k-匿名化のための属性値に応じた属性値一般化手法の提案

奥田　容子† 　沼尾　雅之‡

†電気通信大学情報理工学部

‡電気通信大学大学院 情報理工学専攻

E-mail: †o1111033@edu.cc.uec.ac.jp ‡numao@cs.uec.ac.jp

あらまし k-匿名化技術は，企業が保有する大量の顧客データを，そのプライバシーを守りながら，データベースマーケティングなどのサービスに活用できるものとして注目されている．k-匿名化は，同じグループのレコードを k 個以上にするための k-分割の作成と，同じグループになったレコードの属性値を一般化する 2 フェーズに分けられる．本稿では後者に着目して，属性のもつその属性特有の型を定義し，その型に適した形で一般化を行う方法を提案する．そして，典型的なメール，電話番号，住所，年齢などの属性が適切に一般化できることを示す．

キーワード　セキュリティ，プライバシー，k-匿名化

1. はじめに

近年，多くの企業は大量のデータを所有し，サービスを有効に利用できるように努めている．このデータには個人情報はもちろん，履歴や記録など自覚しないうちに情報として活用されているものもある．これらのデータは非常に利用価値が高いが，個人のプライバシーを侵害する可能性や漏洩したときの影響を考え，その扱いは慎重にしなければならない．また，この大量のデータを有益に活用するためには，利用したい情報を考慮しつつデータを加工する必要がある．このような，利用価値と個人のプライバシーの保護を両立することを目指す研究のことをプライバシー保護データマイニング PPDM(privacy-preserving data mining) [1]と呼ぶ．このPPDMは大きく分けて，匿名化アプローチ，ランダム化アプローチ，MPCアプローチ，暗号化アプローチの４つがあげられる．この中で注目したのが匿名化アプローチである．この匿名化アプローチの中でも有名なk-匿名化には，同じグループのレコードをk個以上にするためのステップと同じグループになったレコードの属性値を一般化するステップでの2つのフェーズにより構成される．

本研究の目的は，２つのフェーズのうちたとえ１フェーズ目の方法が異なったとしても，2 フェーズ目のアルゴリズムを同じインターフェースで使えるような一般化 API を定義することである．

2. 関連研究

2.1 文脈自由文法

文脈自由文法(CFG:Context Free Grammer)[2][3]とは文脈自由言語を形成する文法であり，言語に属する文構造を規定する形式文法の 1 つである．この CFG には 4 つの定義が存在し，これらの定義によって句と呼ばれる部分構造を規定する．CFG の文法 G=(T, N, S, P)は以下のように定義される．

- T : 終端記号(terminal symbol)，つまり実際に G によって記述される言語の文を構成する基本的かつ原始的な記号からなる有限集合．
- N : 非終端記号(nonterminal symbol)は，文を構成する句の特定の集合を示す有限集合である．$N \cap T = \emptyset$.
- S : 開始記号(start symbol)は非終端記号の 1 つであり，文法が生成する言語に属する文(句)全体の集合を表す．
- P : 生成規則(production rule)は各非終端記号を表す句が終端記号や他の句からどのように組み立てられるかを規定する規則である．

CFG が生成する言語 L(G)は開始記号から導出を繰り返して得られる文の集合であり，文脈自由言語である．導出において各非終端記号からどのような記号列が導出されたのかを木の形で表現することができる．例として「0,1 からなる長さ 1 以上の任意の系列からなる言語」を生成する文法 G1(図１)

より，文「101」の生成の仕方(図 2，図 3)を示す．

G1 = (N, T, S, P)
N = {S}
T = {0, 1}
P = {S→0, S→1, S→S0, S→S1}

図 1　文法 G1

S⇒S1⇒S01⇒101

図 2　文法 G1 からの文「101」の導出

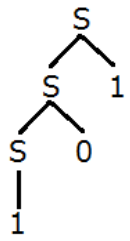

図 3　文法 G1 での文「101」の最右導出木

内部頂点には非終端記号が，葉には終端記号がラベルとして付けられる. 特に, 根には開始記号が必ずラベル付けされる．また，葉のラベルを左から順番に並べた記号列は導出された文となっている．このような木を解析木，または導出木と呼ぶ. 文法 G と文法 G によって生成される文が与えられるとき，この解析木，またはこのような情報を求めることを構文解析するという．

この解析木は同じ文を扱っても生成規則によって様々な形と変化する．この導出過程において最も左の非終端記号から導出を行うものを最左導出とよび，最も右の非終端記号から導出を行うものを最右導出とよぶ．文法 G によって生成される任意の文に対して解析木が一意に定まるとき，その文は曖昧ではないといい，複数の解析木が存在するときその文は曖昧である，という．逆に，曖昧な文を生成する文法は曖昧な文法と呼ばれる．

2.2　字句解析

字句解析系[2]は構文解析系から呼び出される手続きや関数として扱われる．呼び出されるたびに原始のプログラムを読み込んでその先頭から字句を分離し，それを構文解析系に引き渡す．また，その字句に付属する情報が存在すればそれらも引き渡す．字句は取り扱いの便利さから整数値にコード化されているのが普通であるが，字句の区別を字句コードと属性とでどのように分担するかは言語使用と次のフェーズである構文解析系の都合による．通常は個々のキーワード，演算子，区切り記号には別々の字句コードを与え，識別子，定数，文字列にはそれぞれに全体として 1 つに字句コードを与える．各識別子，定数，文字列は属性で区別する．字句解析の役割は主な 3 つである．

- 原始プログラムに則って字句を分離して返す．その字句に属性が存在すればその属性も返す．ここでの属性については，識別子の文字列や定数値などをそのまま返してその後の処理を構文解析に任せる方法と，記号表や定数表に登録してその表内のエントリへのポインタを属性とする方法がある．
- 空白やタブ，改行コードなどのプログラムに意味を与えないコメントなどを読み飛ばす．
- プログラムリストの出力やエラーメッセージ出力のために，行番号を教えたり入力行を保持したりする．

入力された文字列がどのような字句に解析されるかは正規表現で定義することができる．定義された正規表現の規則から非決定性有限オートマトン(NDA: nondeterministic finite automaton)に変換することができる．有限オートマトン(finite automaton)とは，有限の内部状態を持ち，与えられた記号列を読み込みながら状態遷移し，その記号列があるパターンを持つ列であるかを決定するものである．非決定性有限オートマトンとは、入力によらない状態遷移(空列記号に対する状態遷移)をもち、それは非決定的に遷移してもしなくてもよいと状態をもつものである。NDA では、空の状態遷移に対して、状態遷移するかしないかの両方の可能性を調べなければならないので、実際にそのまま実装すると効率がわるい。そのため、非決定的な遷移を取り除き、 決定性有限オートマトン(DFA: deterministic finite automaton) に変換する。決定性有限オートマトンは空記号による遷移がなく， 1 つの状態から同じ記号による異なった状態への遷移がないものをいう．しかし，この作られた DFA が最適なオートマトンである

という保証はないため, ここからさらに DFA の状態数を最小にする必要がある場合もある．最小オートマトンにするために，必ず区別しなければならない状態を判断する基準を以下に示す.

- 最終状態と非最終状態は区別しなければならない.
- 同じ入力記号に対して異なる状態に遷移するような状態は区別しなければならない.
- それ以外の状態は以降同じ動作をするので一緒にして良い.

以上により，正規表現から最小オートマトンを作成することで基本的な字句解析を行うことができる.

2.3 匿名化

2.3.1 準備

本論文で扱うデータテーブルとは項目である属性とその属性値である．属性にはそれぞれ「識別子」と「準識別子」が存在する．識別子はその属性がもつ値が直接個人を特定できるような属性をいう．準識別子とは単体では個人を特定することはできないが，それらを複数組み合わせることで個人情報につながる可能性のあるような属性をいう.「属性値」とは実際に扱われるデータ(レコード)のことである.「特定」とはその情報によって直接個人が判別できることをいい,「識別」とは誰だかは判別できないがその情報によって 1 人に絞り込める状態のことをいう.

「一般化」とは何もしてない素のデータを匿名性のあるデータにするために行う抑制操作のことである. ここで,「抑制」とはデータの全部分, または一部部分の非表示のことを示す.

2.3.2 k-匿名化

データテーブルに対して検索や参照をするときに，属性内で同じ組み合わせを持つデータが k 個以上存在するような状態を「k-匿名性(匿名条件)を満たす」といい，実際にそのデータを加工することを「k-匿名化」と呼ぶ．k-匿名化には以下の 2 フェーズが存在する.

(1) 要素数 k 以上のクラスタ構成操作

(2) クラスタ内データの一般化操作

(1)は同じグループのレコードを k 個以上にするための k-分割の作成であり，(2)は同じグループになったレコードの属性値を一般化する方法である.

クラスタリング[4]のような一般化方法を用いる場合，まず(1)によって k 分割を行った後, (2)のレコード一般化を一度だけ行って終了する．Datafly やμ –Argus[7] [8]では，先に(2)のデータ一般化を行い，(1)によって k 分割を行い，匿名条件に合致していなければ再び(2)のデータ一般化を繰り返す．このデータの抑制方法にも様々なものがあり，例えば，数字ならば最小位から抑制，英字ならば 1 単語ごとに抑制する手法もある．カテゴリ値 (単語) ならば全抑制と抑制の 2 択しかない一般化の手法も存在する,

2.4 k-匿名化の既存研究

2.4.1 クラスタリング

クラスタリング[4]は，データレコードがそれぞれ少なくとも k 個のレコードを有する複数のグループに分割し，その中の準識別子が同じ値になるように一般化する手法である．グループに分割するときに, その数を固定にする(レコード数/k)方法[5]と可変にする方法[6]が存在する．また，一般化する方法でも，属性内の全てのレコードを一般化するような方法と局所的に一部を一般化する方法がある.

この手法の長所はデータ損失が少ないことであり，短所は実行時間が長いことである．ここで最も重要なことは，匿名化を行うときにデータの抑制が少なくてすむように，できる限り類似したレコードを同じグループに入れることである.

2.4.2 Datafly

Datafly[7][8]はテーブル全体を対象にして一般化を行う方法である．この手法はデータテーブルにある属性全てに他視して k-匿名性を満たすように部分的な抑制，またはそのレコードごと消去する．長所は属性間でデータを組み合わせても必ず k 個以上の同じデータが存在するため，必ず k-匿名性を持った結果テーブルを作成し，個人情報を保護することが出来ることである．短所は一般化の際に匿名化の必要がない属性も一緒に一般化してしまう可能性があることである．例えば，表 1 のデータテーブルを一般化するとき，3 行目のデータは匿名性が保たれないとして完全に抑制される(表 2). この

結果，データが必要以上に抑制する可能性がある．

表 1 素のデータテーブル

性別	生年月日	郵便番号
男	1992/1/1	111-111
女	1992/2/1	222-222
男	1992/3/1	222-222
女	1992/4/1	222-222
男	1992/5/1	111-111

表 2 Datafly 用いた k=2 の場合の結果テーブル

性別	生年月日	郵便番号
男	1992	111-111
女	1992	222-222
女	1992	222-222
男	1992	111-111

2.4.3 μ –Argus

μ –Argus[7][8]は，属性内でk-匿名化を満たすのに必要なレコードだけを最小限の抑制にて行う手法である．この手法の長所は結果テーブルのデータの抑制が小さいことである．ある属性を対象としたときにその中で匿名化が必要なレコードのみを抑制するので，他の関係ない属性まで一般化することはない. 例えば, 表 3 のようなデータテーブルが存在する．属性「年代」では「1992」と「1993」の 2 種類のレコードが存在し，属性「郵便番号」においては「111-111」と「222-222」の 2 種類のレコードが存在する．ここで属性内において k=3 での k-匿名条件を満たすためには，レコード「1992」の最下位値「2」と，レコード「222-222」の全文字列の抑制を行う．これによって属性内において k-匿名条件を満たしたデータテーブルが作成される(表 4). 局所的にデータが抑制されているので，直接的には属性内で匿名条件を満たしているように感じるが, 実際にはデータとして識別をすることが可能である．これをテーブル単位で考えてみると，個人を識別できる可能性は残ってしまう．

表 3 素のデータテーブル

性別	年代	郵便番号
男	1992	111-111
男	1993	222-222
男	1993	111-111

表 4. μ -Argus を用いた k=3 の場合の結果テーブル

性別	年代	郵便番号
男	199*	111-111
男	1993	*******
男	1993	111-111

2.4.4 ミクロアグリゲーション

ミクロアグリゲーション[9]は，グループ化した属性内の一般化手法の１つである．数値ならば平均値に，カテゴリ値ならばより大きな区分(上位単語)に置き換える方法である．例えば，都道府県名の属性に「東京」「千葉」「神奈川」が存在した場合，「関東」という上位単語にまとめることができる．この手法の長所はデータ損失が少なく，属性全体を見て一般化できることである．短所は分散が大きいグループではデータ損失が大きくなることである．

3. 提案手法

現在，個人情報は多岐にわたり，登録される情報にもさまざまな属性が存在する．一言に数値といっても年齢や郵便番号，携帯番号，時刻などが存在し，抑制の仕方も用途によって変える必要があるだろう．文字列のメールアドレスなどではドメイン以下の長さは任意であるし，住所に至ってはカテゴリ値である日本語の複数の組み合わせと，番地番号である数値の混合で示されることがしばしばである．

このようにデータが構造化している場合，その構造の一部を匿名化するためにはデータを分解し，抑制したい部分の要素を取り出すことが必要である．データを分解するためにはその属性値に適した形の構文解析を行う必要があり，この構文解析には属性のもつ特徴的な記号を用いることが重要である．ここで，「属性のもつ特徴的な記号」とは，その属性のデータに存在する文字列(文字単体も含む)を用いて，その特徴的な文字列の前後で文字列を区切ることができる記号のこと

である．例えばデータ属性がメールアドレスであり，実データに「1111@mail.jp」と「2222@Mail.com」が存在した場合、この属性の属性値に存在する共通の区切り記号は「@」と「.」である．この区切り記号を用いて構文解析することでデータを構造化し，その構文要素に一般化の順序付けを行うことが出来れば，より適切な一般化を行うことが出来ると考える．

本論文では，k-匿名化を行うにあたっての操作を 2 フェーズに分けた場合の「(2)データを一般化する動作」に着目し，属性ごとに適した型(属性型)を定義し，その型に応じてデータに一般化順序を与え，その与えられた順序に基づいて文字列抑制の一般化を行う API を提案する．

3.1 属性型の定義

属性型の定義とは，データに合わせた解析をするための文法を定義することである. 本論文ではこの文法に従う式を「匿名化式」とよび, 匿名化式をユーザ側から与えられることで, その匿名化式特有の構文規則を生成することができる．

匿名化式は「’ ’」で囲まれた区切り記号と，「< >」で囲まれた順位付けの数字によって生成される．この区切り記号によって字句解析が行われるため，属性値に対して特徴的な記号を用いることが重要である．同じ場所で複数の区切り記号を指定したい場合(「都道府県」など)，は「｜」の記号を用いて区切ることとする．

次に，構文要素に対する順位付けを説明する．ここでは，区切り記号などによって分解された構文要素に匿名化を行う順番をつけるための順位付けを行う．この要素の順位付けもユーザ任意で行うことができ，順序が低いほどその構文要素は先に抑制される．

匿名化式とは別に，属性型を定義する上で必要なものとして区切り記号に対する優先順位と結合性がある．構文解析を行うときに記号による優先順位が必要になる場合，優先順位欄(order)に解析優先順位が高いで左から順に解析を行う記号を「( )」に囲んで記述する．もし同順の文字列が存在する場合は「|」の記号を用いて記述する．例えば，区切り記号 A, B，C に対して A を他の記号より優先順が高く，B と C を同列に扱いたい場合は「(A)(B|C)」となる．

同様に，結合性についても追加記述を行うことがある．このアルゴリズムは通常，最左導出を行うが最左導出による構文解析だけでは表すことができない解析が存在する．このため, 必要に応じて特定の区切り記号を左再起欄(left)に「( )」に囲んで記述し，最右導出を行うように定義する．「delete」は抑制時の区切り記号を表示するかどうかの判断に用いる．

実際に、例として属性が①メールアドレス，②住所の匿名化式と区切り記号に付加された条件を記述する

① メールアドレス

<1>’-’<2>’@’<3>’.’<4> (1.1)

order：(@)(-|.) (1.2)

left：(.) (1.3)

delete: (1.4)

② 住所

<3>(’都’| ’道’| ’府’| ’県’)<2>

(’市’| ‘区’| ’町’| ‘村’)<1> (2.1)

order： (2.2)

left： (2.3)

delete:(都)(道)(府)(県)(市)(町)(村) (2.4)

メールアドレスの匿名化式は 2 種類の区切り記号と 4 つの匿名順位をもつ要素からなる．優先順位(order)と左結合性(left)の条件が記述されているため，これらを踏まえた構文規則が生成される．住所は異なる 4 つの区切り記号が同じ区切り場所に記入されている．これは，実データがいずれかの区切り記号と一致する文字列を解析する．

3.2 属性型に応じた一般化アルゴリズム生成

定義された匿名化式と条件から構文規則を生成し，実データに対して構文解析を行う．さらに，データ一般化アルゴリズムの API を設計する．

3.2.1 構文規則の生成とデータの構造化

与えられた匿名化式と条件を読み込み，その属性型による構文規則を生成する．さらに，区切り記号を用いて実データを字句解析し，生成した構文規則により構文解析を行い，実データを構造化する(図 4). そして構造化したデータの構文要素に，匿名化式に当てはまるように匿名順序を割り当てる．

ここで，さきほど定義したメールアドレスの属性型を用いて構文規則を生成し，実データ「AA-aa@mail.ne.jp」に対するデータ構造化と匿名化の順序付けを行った例を以下に示す．

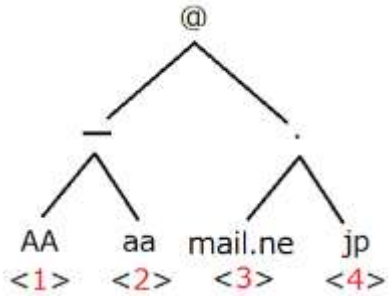

図 4　メールアドレスのデータの構造化の例

まず，データを構文規則に基づいて優先順位の高い区切り記号「@」から解析を行う．次に「@」の左辺から解析するが，「-」は右結合であるため通常に解析を行う．その後「@」の右辺の解析を行うが，「.」は左結合と定義しているので，「mail.ne.jp」は「mail.ne」と「jp」とで分解される，

構造化が完了したら，匿名化式による構文要素の順位付けを行う．重要なことは，匿名化式の型に構文要素の場所と数を合わせることである．この匿名化式(1.1)では「@」の右辺部分は 1 つの区切り記号「.」と 2 つの順位付けされる文字列の集合しか存在しない．そのため「mail.ne」はこれ以上解析が行われることはない．

3.2.2 匿名レベルに応じたデータの一般化

構文解析された構文要素に対して，匿名化式より匿名順位が付与される．この付与された数値と匿名レベルが対応して，匿名レベル以下の数値が付与された構文要素を抑制する．もし，順位最大値より大きなレベルを入力した場合，これは全抑制につながる．

3.2.3　データー般化アルゴリズムの API 設計

属性値一般化のサブシステムとして，既存の k-匿名化システムと連携させるための API を設計する．この API では，属性型を示す匿名化式と条件を記述したファイル(テキストファイル)を初期化で与え，入力として匿名化を行いたい実データの文字列，使用属性ファイルの選択，匿名化のレベルを示す数値を与える．出力は数値で示されたレベルまで抑制が行われた実データの文字列である．図 5 は，設計した API の概要である．

図 5　API の動きの概要

4.　提案システムの利用

提案手法 API を利用したデータ一般化プログラムを実装する　このプログラムの目的は，与えられたデータの集合を k-匿名性のあるデータテーブルに書き換えることである．既存手法 μ –Argus を参考にプログラムを実装し，提案 API は実データの抑制を分担する．プログラムには属性型ファイルとデータファイルが渡される．プログラム内では必要な引数を API に引き渡し，API では与えられたレベルに抑制された文字列が出力される．この抑制されたデータの集合が k-匿名性を満たさなくなるまで提案 API はデータの抑制を行う． k-匿名性を満たしたら，プログラムは終了する．データには表 5 のデータテーブルを用いる．

表 5．データテーブル

メールアドレス	住所
AA-aa@mail.ne.jp	東京都新宿区新宿
BB-b-b@Mail.com	東京都新宿区歌舞伎町
CC@email.jp	東京都渋谷区道玄坂
DD-dd@uec.com	東京都中央区銀座
EE-e-e@email.jp	東京都西新宿
FF@mail.ac.jp	東京都渋谷区神宮前
GG-gg@Mail.com	東京都新宿区大久保

4.1　k-匿名化プログラムとの結合方法

提案手法 API は実装したプログラムから属性型を示したファイルとデータファイルから取り出した 1 つも実データ文字列，さらに匿名レベルの数値を引き渡される．この数値の初期値は匿名化の最高レベルとし，呼び出されるごとにそのレ

ベルを下げる．この API からの出力は，レベルに応じて抑制された文字列である．

この抑制された文字列の集合に対して，実装したプログラムを用いて k-匿名化条件を満たすか満たさないか，実行を継続するかどうかを判断する．

4.2 k-匿名化プログラム

ここでは k-匿名化の条件を満たす文字列は何か，満たさない文字列は何かを定義する．まず匿名優先度が低い(数字が大きい)構文要素をカウントして，その数が k 匿名条件を満たすならば次の構文要素のカウントを始める．条件を満たさない場合，次の検索は中止される．これがプログラムの基本の動きであるが，様々な場面が存在するのでその対処を示す．

- 構文要素が存在しない

  その構文要素は強制的に false として扱われ，その数もカウントされる．

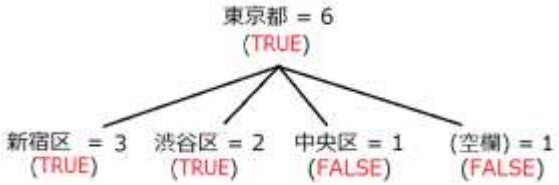

図 6 住所の構文解析

- 条件を満たさない構文要素のカウント数の合計が k 以上存在する

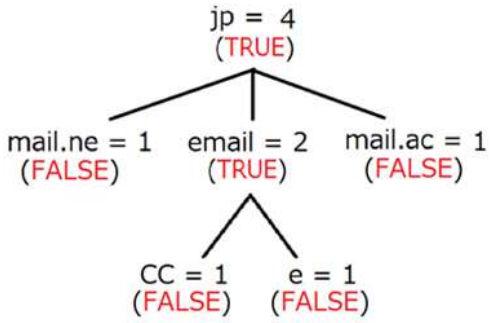

図 7 メールアドレスの構文解析(1)

条件を満たした構文要素のみ，次の検索に進むことができる．2 段目の要素は 3 種類存在するが，そのうち「mail.ne」と「mail.ac」の 2 つは匿名条件を満たしていない．しかし，その 2 つのカウント数を合計すれば k よりも大きくなるので，単独で条件を満たしている残りの「email」は次の検索に進むことができる．．

- 条件を満たさない構文要素のカウント数の合計が k 以上存在しない

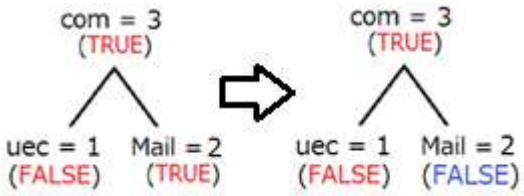

図 8 メールアドレスの構文解析(2)

条件を満たした構文要素の中で，最もカウント数が少ない構文要素を「条件を満たさない」文字列に変更し，抑制の対象とする．2 種類の要素に対して「Mail」は匿名条件を満たすが，「uec」は条件を満たさない．また，条件を満たさない他の要素も存在しないため，「Mail」を条件を満たさない文字列として扱う．条件を満たすものが複数存在した場合，条件を満たす要素の中で一番カウント数の少ない文字列を選択する．もし条件を満たす要素のカウント数が同等だった場合，その中で構文要素の登録が最も遅かったものが条件を満たさない文字列として扱われる．

4.2.1 実データの一般化

抽出した文字列のカウント数が匿名条件を満たす場合，文字列抑制されず次の文字列部分に移る．匿名条件を満たさない場合，そのレコードの現順位以下の構文要素は抑制される・今回の実験では抑制する文字列の文字数は一定である．

メールアドレスの属性型の式(1.1)，(1.2)，(1.3)，(1.4)と住所の属性型の式(2.1)，(2.2)，(2.3)，(2.4)を用いて表 5 のデータを k=2 の条件で匿名化させる (表 6)．定義式のファイルは初期化時に与えるものとし，抑制する文字列の属性に伴って属性ファイルを選択，使用し実行するとする．

表 6. 結果テーブル(k=2)

メールアドレス	住所
******.jp	東京都新宿区**
******.com	東京都新宿区**
****@email.jp	東京都渋谷区**
******.com	東京都****
****@email.jp	東京都****
******.jp	東京都渋谷区**
******.com	東京都新宿区**

5. 考察と今後の課題

本研究では正規表現にて構文解析を行った．その際に最も問題だったのは市区町村と字の部分である．字とは市区町村と丁番地の間にある地区名である．例えば「五日市市××」が市区町村名と字にあたる文字列だったとすると，先述した住所の属性型の式(2.1)では「五日(市)」と「市××」に分解されてしまうことになる．このようになる原因は，区切り記号が通常の文字としても活用されていることにある．また，市区町村名を飛ばして都道府県と字のデータが存在したとすると，字に「市」「区」「町」「村」のどれかの文字が入っていたら，それは市区町村扱いになってしまう．これらに対処するためには，例外を除いて解析を行うようにしなければならない．正確な解析を行うためには住所に特化したシステム等をこの API に組み込むか，定義する属性型の匿名化式を複雑にすることが必要である．

この匿名化式では区切り記号を用いて解析を行うため，共通する記号が存在し，かつその記号を匿名化式に正しく記入する必要がある．あまりに属性内のデータにばらつきがあり型が一致しない場合，文字列の解析がうまく行われず最後の解析までデータの大部分が残ってしまう．このような属性に本研究の API は有用ではないことがわかる．一方で，データの細かい解析を行いたい場合はそのすべてを匿名化式に記述しなければならない．加えて優先順位や結合性の定義を追加する場合，さらに記述を行う必要がある．記述量が多いほど構文解析の幅は広がるが，ユーザ側への負担を考えると妥協点をどこにするかは論点だと考える．

今後の課題としては，今後の課題としては，特殊な複合データにも対応するアルゴリズムを考えたいと思う．複合データとは，特殊記号なしにデータが混合している場合を示す．例えば本論文で例にあげた住所だが，一般的には番地やマンション名，その部屋番号まで記入する．ここでの字と丁番地の間にはなにも記号は存在しない．これより，今のアルゴリズムでは字と丁目が分解されることなく抽出される．これを分解する特殊な解析を行うことでさらに多くのデータを解析できると考える．

また，本研究では区切り記号を用いない年齢のような文字列に対する匿名化式を考えなかったため，そのような場合の構文解析や一般化法も考慮したいと思う．

参考文献

[1] 佐久間 淳, 小林 重信, “プライバシ保護データマイニング”, 人工知能学会誌 Vol.24, No.2, 2009

[2] 辻野嘉宏, “形式言語と形式文法” 情報工学入門選書　第 10 巻　コンパイラ”, pp.27-70, (株)昭晃堂, 1996.

[3] 田中穂積, “自然言語の高速な構文解析　―一般化 LR 構文解析アルゴリズム―”, 電子情報通信学会誌, vol.77, No.10, pp.1035-1042, 1994.

[4] 新井淳也，鬼塚真，塩川浩昭，“クラスタリングと空間分割の併用による効率的な k-匿名化，”DEIM Forum, 2014.

[5] J. Domingo-Ferrer and V. Torra. Ordinal, “Continuous and Heterogeneous k-Anonymity Through Microaggrega-tion,” *Data Mining and Knowledge Discovery*, vol.11, no.2, pp195-212, 2005.

[6] A. Solanas and A. Martinez-Balleste, “V-MDAV: A Multi-variate Microaggregation With Variable Group Size,” In *17thCOMPSTAT Symposium of the IASC*, 2006.

[7] 村本俊祐，上土井陽子，若林真一，“k-匿名性を利用したデータ一般化によるプライバシー保護，”DEWS，2007.

[8] L. Sweeney, “Achieving k-anonymity privacy protection using generalization and suppression,” International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, vol10, no5, pp.571–588, 2002.

[9] M.K.Wolf, “Microaggregation and disclosure avoidance for economic establishment data,” annual