ブートストラップ法を用いたTwitterからの不具合文抽出

栗原光平† 嶋田和孝‡
†九州工業大学大学院 情報工学府
‡九州工業大学大学院 情報工学研究院
{k_kurihara, shimada}@pluto.ai.kyutech.ac.jp

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 [1]．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 [2], [3] や，不具合事例文から製品・部品を示す語を抽出する研究 [4] 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に時差がある，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．

そこで，本論文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 GeneratedMedia) 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．Twitter は，今していることや感じたことを 140 文字以内で投稿する「マイクロブログ」と呼ばれるコミュニケーションサービスであり，多くのユーザにより大量の情報が発信されている．現在，国内のユーザ数は 3000 万人以上，一ヶ月の日本人の総ツイート数は 2012 年 6 月の時点で 1 億件を超えており，一般のメディアには登場しない個人の経験に基づく不具合情報も Twitter 上に存在すると考えられる．

本研究では，特定の製品の不具合について述べた文（以下，不具合文）を Twitter 上から自動で抽出することを目的とする．情報抽出の分野では，一般的に機械学習による手法が用いられることが多いが，機械学習による手法では Twitter 上から訓練データを収集するのに大きな時間的コストがかかるという問題がある [8]．そこで，本研究では半教師あり学習であるブートストラップ法 [5],[6],[7] を用いたルールベースの抽出手法を提案する．ブートストラップ法は，知識抽出などの分野において用いられており，最初に入力として与えた少数のシードをもとにして，大量の知識を自動獲得することができる．また，本研究では不具合文を抽出するための手がかりとして，製品の異常や不具合の症状を示す表現（以下，不具合表現）に着目し，それらをブートストラップ的に自動獲得することで，より多様な不具合文の抽出を目指す．

2 不具合文と不具合表現

ここでは，本研究で扱う「不具合文」と「不具合表現」について，それぞれの定義と 2 つの違いについて詳しく述べる．まず,「不具合文」とは，製品の不具合や異常について述べているツイートのことである．対して「不具合表現」とは，不具合文において製品の異常や不具合の症状そのものを直接的に表している部分を指し，文章ではなく句やフレーズとしてのまとまりで扱う．図 1 に，スマートフォンに関する不具合文と，それに出現する不具合表現の例を示す．

不具合文「なんで !? スマホの電源がつかない … 」 不具合表現：電源がつかない

図 1: 不具合文と不具合表現の例

3　ブートストラップ法による不具合文抽出

ここでは，本研究の提案手法であるブートストラップ法を用いた不具合文抽出について詳しく述べる．まずブートストラップ法による処理の全体の流れについて詳しく述べた後，不具合表現の獲得方法および信頼度のスコアリング処理について説明する．最後に，獲得した不具合表現を用いた不具合文の抽出方法について詳しく述べる．

3.1　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．なお，一定数の反復を繰り返すか，獲得できる不具合表現が無くなった時点で反復を終了する．

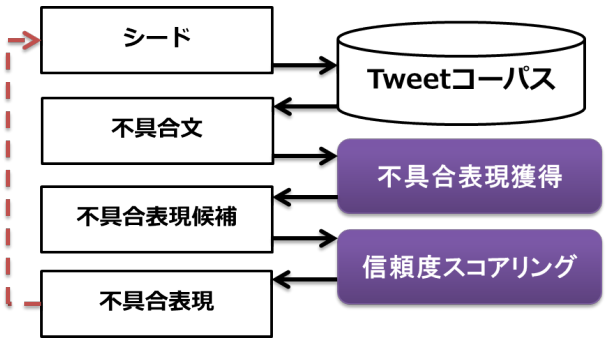

図 2: ブートストラップ法の処理の流れ

3.2　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，不具合文を人手で分析して得られた以下の 3 つの特徴をもとに作成した．

特徴 1 特定の副詞の出現

「勝手に」「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係する副詞は，不具合とも強い関連性を持つと考えられる．また，そのような特長的な記述に着目した問題発見手法も提案されている [9]．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，それらと一緒に出現した動詞や名詞のまとまりを複数の品詞列パターンをもとに獲得する．

特徴 2 動詞の未然形と否定の助動詞の連続

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．この事実にもとづき，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，品詞列パターンにより獲得する．

特徴 3 ネガティブな単語の出現

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，品詞列パターンによって獲得する．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，それらをもとに不具合表現を獲得する．

3.3　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る．よって，ブートストラップ法を用いる際には，毎回の出力の精度をできる限り高く保つことが重要となる．

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．単語の信頼度は，以下の仮定にもとづいて求める．

- 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

ある単語 w の信頼度の具体的な計算式は次のようになる．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．

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

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ら，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，反対に入力として用いる不具合表現の数は低下するため，再現率は低くなる．

3.4 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．

4 実験

提案手法の有効性を確認するため，実際の Twitter 上のデータを用いて不具合文の抽出実験を行った．今回は不具合対象となる製品に携帯電話・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．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．

壊れる, おかしい, 異常, 故障, 破損, フリーズ, バグ

以上の条件でブートストラップによる反復を 5 回行い，その結果に対し人手でアノテーションし適合率によって評価した．反復回数ごとの不具合文の抽出数と適合率の推移を図 3 に示す．

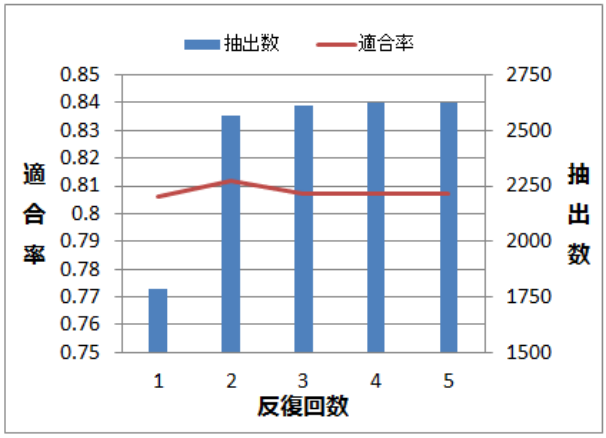

図 3: 反復回数と抽出数・適合率の推移

勝手に電源がつく，急に電源が落ちる，電源が落ちない，画面がつかない，ボタンが押せない

図 4: 獲得した不具合表現の例

2 回目の反復では，1 回目に抽出した不具合文から獲得した不具合表現をシードとして用いているにも関わらず，適合率を維持したまま抽出数が大幅に増加していることがわかる．また，3 回目以降の反復では抽出数の増加は少なかったものの，途中でノイズが増大することもなく，高い適合率を維持したまま反復を繰り返すことに成功している．この結果から，不具合表現のスコアリングが有効に働き，不具合に関連のある表現だけを正しく獲得することに成功していることがわかる．例として，ブートストラップ的に獲得できた不具合表現の一部を図 4 に示す．

電源に関するもの，ボタンに関するものなど，携帯電話やスマートフォンに関連する様々な不具合表現を獲得できていることがわかる．電源に関する不具合表現に着目すると,「電源が落ちない」と「電源が落ちる」という真逆の意味を持つ 2 つの表現が，副詞の有無により異なる不具合表現として成り立っていることがわかる．このような事例は，例えば単純な極性情報（Positive/Negative）などを用いた情報では解決できないため，提案手法がより多様な不具合表現の獲得に有効であることがわかる．

しかし，この結果は適用したデータに依存するものであり，データ数を変えると結果も変動する可能性がある．特に，データ数がより大量になった場合には，出現する表現の数も増えることから，不要な表現を誤って獲得しノイズが爆発してしまう危険性が高くなる．そこで，この結果がデータ数に依存したものでないことを確認するために，データ数以外の条件は同一のま

ま，1 万・5 万・10 万・50 万件の 4 種類のデータを用いて追実験を行った．その結果を表 1 に示す．

表 1: 各データ数ごとの抽出結果

データ数	抽出数
10000	121
50000	623
100000	2623
500000	9088

10 万件の場合から線形的に抽出数が増加したと仮定すると，50 万件のデータを適用した場合に予想される抽出数は 5 倍にあたる約 13000 件である．それに対し，実際の抽出数は約 9000 件となっており，やや下回る結果となったが，ノイズの爆発は防止できていることがわかる．このことから，より大量のデータを適用した場合でも，ノイズが爆発することなく，安定してブートストラップ的な抽出が行えていることが推測される．実際に抽出できた不具合文としては以下のようなものがあった．

- 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

これらの例ではどちらの文にも「壊れた」などの直接的に不具合を示す語は出現していないが,「勝手に電源が切れる」や「電源がつかない」といった部分を不具合表現として認識することで抽出に成功している．

また，データ数が 1 万件の場合の抽出結果を分析したところ，獲得できた不具合表現の数が非常に少なく，2 回目以降の反復でも新たな不具合表現を獲得することができていなかった．提案手法では，不具合文からどれだけ多くの不具合表現を獲得できるかが重要であり，そのためには 1 回目の不具合文抽出でより多くの文が抽出できることが望ましい．よって，提案手法が十分に効果を発揮するためには，ある程度大量のデータセットを用意する必要があるといえる．

5 まとめ

本論文では，ブートストラップ法を用いて Twitter から不具合文を自動抽出する手法を提案した．不具合表現をブートストラップ的に自動で獲得する際には，単語ごとに不具合らしさについての信頼度をスコアリングすることで，不具合表現の信頼度を求めた．実際の Twitter 上のデータを用いた実験では，高い適合率を維持したまま，抽出数を増加させることに成功し，多様な不具合表現が獲得できていることも確認できた．また，データ数を変更しての追実験では，少数のデータには適していないという特徴が明らかになったものの，より大規模なデータを適用した場合でもノイズが増大することなく，安定して不具合文を抽出できることが確認できた．

大量のデータから安定して不具合文を抽出することが可能になったことから，今後は得られた情報のより効果的な活用を支援する方法について検討していきたい．具体的には，抽出した不具合文から特徴や傾向を自動で検出したり，それらを効果的に視覚化して提示するような，人手による分析を支援するシステムの作成などが考えられる．

参考文献

[1] 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 ”, DEIM2012，E2-5，(2012).

[2] 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).

[3] 野畑周, 佐田いち子, 井佐原均, “ 新聞記事中の事故・事件名の自動抽出 ”, 情報処理学会, 研究報告 2005-NL-167, pp.125-130 (2005).

[4] 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.J95-D No.3, pp.697-706 (2012).

[5] P. Patrick, and M. Pennacchiotti, “ Espresso: Leveraging generic patterns for automatically harvesting semantic relations ”, ACL, pp.113-120 (2006).

[6] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, “ Sarcasm as Contrast between a Positive Sentiment and Negative Situation ”, EMNLP, pp.707-714 (2013).

[7] 加藤誠, 大島裕明, 小山聡, 田中克己, “ 共起に基づく Web からの類似関係のブートストラップ抽出 ”, 日本データベース学会論文誌, pp.11-16 (2009).

[8] 栗原光平, 嶋田和孝, “ ルールと機械学習を用いた Twitter からの不具合情報の抽出 ”, 電子情報通信学会技術研究報告, 信学技報 114.81, pp.1-6 (2014).

[9] 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 ”, NLC, 言語理解とコミュニケーション 111.119, pp.31-35 (2011).