Vol.2013-CSEC-62 No.19
Vol.2013-SPT-6 No.19
2013/7/18

非構造テキストデータの機微情報秘匿化支援技術

及川孝徳[†1]　森川郁也[†1]　伊藤孝一[†1]　津田宏[†1]

近年，ネットワーク，クラウドコンピューティングの発達とともに，利用者が自分の持つデータを外部業者に預け，分析結果を得る形態のサービスが増加している．しかし，預けるデータに利用者の機微情報が含まれる場合，セキュリティ上の懸念が生じる．そこで本稿では，非構造テキストデータを対象に，ルールを使った機微情報秘匿化方式を想定し，その課題を解決するルール作成支援システムを提案する．想定する秘匿化方式では，予め対象テキストデータの秘匿・開示箇所を，行の形式毎に正規表現でルール化し，秘匿化時には，ルールに含まれる形式の行はルールにそって秘匿・開示を行い，未知形式の行はすべて秘匿することで秘匿漏れを防ぐ．提案するルール作成支援システムは，既存のルールを再利用して，ルールを自動作成することで，ルール作成コストを削減する．また，ルール同士を比較し，同じ文字列にマッチする可能性があるルールペアを検出することで，ルール干渉による秘匿漏れの発生を事前に防ぐ．提案手法によって，非構造テキストデータ内機微情報を過不足無く秘匿化する方式の，効率よい実現が期待される．

Supportive Techniques for Masking Sensitive Information in Unstructured Text

Takanori OIKAWA[†1]　Ikuya MORIKAWA[†1]
Kouichi ITO[†1]　Hiroshi TSUDA[†1]

Recently, services which receive and analyze user data are increasing as advancement of networks and cloud computing. However, sending data containing sensitive information to external services may cause unintended disclosure of information. Hence, we propose a system which masks sensitive information in unstructured text based on some masking rules. In the system, regular expressions are used as rules to determine which word in a line is safe to remain unmasked. The system fails safely because it masks a whole line if any masking rule can't be applicable for the line. We also propose methods to assist configuring rules for such a system, by making use of existing rules to reduce efforts for making new rules, and by finding inconsistent pairs of rules which may cause masking failure.

1. はじめに

近年，ネットワーク，クラウドコンピューティングの発達とともに，リモート保守やエネルギーマネジメントサービス等，自分の持つデータを外部に預け，分析結果を得る形態のサービスが増加している．しかし，サービスに預けるデータには利用者の機微情報が含まれる場合がある．例えば，プログラムエラーの原因究明等で使用されるWindowsイベントログには，電子メールアドレスやIPアドレス等の，利用者の個人情報が含まれる．そのため，利用者によっては，情報漏洩をおそれ，外部サービスにデータを預けることができない場合がある．また，サービス提供側は，機微情報を含むデータを扱う場合，社内におけるアクセス制御や，分析完了後のデータ削除証明等，セキュリティ対策が必要になるため，管理コストが増加する．そこで本稿では，サービスに必要ない機微情報を秘匿化する手法を検討する．機微情報を秘匿化することで，データの安全な分析が可能になる．

本稿では，秘匿化対象を，ある程度規則性を持った非構造テキストデータとする．対象とするテキストデータは，1行～複数行単位に決まった形式で情報が記述される．例えばユーザー名の場合は「UserName : xxxxx」，エラー発生箇所の場合は「'xx/xx/xx'でエラーが発生しました」といった形式である．対象とするデータの1つとして，ログデータがあるが，ログ管理の課題[1]として，1つの生成元が複数のログを生成しうること，一貫性のない内容・形式であること，が挙げられ，ログに含まれる形式は膨大になる．よって，事前にログに出現しうるすべての形式・出力される情報を網羅することはできず，対象データに含まれる機微情報を事前に把握することもできない．

そこで本稿では，ルールを使った秘匿化方式を想定する．ルールを使った秘匿化方式は，まず，予め人手で，出来る限りの形式と，形式ごとの秘匿・開示箇所を正規表現でルール化する．そして秘匿化処理時は，対象のデータがルールに含まれる形式であれば，ルールにそって秘匿・開示を行い，ルールに含まれない未知の形式であれば，全体を秘匿する．本方式では，ルールに含まれる形式のデータは機微情報だけが秘匿され，未知形式のデータは全体が秘匿される．よって，未知の形式で機微情報が出現した場合でも，機微情報の秘匿漏れは発生しない．

想定する秘匿化方式は，ルールが正しい前提において，秘匿漏れは起こらず，安全である．しかし，ルールに含まれない形式は全体が秘匿されてしまい，分析支障をきたす．よって，ルールはなるべく多くの形式を含んでいる必要が

†1 株式会社富士通研究所
Fujitsu Laboratories Ltd.

Vol.2013-CSEC-62 No.19
Vol.2013-SPT-6 No.19
2013/7/18

あり，ルール作成コストが大きくなる．ある程度規則性のあるテキストデータにおける，テンプレート作成を支援する研究として，共通部分文字列からテンプレートを抽出する研究[2][3][4]がある．提案手法では，秘匿化対象のテキストデータにおいて，部分的に類似している形式が多いという性質を利用し，今までに作成した既存ルールを再利用することで，ルール作成コストの削減を試みる．ルールの再利用は，固定文字列テンプレートだけでなく，正規表現で記述された可変部分も再利用できるため，有効な支援が期待できる．

正規表現で記述されたルールは，ルール作成者の意図していない干渉が起こる可能性がある．例えば，「name : [a-z]+」と「[a-z]+ : true」はそれぞれ別の形式を対象として作成されたルールにもかかわらず，共に「name : true」という行にマッチする．このとき，両ルールの秘匿箇所が異なれば，ルールの適用順によって，機微情報の秘匿漏れが起こる．単純な解決策として，対象行にマッチする全ルールの秘匿箇所をすべて秘匿する方法があるが，常に全ルールと対象行のマッチを判定しなければならず，効率が悪い．提案手法では，ルール同士を比較し，ルールが内包する文字列の中に一致する文字列が含まれるかどうかを探索，干渉する可能性があるルールペアを抽出し，秘匿化処理以前の，ルール作成時点で，課題を解決する．

2 つの課題解決方針を実現するには，正規表現と文字列，正規表現同士の類似度を算出する技術が必要になる．提案手法では，類似度に編集距離を用い，正規表現に対応する編集距離算出を行う．編集距離とは，2 つの文字列がどの程度異なっているかを示す数値であり，文字の削除・挿入・置換の編集操作によって，片方の文字列をもう片方の文字列に変形するのに必要な最小回数として与えられる．具体的には，比較する 2 文字列を各軸とする格子状の編集距離算出グラフを作成して算出する．編集距離算出グラフの各エッジには編集操作の重みがついており，始点から終点までの最短経路長が編集距離となる．また，求めた最短経路から，一方の文字列をもう一方の文字列に変形するのに必要な編集操作を得ることができる．正規表現の編集距離に関する研究として，オートマトンを拡張し，設定した編集距離以下での，正規表現と文字列のマッチを判定する研究[5]があるが，正規表現同士の編集距離算出はできない．

提案手法は，点・経路を追加することで，編集距離算出グラフ上に，正規表現が内包する文字列をすべて表現し，編集距離算出を行う．提案手法は，編集距離算出グラフの拡張であるため，入力する 2 文字列に区別はなく，正規表現同士の編集距離算出も可能である．想定する機微情報秘匿化方式に用いるルールは，正規表現で記述されるため，提案手法によって，ルールと文字列，ルール同士の編集距離算出が可能になる．

1 つ目の課題，ルール作成コストは，既存ルールの再利用によって解決する．まず，ルールと文字列の編集距離算出を用いて，ルール作成対象行に類似する順に既存ルールを抽出する．次に，抽出された既存ルールから，対象行と一致する部分を引用してルールを自動作成し，ルール作成者に提示する．引用する一致部分に条件を設定することによって，自動作成ルールに一定の安全基準を持たせることができる．ルール作成者は，提示されたルールをそのまま利用するか，簡単な修正を加えてルールとすることができるため，ルール作成コストの削減が期待される．

2 つ目の課題，ルール適用範囲の干渉は，ルール同士の編集距離を用いて判定する．編集距離が 0 であるルールのペアは，共通でマッチする文字列が存在する．よって，編集距離が 0 であるルールのペアを抽出し，抽出されたルールに対し人手で，ルールを修正，もしくは，ルールの適用に優先順位をつけることで，干渉を解消する．

本稿では，ルールを使った機微情報秘匿化方式を想定し，想定する方式の課題を解決する，ルール作成支援システムを提案する．提案手法によって，秘匿漏れの無い機微情報秘匿化方式の，効率よい実現が期待される．

本稿の構成は以下のとおりである．第 2 章では関連研究について述べ，第 3 章で提案手法が想定する秘匿化方式について説明する．第 4 章で提案手法について説明し，第 5 章では評価実験を行い，第 6 章でまとめを述べる．

2. 関連研究

2.1 非構造テキストデータの機密情報秘匿化

非構造テキストデータの秘匿化に関連するサービスとして Splunk[6]が挙げられる．Splunk は，ログデータやシステムの設定情報等を収集・インデックス化し，データをまたがった分析を可能にするデータ分析支援システムである．Splunk には anonymizer と呼ばれる，データの一部を秘匿化する機能がついている．anonymizer は，ユーザーが正規表現を記述し，秘匿箇所を指定することで，データ内の正規表現にマッチする部分を別の文字列に置き換え，秘匿化を行う．anonymizer は，正規表現によって指定した箇所のみを秘匿する方式であるため，事前に想定していない機微情報に関しては，秘匿漏れの危険性がある．また，正規表現の作成は完全に人手である．

2.2 テンプレート作成支援に関する研究

ある程度規則性をもったテキストデータにおけるテンプレート作成を支援する研究として池田ら[2][3][4]の研究が挙げられる．池田らは，テンプレートに代入される文字列の出現頻度はベキ分布に従うと仮定し，部分文字列増幅法という手法を用いたアルゴリズムを提案，複数の異なる形式が含まれるテキストデータからも，テンプレートを発見できることを示した．

2.3 正規表現の編集距離に関連する研究

正規表現の編集距離に関する既存研究として，文字列の

あいまい検索を目的とした[5][7][8][9]が挙げられる．Navarro ら[5]の NR-grep はオートマトンを拡張し，予め設定したコスト以下の編集操作で正規表現と文字列がマッチするかを判定する．NR-grep は与えたコスト以下でのマッチ判定を目的としているため，最小コスト，つまり，編集距離を算出する場合にはアルゴリズムを複数回実行する必要がある．また，オートマトンの入力は文字列に限定されるため，NR-grep では正規表現同士の比較はできない．そのため，ルール同士の編集距離算出が必要になる提案手法には適さない．

3. 想定する機微情報秘匿化方式

3.1 秘匿化対象データについて

本稿では，データベースや XML といった構造を持たない非構造テキストデータ，かつ，アプリケーションログやシステムログ等の，ある程度規則性を持ったデータを対象とする．例えば，ログデータの場合，1～複数行単位に決まった形式で情報が記述される．ただし，ログ管理の課題[1]として，1 つの生成元が複数のログを生成しうること，一貫性のない内容・形式であることが挙げられ，ログの形式は膨大になる．また，生成元のシステム更新によって形式が変化することもある．そのため，ログに出現するすべての形式を事前に網羅することはできない．よって，ルールを使った秘匿化方式においては，ルールに含まれない形式で機微情報が出力される危険性を考慮しなければならない．

3.2 想定する秘匿化方式概要

ルールを使った機微情報秘匿化方式は，予め人手で，対象のデータに含まれる形式の秘匿・開示箇所を，出来る限りルール化する．そして秘匿化処理時には，対象のデータがルールに含まれる形式であれば，ルールにそって秘匿・開示を行い，ルールに含まれない未知の形式であれば，全体を秘匿する（図 1）．

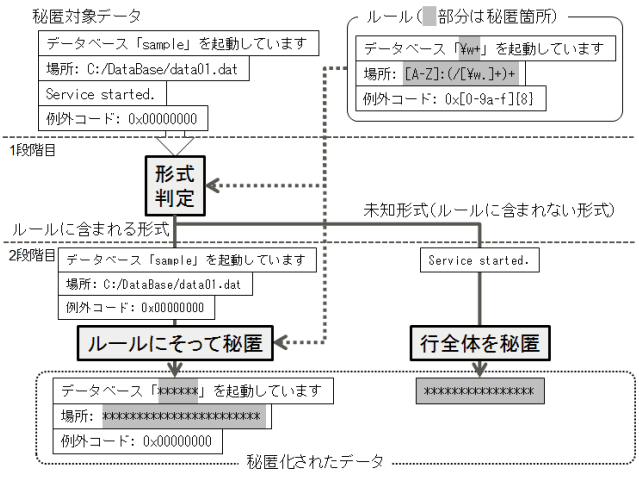

図 1　ルールを使った秘匿化処理の流れ

本方式では，ルールに含まれる形式のデータは機微情報だけが秘匿され，ルールに含まれない，未知形式のデータは全体が秘匿される．よって，未知の形式で機微情報が出現した場合にも，機微情報の秘匿漏れは発生しない．

3.3 使用するルール

本方式のルールは行単位の正規表現で記述する．正規表現とは，文字列群を一つの文字列で表現した文字列であり，「(AB|XYZ)」（AB または XYZ）のような「選言」と，「(AB)+」（AB の 1 回以上の繰り返し）のような「量化」の 2 つの特徴を持つ．ルールに正規表現を使用するのは，正規表現が人間に対して可読性が高く，機微情報が秘匿される条件への納得性が高いためである

3.4 秘匿化方法

ルール作成の際，秘匿箇所に合わせて以下の 3 つから秘匿方法を選択する．

- 墨塗り
  秘匿箇所を別の文字列に置き換える．秘匿箇所に関係なくすべて同じ文字列に置換する
- トークン化
  秘匿箇所を別の文字列に置き換える．同じ秘匿箇所が同じ文字列になるように置換する．
- 暗号化
  秘匿箇所を暗号化する．

秘匿箇所に関する情報を全く残したくない場合は，墨塗りを選択する．秘匿箇所自体は隠しつつ，値の一致・不一致や，秘匿箇所の時間変化を知りたい場合にはトークン化を用いる．複数の担当者・組織でデータを扱い，担当者・組織毎に開示する情報を変えたい場合は，暗号化によって，秘匿箇所の閲覧者を制限することができる．

4. 提案手法

提案するルール作成支援システムは大きく 2 つの要素からなる．1 つ目は，既存ルールの再利用を行う，ルール自動作成である．2 つ目は，作成したルール同士を比較し，適用範囲の干渉を判定するルール干渉検知である．

本章では，4.1 節でルール作成支援システムの構成技術を実現する，正規表現に対応する編集距離算出技術について説明し，4.2 節，4.3 節で，ルール作成支援システムを構成する各技術の詳細を説明する．

4.1 正規表現に対応する編集距離算出

文字列同士における編集距離とは，片方の文字列をもう片方の文字列に変形するのに必要な編集操作の最小回数を指す．正規表現は，文字列群を一つの文字列で表現した文字列である．そこで，正規表現が内包する文字列の中で，比較対象の文字列との編集距離が最も小さい文字列との編集距離を，正規表現と文字列との編集距離とする．正規表現同士の比較も同様に，各正規表現が内包する文字列の中で最も編集距離の近い文字列ペアを抽出し，その文字列間の編集距離を，正規表現同士の編集距離とする．

提案手法は，編集距離算出グラフに，正規表現の「選

言」・「量化」表現の「(」・「|」・「)」の記号点と，各軸に平行な距離 0 のエッジを追加することで，グラフ上に「選言」・「量化」を表現し，正規表現が内包する文字列群すべてを表す（図 2）．作成した編集距離算出グラフに対し，始点から終点までの最短経路探索を行い，編集距離を求める．提案手法は，編集距離算出グラフの拡張であるため，入力する 2 文字列に区別はなく，どちらにも正規表現をとることができ，正規表現同士の編集距離算出が可能である．

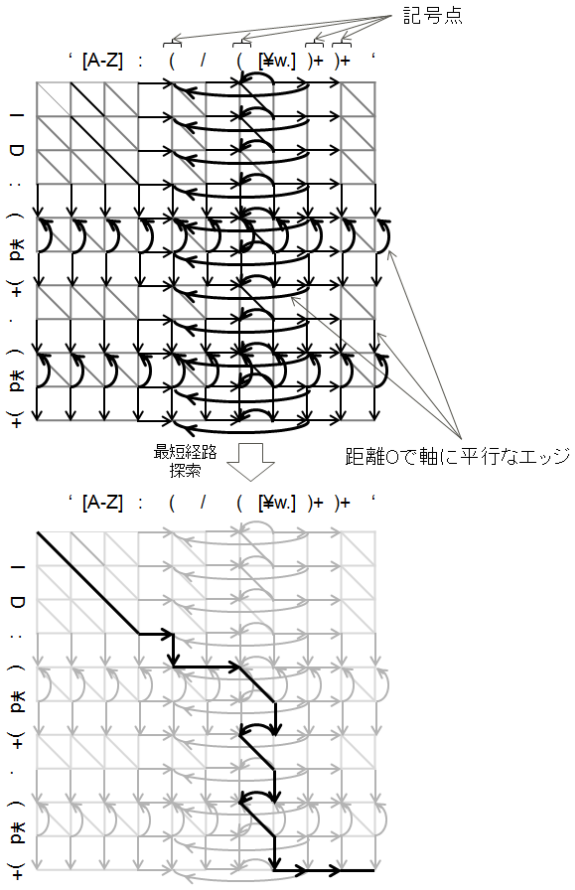

図 2　正規表現「’[A-Z]:(/[¥w.]+)+’」と
正規表現「ID:¥d+¥.¥d+」の編集距離算

提案するルール作成支援システムにおいて，ルールは正規表現で記述される．よって，本技術により，ルールと文字列の編集距離算出，ルール同士の編集距離算出が可能になる．また，編集距離算出時，片方をもう一方にマッチさせるのに必要な編集操作を求めることができる．

4.2　ルール自動作成

ルール自動作成ツールは，既存ルールを再利用し，ルール作成対象行にマッチするルールを自動作成する．

まず，4.1 節のルールと文字列の編集距離算出を利用して，ルール作成対象行に類似する順に既存ルールを抽出する．次に，既存ルール更新・新規ルール作成それぞれの場合のルールを自動作成し，ルール作成者に提示する．

既存ルールの更新の場合は，1 つの既存ルールを引用してルールを自動作成する．対象行に最も類似する既存ルールを，編集距離算出時に求められる編集操作を利用し，既存ルールの適用範囲を残したまま，対象行にもマッチするように自動修正して，ルール作成者に提示する（図 3）．既存ルールの更新は，ルールの数を変えないまま，適用範囲を適切に広くすることが可能である．出力元のシステム更新によって，形式が変化する際等，ある 1 つの既存のルールからの変化が小さい場合，ルールを複雑にすることなく全体のルール数を抑えることができるため，既存ルール更新は新規ルール作成より有効である．

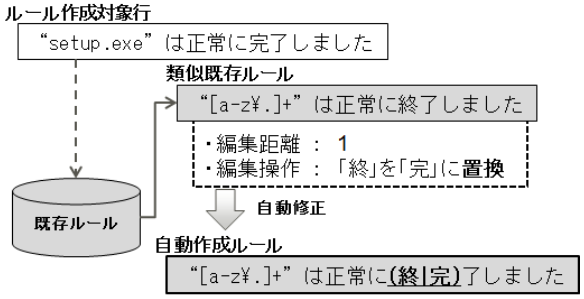

図 3　既存ルール更新のルール自動作成の流れ

新規ルール作成の場合は，対象行との類似度が高い順に，既存ルールから，対象行との一致部分を引用し，ルールを作成，ルール作成者に提示する（図 4）．引用する一致部分に，長さ・単語数・文字種等の条件を設けることで，自動作成ルールに一定の安全基準を持たせることができる．

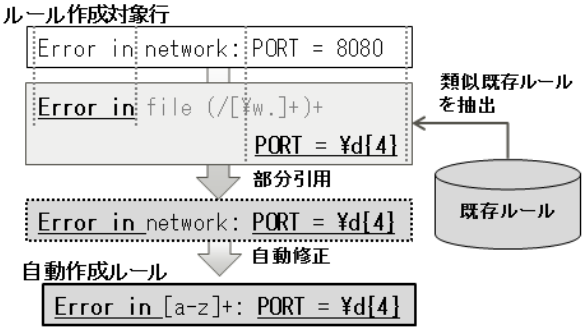

図 4　新規ルール作成のルール自動作成の流れ

ルール作成者は，提示された更新ルール・新規ルールをそのまま利用するか，簡単な修正を加えてルールとすることができるため，ルール作成コストの削減が期待される．

4.3　ルール干渉検知

ルール干渉検知は，作成されたルールの適用範囲の干渉を判定する．適用範囲の干渉とは，ある文字列に対して，2 つ以上のルールがマッチすることである．編集距離が 0 のルールのペアは，共通でマッチする文字列が存在する．よって，4.1 節のルール同士の編集距離算出を利用し，作成されたルールの中から編集距離が 0 であるルールのペアを抽出し，ルール作成者に提示する．ルール作成者は，提示されたルールのペアに対し，ルール自体を修正，もしくは，ルールの適用順に優先度をつけることで干渉を解消する．

5. 評価実験

本章では，ルール作成支援システムのうち，ルール自動作成について評価を行う．

5.1 使用データ

実験には秘匿化対象データとして，Windows イベントログのアプリケーションログを使用した（図 5）．アプリケーションログは，イベント単位で，Date・EventID・User・Computer・Description 等のフィールドから構成される．本実験ではフィールドのうち非構造テキストデータである，Description 部分のみを抽出し，実験データとして使用した．

```
Event[0001]:
  Log Name: Application
  Source: Microsoft-Windows-Security-SPP
  Date: 2013-01-01T00:00:00.000
  Event ID: 1033
  Task: N/A
  Level: 情報
  User: N/A
  Computer: Oikawa-PC
  Description:
これらのポリシーは、上書き専用属性でのみ定義されてるため、除外されています。
ポリシー名=(XXXXXXXXXX)
アプリケーション ID=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
Sku ID=XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

図 5　アプリケーションログ例（一部加工）

実験データについて表 1 に示す．形式数は，実験データを網羅するのに必要なルールの数にあたる．アプリケーションログをイベント ID で分類し，各イベント ID の Description フィールドの値を行単位でルール化したときのルール数として求めた．ルールの作成は人手で行った．

表 1　実験データ

EventID	形式数	サイズ	行数	収集期間
127 種	275 種	9822KB	349416 行	2012.11～2013.5

作成したルールを用いて，実験データに含まれる形式の頻度分布を調査した（図 6）．形式毎の出現数はほぼ指数的に減少している．実験データは全形式の約 9.5%の形式によって，全行の 90.6%が構成されている．

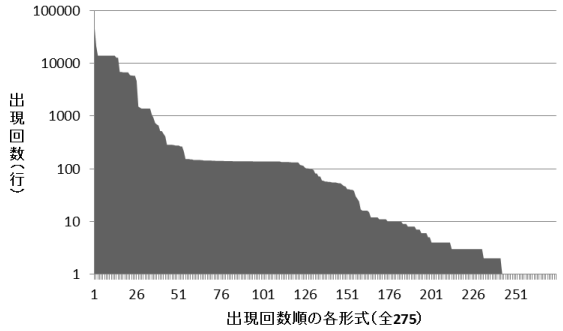

図 6　出現回数順の各形式出現回数

5.2 実験方法

実験は以下の手順で行う．

① 実験データに含まれる形式のうち，何%かの形式を選択し，既存ルールとする．実際のルール作成作業を想定し，出現数の多い形式順に選択した．

② 実験データ中の，既存ルールにマッチしない行に対して，ルールの自動作成を行う．自動作成ルールは，すべて新規ルール作成とした．

③ ルール作成対象行のうち，既存ルールの引用でカバーできた割合を自動作成ルールの引用率として算出する．

④ 自動作成ルールをそのまま使い，既存ルールにマッチしない行に対し，秘匿化を行う．

⑤ ルール毎に，秘匿化結果と，人手で作成した正解セットを比較し，開示精度・再現率を算出する．精度・再現率は文字単位で算出した．

上記実験を，既存ルールに使用する形式の割合を変えながら行う．実験データを網羅するルール，正解セットは人手で作成した．本実験では，ファイルパス，IP アドレス，URL，PC 内のアプリケーション名，実行ファイル名，一部の ID が機微情報であると仮定し，秘匿箇所とした．

また，ルール自動作成における，一致部分の引用条件は以下とした．

- 英字・数字は，2 単語以上連続した場合に引用
- 全角文字，記号は 3 文字以上連続で引用
- 文字クラス（[a-z]等）を含む箇所は，固定文字列と連結して引用される場合のみ引用する
- 条件が同じ引用部分に秘匿・開示の両方がある場合，秘匿を優先

既存ルールから引用できなかった箇所は，記号は開示箇所とし，文字部分は「[a-z]」「[0-9]」「.」等の正規表現の文字クラスに置換し，秘匿箇所とした．非引用部分を文字クラスに置換するのは，実際の運用を想定し，作成されたルール内に機微情報が含まれないようにするためである．

5.3 実験結果

自動作成ルールの作成数と，引用率を表 2 に示す．既存ルール割合が増加するにつれ，自動作成ルール数が作成対象の正解ルール数に近づくことがわかる．引用部分，特に値に幅を持った引用部分が増加するため，適切な適用範囲を持つルールを自動生成できるようになっていったと考えられる．既存ルール割合 90%において，自動作成ルール数が作成対象の正解ルール数より少なくなっているのは，自動作成ルールにおける非引用部分を文字クラスに置き換えた結果，他形式の行にもマッチし，適用されたためである．

表 2　自動作成ルール数と平均引用率

既存ルール割合	60%	70%	80%	90%
作成対象の正解ルール数	110	82	55	27
自動作成ルール数 (正解数との差)	127 (+17)	98 (+16)	58 (+3)	26 (-1)
平均引用率	0.597	0.619	0.669	0.722

また，各既存ルール割合における，引用率と全体の割合

の変化を図 7 に示す. ルール全体における割合が最も大きいのはどの既存ルール割合においても，引用率 70%であるが，既存ルール割合が増えるにつれ，60%以下の分布が減少していることがわかる．

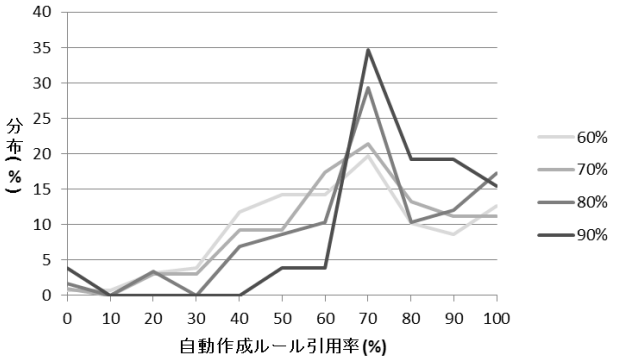

図 7　自動作成ルール全体における引用率の分布

自動作成ルールの開示の平均精度と平均再現率を表 3 に示す. 既存ルール割合が 80%以上の自動作成ルールに関しては，ほぼ秘匿漏れを起こさずに，開示すべき箇所のうち 6 割を開示できることがわかる．

表 3　自動作成ルールの平均精度・平均再現率

既存ルール割合	60%	70%	80%	90%
平均精度	0.928	0.938	0.98	0.993
平均再現率	0.523	0.577	0.608	0.66

開示誤判定には 2 つのケースがあった．1 つ目は，自動作成ルールにおいて，引用処理後，非引用部分の記号を開示箇所としたことによる，誤判定である．記号周辺の他の文字は秘匿されており，人手で作成したルールで秘匿と判定された記号もそれ単体では開示されても構わないとみなせるため，秘匿漏れにはあたらないと考えられる．2 つ目は，他形式で開示されていた箇所の引用によって秘匿箇所が開示されたケースである．例えば「Windows Search」は，「Windows Search サービスが正常に停止しました」という固有の開示形式が存在する．しかし,「アプリケーションまたはサービス’Windows Search’をシャットダウンできませんでした」のように機微情報（アプリケーション名）が含まれうる形式（「’」に挟まれた箇所）にも出現するため，開示誤判定が起こった．ただし，既存ルール割合が増えるにつれ，「’」に挟まれた文字列を秘匿する形式が含まれるようになり，開示誤判定は無くなった．そのため，既存ルール割合が増えるとともに，平均精度は向上している．

また, 表 3 より, 開示すべき箇所をすべて開示するには, 開示再現率を 4～5 割引き上げる必要があり, それが人手の修正作業にあたると考えられる．開示再現率を 0 から 1 にすることがルール作成作業だと仮定すると，ルール自動作成によって，作業全体の 5～6 割を削減できるといえる．

自動作成ルール例とその秘匿化結果を図 8 に示す．自動作成ルール例の「HRESULT:0x[0-9a-f]{8}」のように，値に幅を持った部分ルールの引用は，提案手法でしか実現できないため，提案手法が有効に機能していることがわかる．

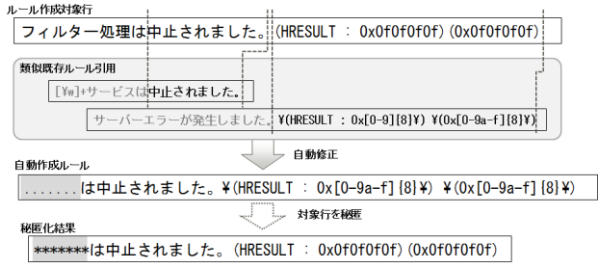

図 8　自動作成ルールと秘匿結果例

本実験より，対象の非構造テキストデータのルール作成において，既存ルールの再利用が可能であることがわかった．また，実験求められた開示再現率から，作業コスト削減量を推測することができた．実際の作業コスト削減量については，今後評価を行なっていきたい．

6. まとめ

本稿では，非構造テキストデータを対象に，ルールを使った機微情報秘匿化方式を想定し，想定する方式の課題を解決するルール作成支援システムを提案した. 実験の結果, 既存ルールの再利用によって，ルール作成コストの削減が実現できることがわかった．

提案するルール作成支援システムによって，過不足無く安全な機微情報秘匿化方式の実現が期待される．そして，機微情報秘匿化によって，データの安全な利活用が可能になる．

今後の展望として，ルール自動作成ツールの性能向上，複数行で構成されたルールへの対応に向けた改良が挙げられる．また，実際の作業コスト削減量について評価を行なっていきたい．

参考文献

1) NIST :コンピュータセキュリティログ管理ガイド (2003).
2) 池田大輔, 山田泰寛, 廣川佐千男: 文字列の頻度分布による共通パタン発見, 第 72 回情報処理学会情報学基礎研究会, PP.25-32 (2003).
3) 池田大輔, 山田泰寛, 廣川佐千男: 部分文字列増幅法による共通パタン発見アルゴリズム, 第 47 会情報処理学会数理モデル化と問題解決研究会, pp.45-48 (2003).
4) 池田大輔, 山田泰寛: 総出現数による文字列の頻出パタンマイニングと共通パタン発見への応用
5) G.Navarro: NR-grep A Fast and Flexible Pattern Matching Tool,Software Practice and Experience, Vol.31, No.13, pp.1265-1312 (2001).
6) Splunk http://ja.splunk.com/
7) TRE http://laurikari.net/tre/
8) FREJ http://frej.sourceforge.net/
9) Sun Wu and Udi Manber: Agrep – a fast approximate pattern matching tool, In Proceedings of USENIX Technical Conference, pp.153-162 (1992).