点で問題があるのかを把握することができる．また，この例における以下のツイートに着目する．

「起動」→「落ちる」の具体例

・携帯勝手に電源落ちる →勝手に再起動する
・寝てる時に携帯が何回かブルブル震えてたので、電話かと思いきや起きたが、何故か一人で勝手に落ちて再起動してたらしい。スマートフォン、ホント君は使えない子。今日かつて優秀な携帯を作ってきた日本の携帯メーカーが撤退して行く中、しみじみ思う。

これらの不具合文からは「電源が落ちた後に再起動した」といったように，どうしてその不具合が起きたのかまで把握することができる．このような結果から，3 段階での可視化の有用性を確認できた．一方で,「再」と「起動」が分割されて処理されているなど，前処理で改善できる問題も散見される．このような問題への対処は課題の一つである．

視覚的に確認できることで得られる興味深い事例についても議論する．図 10 は第 1 段階で「ケータイ」をクリックした場合の結果を拡大したものである．ここで，図の下部に見られるようなひらがなのみで構成された長い文字列（ゆうりょくがいちいちめんどくさい）があることに気づく．この文字列をクリックすると図 11 のような実例がポップアップされる．この実例を見れば分かるように，これは IME の不具合を述べたツイートである．このような事例は可視化の効果を体現したものであり，視覚化することでより気づきやすくなったものだと考えられる．

4. おわりに

本論文では，Twitter を対象とした不具合情報抽出の結果を可視化することで，その内容を分析する手法を提案し，その結果について述べた．ワードクラウドで可視化することで，不具

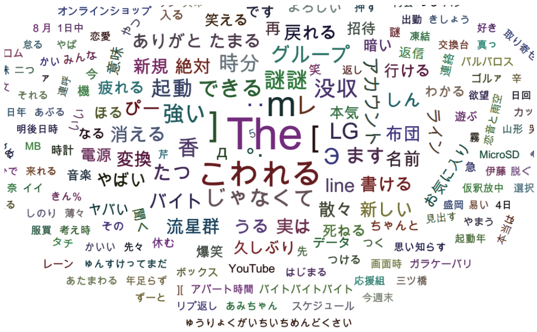

図 10　第 1 段階で「ケータイ」を選択した場合．図下部にひらがなのみの文字列がある.

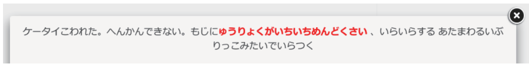

図 11　不具合の実例.

合の全体像だけでなく，抽出手法の頻出エラーなどを容易に把握することが可能になった．しかしながら，ワードクラウドには不要な単語や適切に処理できなかった単語なども存在し，多くの課題が残る．また，クラスタリングの精度も十分ではないと考えられるため，より正確に不具合の症状ごとに情報をまとめる技術について考察が必要である．また，単純な可視化だけではなく，たとえば，あるクラスタを一言で説明するとどうなるのか，という文書要約的な側面からのアプローチも興味深く，今後の課題として挙げられる．

文　献

[1] James R. Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *Proceedings of the Pacific Association for Computational Linguistics*, pp. 172–180, 2007.
[2] Kohei Kurihara and Kazutaka Shimada. Trouble information extraction based on a bootstrap approach from twitter. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC29)*, pp. 471–479, 2015.
[3] 粟納裕貴, 馬強, 吉川正俊. 失敗知識データベースを用いた失敗事象の原因分析. In *DEIM2012, E2-5*, 2012.
[4] 村上拓真, 那須川哲哉. 特徴的な記述を利用した問題発見手法の実現. 電子情報通信学会 言語理解とコミュニケーション研究会, NLC, pp. 31–35, 2011.
[5] 大森信行, 森辰則. 不具合事例文からの製品・部品を示す語の抽出 -語の実体性による分類-. 電子情報通信学会論文誌 D, Vol. 95, No. 3, pp. 697–706, 2012.
[6] 酒井浩之, 梅村洋之, 増山繁. 交通事故事例に含まれる事故原因表現の新聞記事からの抽出. 自然言語処理, Vol. 12, No. 2, pp. 99–123, 2006.
[7] Toshinori Sato. Neologism dictionary based on the language resources on the web for mecab, 2015.
[8] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Longman Publishing Co., Inc., 1997.
[9] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. Technical report, University of Minnesota, Technical Report 00-034, 2000.
[10] 武田浩一, 野美山浩. テキスト情報の可視化を利用した情報検索. 情報処理., Vol. 41, No. 4, pp. 343–350, 2000.