DBSJ Journal, Vol.11, No.1
June 2012

DB抽象化とゲーム理論に基づくマイクロブログからの構造データ抽出GWAPの開発

Applying DB Abstraction and Game Theory to Development of a GWAP for Extracting Structured Data from Microblogs

福角 駿♥　森嶋 厚行♦
品川 徳秀♦　北川 博之♣

Shun FUKUSUMI　Atsuyuki MORISHIMA
Norihide SHINAGAWA　Hiroyuki KITAGAWA

計算機ネットワーク技術の発達により不特定多数の群衆に作業を委託するクラウドソーシングに注目が集まっている．クラウドソーシングで用いられるアプリケーション手法の一つとして，ゲームの副作用として何らかの目的を達しようとするGWAP (Game With a Purpose)がある．本論文ではマイクロブログを対象とし，GWAPを用いて非構造データから構造データを抽出する手法を提案する．ESPゲームなどの既存GWAPと比べた提案GWAPの新規性は，プレイヤの入力として，データそのものだけでなく計算機がデータを抽出するためのルールも許可することである．ゲーム理論に基づき設計することにより，構造データ抽出の作業を，徐々に人手から計算機に移行させるという，伝統的なソフトウェア設計ではスコープ外であった設計の試みを行っている．我々は，ゲーム理論を用いて提案GWAPの妥当性を証明し，また，実験により構造データ抽出の作業が徐々に人手から計算機に移行することを確認した．

As the computer network technology evolves, crowdsourcing is attracting much attention. GWAP (Game With a Purpose) is a crowdsourcing technique to use games to generate useful side effects. This paper proposes a GWAP to extract structured data from unstructured data, focusing on microblogs. The novelty of the proposed GWAP is to allow the users (players) to give not only data, but data extraction rules as the inputs in the game play. We conducted a novel attempt of software design to address both of the human and machine computation in which the process of data extraction is gradually shifted from humans to the machine. We used the game theory to prove the validity of the GWAP, and our experimental results showed that the process of data extraction was gradually shifted from humans to the machine.

1. はじめに

近年，Twitter を代表としたマイクロブログを通じて発信される情報の量が増大している．東日本大震災においては，つくば市においても電車の営業停止，長期の断水被害などの影響があったが，電車の運行情報や給水所などについての情報源としてマイクロブログなどの情報をまとめた「まとめサイト」が役に立った．このマイクロブログに代表されるように，Web上には役に立つ情報が多く存在するが，これらは非構造データであり，その後の再利用や管理が困難である．したがって，非構造データから構造データを抽出する数多くの研究がこれまでに行われてきている[1][2]．

非構造データから構造データを抽出する方法はいくつかに大別される．利用状況や対象の多彩さなどにより，それぞれ一長一短である．第一に，ラッパーなどの自動変換ソフトウェアを構築して抽出する方法がある．第二に，自然言語処理技術により構造データを抽出する方法がある．第三に，人手により構造データを抽出する方法がある．第三の方法には，まとめサイトの管理者のように特定の人物が行うものと複数の人物が参加して行うものがある．後者には，クラウドソーシングに代表されるように不特定多数の群衆が行うものから，比較的少数の協力者によって行われる手法まで含まれている．特にクラウドソーシングは近年注目を集めているアプローチである．

本研究では，第三のアプローチで利用可能な手法として，GWAP (Game With a Purpose) [3]により，マイクロブログから構造データを抽出する手法を提案する．GWAPとは，ゲームを行う副作用として，何らかの目的を達しようとするものである．たとえばESP Game [4]では，画像に対するキーワード当てゲームを用いて，画像のメタデータ収集を行う．

本研究でマイクロブログを対象とする理由は，ハッシュタグなどを利用して同一テーマの情報を比較的集めやすく，また，広範に多様なデータを収集するのが容易なため，構造データ抽出の適用例として向いていると考えられるからである．技術的にはマイクロブログに限らずWeb上の非構造データに適用可能である．

本論文の貢献は次の通りである．

(1) 構造データ抽出へのGWAPの適用. これまで，多くのクラウドソーシングやGWAPがデータ指向であるにもかかわらず，これらがデータ工学の視点で語られるのはクラウドソーシングによるSQL処理に限定されていた．本論文では，DB抽象化の道具の一つであるDatalog [5]を拡張することにより，非構造データからの構造データ抽出GWAPを自然な形で記述できることを示す．

(2) 新たなゲーム構造の提案. 既存のGWAPによるデータ生成では，複数のプレイヤがデータを入力し，そのデータそのものを生成物とすることが一般的である[4][6]．これに対し，本論文では，プレイヤが単にデータを入力するだけでなく，データ抽出のためのルールという“メタなデータ”を入力するという選択肢を持つ新たな構造を持つゲームを提案する．

(3) ゲーム理論に基づく設計の妥当性の証明. 提案GWAPが合理的なプレイヤの下でデータの品質を保ち，必ず終了することを示す．

(4) 実験による検証. 実験により，データ抽出ルールをプレイヤが入力可能とすることが，GWAPによる構造データ抽出にどう影響を与えるのかを明らかにする．特に，ゲームを適切に設計することで，データ抽出作業を人手から計算機に徐々に移行できる事を確認したのは重要な貢献と考えている．

本論文は次のように構成される．2章では，関連研究・システムについて述べる．3章では，提案システム TweetPecker について議論するために必要な準備を行う．4章では，GWAPによる構造データ抽出システム TweetPecker について提案する．5章では，提案システムが合理的なプレイヤの下で正しく動作し，かつ，必ず終了することを示す．6章では実験とその結果について説明する．7章は，まとめと

♥ 学生会員　筑波大学大学院 システム情報工学研究科
shun.fukusumi.2011b@mlab.info
♦ 正会員　筑波大学 図書館情報メディア系
{mori, siena}@mlab.info
♣ 正会員　筑波大学 システム情報系
kitagawa@cs.tsukuba.ac.jp

今後の課題である.

2. 関連研究

1 章で説明したように, 非構造データから構造データを抽出する方法は 3 つに大別される. (1)ラッパーなどの自動変換ソフトウェアを構築, (2)自然言語処理技術により構造データを抽出, (3)人の力によってデータを抽出.

(1)の例としては, Web データを XML に変換するラッパーを構築する研究[1]がある. ラッパーを構築して抽出する方法は, 抽出の際に人手が不要であるという利点がある. しかし, ラッパーを作成するためには十分な量のデータが必要であったり, ラッパー構築そのものにコストがかかったりする事が問題である.

(2)の例としては, Wikipedia と Web の情報からオントロジーを自動構築する研究[7]がある. 自然言語処理により抽出する方法は, (1)と同様に, 抽出の際に人手が不要であるという利点がある. しかし, この方法はドメインを限定する等の工夫を行わなければ必ずしも正しい結果を得ることができない.

(3)の例としては, まとめサイトの管理者が抽出を行うように特定の人物が行うものと, 管理者が第三者に依頼し, 分担して抽出を行うものがある. これらは, 人手で行うため幅広いドメインに対応できたり, 情報の誤表記にも対応できたりと柔軟性が高い. このうち, 前者は特定の人物で行われるためデータの品質を保ちやすい. しかし, 前者はコストがかかり, 大量のデータに対応するのは難しい. 後者は, クラウドソーシングに代表されるように不特定多数の人物が行うものや比較的少数の協力者によって行われる手法まで含まれる.

クラウドソーシングによるアプローチは多くの人々の参加によりデータを抽出するので各々の負担は小さいという利点がある. 一方, 不特定多数の人物によりデータを抽出するため品質を保つことは比較的難しいといった欠点がある. しかし, 現実には上記の(1)(2)が時間的, 技術的理由から採用できず, 可能なアプローチが(3)に限定されることも多いことから, このアプローチの研究は重要である.

提案する TweetPecker は, (3)の手法の一つとしてとらえることができる. 比較的少人数のグループから不特定多数のクラウドソーシングまで適用可能であるため, 利用可能な人的リソースの規模に応じて柔軟に対応できるという利点がある.

3. 準備と問題

3.1 Cylog

GWAP による構造データ抽出を議論するため, 本論文では, CyLog 言語[8]を用いる. CyLog は, Datalog [5]に似た表記法を利用し, 人と計算機の処理をシームレスに記述することが可能なプログラミング言語である. CyLog では, ポイントの付与を含む TweetPecker の全体を記述できるが, 本論文では簡単化のため, TweetPecker の主なデータフローのみを CyLog で説明する. 本節では, そのために必要な CyLog の説明を行う.

CyLog のコードは Datalog と同様, **ファクト**もしくは**ルール**の並びとして記述する. ルールは, 右辺の条件を満たすデータが存在する時, 左辺のデータが存在することを表す. 例えば, データベースに格納されている画像に人がキーワード付けを行う処理を考える. この処理の内容が, (1) データベースに格納されている画像 i に関して, (2) 登録された複数の人 p がキーワードとなる語 w の入力を行い, (3) その w が辞書に載っていればメタデータとして採用するとする. これは次のルールで記述される.

```
Metadata(i, p, w)
   <- Image(i), Person(p),
      Keyword(i, w)/open[p], InDict(w);
```

ここで, Image, Person, Keyword, InDictはそれぞれ, 画像, 登録された人, 画像に対するキーワード, 辞書を表す述語である. これらの述語を用いて記述されたImage(i)などの式を原子式(Atom)と呼ぶ. 右辺の原子式は左から評価され, 引数に指定された変数が束縛される. ここで/openは未束縛変数が表す属性値をプレイヤに問い合わせる指定であり, オプションとして誰に聞くかを指定することができる([p]). /openが付いた原子式を特にオープン原子式(Open Atom)と呼ぶ. 与えられた入力はあたかもリレーション中に存在していたかのように評価が行われる. オープン原子式に未束縛変数が存在しそのような処理の例として, データベースが町の写真の集合を格納している時, その各写真iが現時点でも適切かどうか(新しいビルが建ったなどの理由で不適切になっていないか)を判断する処理を考える. これは次のように記述される.

```
ValidImage(i) <- Image(i), Valid(i)/open;
```

上記では, i が Valid であるかどうかの判定を人に問い合わせ, 真と判定された画像のみを ValidImage リレーションに格納する.

3.2 TweetPecker 開発における問題

CyLogを利用すると, 人手を利用してTwitter上の天気に関する発言をもつリレーション Tweet(tw)から構造データ Weather(place, date, condition, time)(ただし, place は場所, date は日付, condition は天気, time は時刻である)を抽出する処理は次のように記述できる.

```
Weather(place,date,condition,time,tw)/open[p]
     <- Tweet(tw), Person(p);
```

しかし, このコードでは単に任意の入力を受け取るだけであり, データの品質を保証することができない. また, データ抽出のペースの上限が人手によるデータ入力の速度で決まってしまう.

本論文では, これらの問題に配慮して, 構造データ抽出のための GWAP である TweetPecker の開発を行う. 以下では次の順に説明する.

(Simple TweetPecker) 上記のコードとほぼ同じであるが, 人の合理性を前提として結果の正しさを保証するよう設計した GWAP.

(Full TweetPecker, もしくは単に TweetPecker) 人に単なるデータ入力だけではなく, 計算機がデータ抽出するためのルールを入力する事を許した GWAP. これにより, 抽出作業が徐々に人手から計算機に移行される. また, Simple TweetPecker と同様に設計の妥当性を証明する.

4. 提案システム TweetPecker

本章では, GWAPによる構造データ抽出システムTweetPeckerの説明を行う. まずは, Simple および Full TweetPecker に共通の概要を説明し, 次にそれぞれについて説明する.

4.1 TweetPecker の全体像

TweetPecker は, GWAP により Twitter 上の発言(以下**ツイート**と表記)からリレーションを抽出するシステムである. ここで, ツイートは発言内容だけでなく発言者, 発言日時などの情報をもつ. 以降では, GWAP を行う人々(2 人以上)の各自をプレイヤと呼ぶ.

TweetPecker による構造データ抽出の流れを次に示す(図 1).

1. システム利用開始時に必要な入力は, 対象となるツイート集合を表すハッシュタグと, 抽出するリレーションのスキーマ $St(a_1,a_2,\ldots,a_N,tw)$ (例えば, Weather(place, date, condition, time, tw))である(図 1(1)). ここで, tw はツイートを格納する省略可能な属性である.
2. 入力されたハッシュタグを含むツイートの集合を表すリレーション Tweet(tw) と, そこからスキーマ St を持つ空のリレーショ

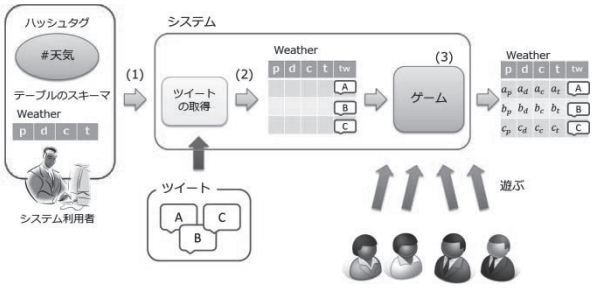

図 1 TweetPecker の概要
Fig. 1 Overview of TweetPecker

ンを作成する(図1 (2)). 例えば, システム利用者がハッシュタグ「#天気」とスキーマ Weather(place, date, condition, time, tw) を入力すると, "#天気"というハッシュタグを含むツイート集合を表すリレーション Tweet(tw) を作成し, リレーション Weather を作成する.

3. その後, プレイヤにゲームを行わせることで各 tw に対するリレーション Weather の各タプルの属性値を決定する(図1 (3)). 決定された値に応じて, 各プレイヤにポイントを付与する.

4.2 Simple TweetPecker: データ入力のみの GWAP

Simple TweetPecker では, 複数のプレイヤから入力されたデータが一致した場合にそのデータを抽出データとして決定する. 以下では, 簡単化のため Weather の抽出データを place に限定する(この簡単化は一般性を損なわない). 本節では, まず CyLog を用いて Simple TweetPecker のデータフローを説明する. 次に, ユーザインタフェースを説明する. 最後にポイントの付与について説明する.

4.2.1 データフロー設計

プレイヤの集合を表すリレーションを Player(p) とし, ツイートの集合をリレーション Tweet(tw) とする. また, 抽出結果を格納するリレーションを Weather(place, tw) とする. このとき, Simple TweetPecker のコードは次の通りである.

```
1. Input(place, tw, p)
    <- Player(p), Tweet(tw),
       Extract(tw, place)/open[p];
2. Weather(place, tw)
    <- Input(place, tw, p1),
       Input(place, tw, p2), p1!=p2;
```

各ルールの説明を次に示す.

ルール 1. 各プレイヤ p は, ツイート tw に対応する属性 place の値を入力する. 結果は Input リレーションに格納する.

ルール 2. 同一のツイート tw に対して, 2 人のプレイヤ p1, p2 の入力が等しい場合, Weather リレーションの属性 place の値として決定し, 採用する.

4.2.2 ユーザインタフェース

ルール 1 の属性値の入力では, プレイヤは割り当てられたタプルの各属性値を図 2 の HTML フォームへ入力する. ただし, 例えば図 2 の天気属性のように既に属性値が決定しているものは入力できない.

TweetPecker のインタフェースでは, プレイヤは次の 2 つの入力方法がある. 1つは, 値をテキスト欄へ**直接入力する**方法である. もう1つは, 値を**選択入力する**方法である. Simple TweetPecker では, 選択入力は「値なし」という値のみである.

入力に対しては, 次節で説明するポイントが与えられる. 本インタフェースでは, 累積ポイントを常に棒グラフでアニメーション表示し, 自身と他のプレイヤとの現時点の勝ち負けをわかりやすくすることによって, 互いに競いあう状況を作るよう工夫している.

4.2.3 プレイヤに与えられるポイント

ID	ツイート	発言日時	発言者
73	はままつ早苗のお天気情報ー！！ 今日13日の静岡県西部のお天気は晴れでしょう。#天気 #兵松	11月13日(日) 06時08分	Hamamatsu_Sanae

属性名	決定済みの値	選択入力	直接入力
地域		◎値なし	
天気	晴れ	○値なし	
日付		◎値なし	
時間帯		◎値なし	

送信 リセット

図 2 Simple TweetPecker における属性値入力フォーム
Fig. 2 Value input form of Simple TweetPecker

Simple TweetPecker においては, プレイヤによるデータ入力時ではなく, 各属性値が決定されたときに初めて各プレイヤにポイントが与えられる. このとき, 与えられるポイントは次の通りである.

- 決定された属性値が直接入力による値の場合, その値を入力したプレイヤに 5 点与える.
- 決定された属性値が「値なし」の場合, その値を入力したプレイヤ p_i と p_j に 3 点与える.

Simple TweetPecker は, 入力されたデータに関する単純な同調ゲーム[4]になっている. 同調ゲームとは, ゲームの振る舞いが同じであるときにポイントを与えるという構造をもつゲームである. これにより, 人が合理的に行動した場合, 正しい値がリレーションに格納される事を保証する. 単純な同調ゲームの証明は自明であるためここでは省略する.

4.3 TweetPecker: 抽出規則の入力を許可する GWAP

続いて, (Full) TweetPecker について説明する. TweetPecker ではプレイヤは単に属性値を入力するだけでなく, **データ抽出規則**を入力するという選択肢が与えられる. これにより, 序盤はプレイヤによるデータ抽出のみが行われるが, ゲームが進行するにつれて徐々に計算機によって抽出が行われるようになる(詳細は実験で説明する).

4.3.1 データフロー設計

TweetPecker のデータフローは, Simple TweetPecker のルール 1, 2 に加え, 次のルール 3,4,5 を加えたものである.

```
3. Rule(r, p)/open[p] <- Player(p);
4. Input(place, tw, "c")
    <- Tweet(tw), Rule(r, p),
       ExtractByRule(tw, r, place);
5. Weather(place, tw)
    <- Input(place, tw, "c"),
       Player(p1), Player(p2),
       Check(place, tw)/open[p1],
       Check(place, tw)/open[p2], p1!=p2;
```

各ルールの説明を次に示す.

ルール 3. 各プレイヤ p は, データ抽出規則 r を入力する. 現在の TweetPecker が入力として受け付けるデータ抽出規則は 4.3.2 節に後述する.

ルール 4. データ抽出規則 r を利用して, ExtractByRule 述語によりシステムがツイート tw から値 place を抽出し, Input リレーションに格納する. その際, データ抽出者として計算機を表す"c"を格納する.

ルール 5. 計算機が抽出した値が正しい値かどうかの真偽判定を 2 人のプレイヤ p1 , p2 のそれぞれに対して問い合わせる(Check (place, tw)/open[p1], Check (place, tw)/open[p2]). 両プレイヤが正しいと判定した場合に, Weather リレーションの属性 place の値として採用する.

4.3.2 許可するデータ抽出規則

TweetPecker では，プレイヤが入力したデータ抽出規則を利用し，計算機がデータ抽出を行う．したがって，データ抽出規則は，人間が比較的簡単に書けて，かつ，計算機が実行できるよう意味が明確であることが望ましい．現在，TweetPecker が受け付けるデータ抽出規則は，「ツイートがあるパターンにマッチすれば，属性の値はこのようになる」という形に限定している．たとえば，「ツイートの発言内容が正規表現/夕立/にマッチすれば，属性“時間帯”の値は“夕方”である」といったものである．形式的には，データ抽出規則は次の項目からなる 4 つ組み(text, cond, attr, value) である．

text : **対象テキスト** ツイートは，発言内容のほかに発言日時などの情報を持つため，次のマッチング条件をツイートのどの情報に適用するかを指定する．発言内容，発言者名，発言日時の 3 つから選択可能とする．上記例の場合，“発言内容”である．

cond : **マッチング条件** ツイート中にどのような文字列が含まれるかを表す．正規表現を利用可能である．上記例の場合“/夕立/”となる．

attr : **抽出値属性** 抽出する値を格納する属性名．上記例の場合，“時間帯”である．

value : **抽出値** 属性値として抽出する値である．例の場合，“夕方”になる．

4.3.3 ユーザインタフェース

TweetPecker のインタフェースは Simple TweetPecker と次の 2 つの点で異なる．

(1) Simple TweetPecker のインタフェースに加え，“データ抽出規則を作成する”ためのインタフェースを持つ．データフローのルール 3 によるデータ抽出規則の入力インタフェースは，前節の 4 つ組を入力するための単純な HTML フォーム(本論文では省略)である．プレイヤがデータ抽出規則を入力することを選択すると，そのフォームが表示される．

(2) 属性値の入力画面において，計算機が抽出した値を選択入力することが可能である．これは，4.2.2 節で説明した Simple TweetPecker における属性値の入力画面を拡張し，データ抽出規則の適用によって計算機が抽出された値を選択肢としてフォームに表示することによって実現する(図 3)．したがって，プレイヤは属性値を入力する際，この値を選択することが可能になる．この作業は，データフローのルール 5 に対応する．ただし，自分が作成したデータ抽出規則により抽出された値は選択することができない(単純化のため，データフローではこのためのルールは省略している)．

4.3.4 プレイヤに与えられるポイント

Full TweetPecker では，4.2.3 節で説明したポイントに加えて，次の場合にもポイントを与える．

- 決定された属性値がデータ抽出規則により抽出された値の場合，その値の真偽判定を行ったプレイヤ p_i と p_j に 3 点与える．一般に，同じ値を抽出する規則は複数存在し得るが，その場合には，その値を抽出する規則を最初に作成したプレイヤ p_k に 2 点を与える．
- 決定された属性値以外の値を抽出する抽出規則を作成したプレイヤを 1 点減点する．

5. ゲームの妥当性

本章では，TweetPecker の妥当性を議論するため，合理的なプレイヤの下でゲームが次を満たすことを示す．(1) 正しいデータと抽出規則が入力される．(2) 必ずゲームが終了する．

5.1 入力された値と抽出規則の正しさ

定理 1. データ抽出規則を入力するプレイヤを p_1，属性値を入力するプレイヤを p_2，p_3 とする．この時，ゲームの解は正しいデータ抽出規則と属性値を入力することである． □

証明. 図 4 は TweetPecker による 3 人のゲームを図示したゲーム木[9]

ID	ツイート	発言日時	発言者
73	はままつ早苗のお天気情報ー！！ 今日13日の静岡県西部のお天気は晴れでしょう。#天気 #浜松	11月13日(日) 06時08分	Hamamatsu_Sanae

属性名	決定済みの値	選択入力	直接入力
地域		◎静岡県浜松市 ◎静岡県 ◎値なし	◎その他の値
天気	晴れ	◎晴れ ◎値なし	◎その他の値
日付		◎11月13日 ◎値なし	◎その他の値
時間帯		◎値なし	◎その他の値

送信 リセット

図 3 Full TweetPecker における属性値入力フォーム
Fig. 3 Value input form of Full TweetPecker

である．ここで，終点は各プレイヤの利得を表し，点線枠内で表される p_3 に関する情報集合は，p_3 が p_2 の手を知らないことを示す．v_A は，正しい属性値であり，v_B は，誤った属性値である．また，r_A は，v_A を抽出する抽出規則であり，r_B は v_B を抽出する抽出規則である．プレイヤ p_1 は，r_A もしくは r_B を入力するとし，p_2 と p_3 はそれぞれ v_A か v_B のどちらかを入力するとする．実際には，誤った属性値は複数存在するが，1つと仮定しても一般性を損なわないため，ここでは誤った属性値として v_B のみを扱う．

p_2 と p_3 が行う部分ゲームは，4.2.3 節で説明した同調ゲームであり，同調した場合に得られる利得はどの選択肢でも同じである(図 4 の(A)と(B), (A')と(B'))．しかし，選択肢のうち，正解になりやすいものは選ばれる可能性が高いので，利得の期待値は異なる．したがって，p_2 と p_3 が行う部分ゲームの解が決まり(図 4 の(A), (A'))，さらに，p_1 は彼らが選ぶ属性値を出力するデータ抽出規則を入力することになる(図 4 の(A))． □

5.2 ゲームの終了性

定理 2. 対象となるツイート集合が有限の時，TweetPecker は必ず終了する． □

証明. ツイート集合のサイズを M とし，抽出する構造データの属性の数(4.1 節の $a_1, a_2, \ldots, a_N$)を N とする．TweetPecker は一般的な同調ゲームと異なり，プレイヤは属性値の入力と抽出規則の入力と 2 つの行動が可能であるが，ゲームのデザインにより抽出規則の入力だけでは属性値は決定しないため，ゲームの終了には少なくとも 2MN 回の属性値の入力が必要である．ここで，抽出規則は一般に無限にあるため，全てのプレイヤが永遠に抽出規則を入力し続け，ゲームが終了しない可能性がある．

TweetPecker では，4.3.4 節で説明したように，あるデータ抽出規則によって計算機が抽出した値が属性値として採用される場合，その抽出規則を作成したことによるポイントが与えられるプレイヤはただ1人であり，「利得が得られる」データ抽出規則もただ1つである．したがって，対象となるツイート集合が有限の時，「利得が得られる」データ抽出規則の集合も必ず有限であり，合理的なプレイヤはいつかは必ずルールではなく属性値を入力する．したがって，対象となるツイート集合が有限の時，TweetPecker のゲームは必ず終了する． □

6. 実験

本章では，データ抽出規則の入力を許すことが，GWAP による構造データ抽出に対してどのような影響を及ぼすかを調査する実験と，抽出したデータの品質を調査する実験の2つを行った．

6.1 実験 1: データ抽出規則の入力を許すことによる影響の調査

6.1.1 実験方法

Simple TweetPecker(以下，Simple)と Full TweetPecker(以下，Full)の 2 つを用意する．各ゲームにつき，プレイヤを 6 人用意してゲーム

を行わせる. ツイート集合と抽出する構造データのスキーマは両ゲームとも同じとする. ゲームは 60 分間行い, プレイヤの振る舞いをログ

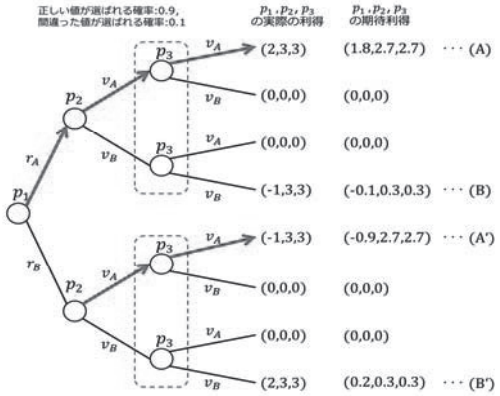

図 4　TweetPecker におけるゲーム木
Fig. 4 Game tree of TweetPecker

データに記録する. 実験終了後に, ログデータを分析し, 各ゲームの決定された属性値の数やプレイヤのポイント分布を比較する.

6.1.2 実験データ

今回の実験では, 入力であるリレーションのスキーマを Tenki(地域, 天気, 日付, 時間帯), ハッシュタグを“天気”とする. 2011 年 11 月 2 日から 2011 年 12 月 2 日の間に集めた 670 件のツイートを利用し実験を行った.

6.1.3 実験結果と考察

実験結果を表 1 および図 5〜図 7 に示す.

決定された属性値の数. 各ゲームにおいて決定された属性値の数について説明する. 表 1 は, 実験によって決定された属性値の数等をゲーム毎に比較した表である. 図 5 は決定された属性値の数の累積である. 縦軸が決定された属性値の数であり, 横軸は経過時間である. 表 1 から分かる通り, Full の方がより多くの属性値が決定された. また, 図 5 に示してあるように, 決定された属性値の数はゲーム序盤では Simple の方が多いが, 時間が経過するとともに Full の方が多くなっている. Full で序盤に決定された属性値が少ないのは, データ抽出規則の作成に労力を割くためより属性値の入力のペースが遅く, したがって属性値の決定のペースも遅いからと考えられる. その後, データ抽出規則の作成とともに, プレイヤは直接入力ではなく計算機が抽出した値を選択することが多くなるので, 入力のペースが加速する. Simple の場合よりも入力速度が速くなるのは, 選択肢から選んで入力する方が直接入力するよりも速く入力できるからである.

直接入力と計算機による抽出の割合の変化. 次に, Full において人が直接入力した値と計算機が抽出した値の割合について説明する. 図 6 は Full において決定された属性値が, 人が直接入力した値なのか計算機が抽出した値なのかの内訳を表している. ただし, 決定された属性値の内, 「値なし」であるものは除外している. 図 6 からゲーム開始直後は直接入力された値, すなわち人手によって抽出された値が多いが, 徐々に計算機によって抽出された値が多くなっていることがわかる. また, 時間の経過とともに計算機によって抽出された値が多くなるものの, 人手による抽出された値も 10%から 20%ほどあることから人手による抽出も必要であることがわかる.

ポイントの分布. 次に, 各ゲームにおいてプレイヤが得たポイントについて説明する. 図 7 は時間毎の各プレイヤのポイントの累積を表している. 横軸は経過時間である. 図 8 は各プレイヤに付与されたポイントの内訳である. 図 8 のグラフ中の各項目を説明する.

1. **直接入力による得点.** プレイヤがある属性値に対して直接入力し, その値が決定された場合に付与されるポイント.
2. **選択入力による得点.** プレイヤが選択肢の値を選んで入力し, その値が決定された場合に付与されるポイント. 選択肢の値と

表 1 決定された属性値やプレイヤが入力した属性値の数
Table 1 Numbers of inputs and decided values

項目 / ゲーム名	Simple	Full
A: 決定された属性値の数	904	1054
B: 直接入力による値の数	639	126
C: 計算機が抽出した値の数	0	818
D: 属性値が「値なし」である数	265	110
E: プレイヤの入力した属性値の数 (直接入力と選択入力の両方)	2751	2707
F: 決定された属性値の数の割合(2A / E)	66%	78%

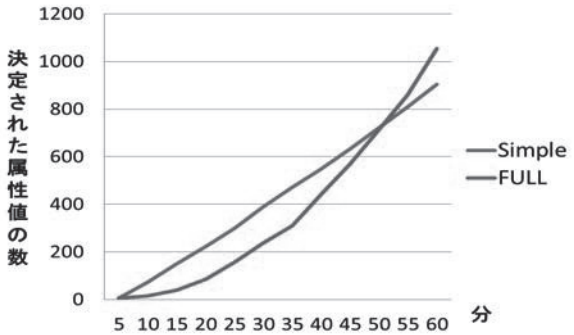

図 5 決定された属性値の数
Fig.5　Number of decided values

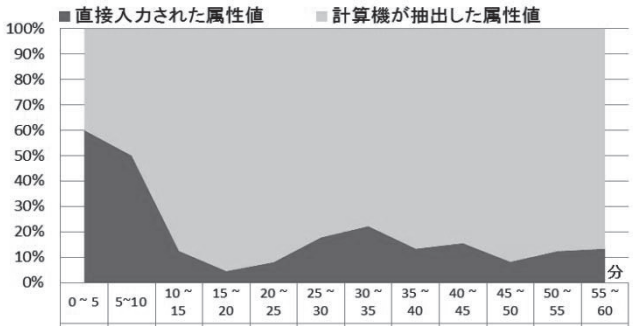

図 6. Full における決定された属性値の内訳
Fig.6 Breakdown of decided values on Full

は, 「値なし」もしくは計算機が抽出した値である.
3. **作成したデータ抽出規則による加点.** プレイヤがデータ抽出規則を作成し, その規則によって抽出された値が属性値として決定された場合に付与されるポイント.
4. **作成したデータ抽出規則による減点.** プレイヤがデータ抽出規則を作成し, その規則によって抽出された値が誤っていた場合に引かれるポイント.

図 7 からわかるように, Full では Simple に比べ, 各プレイヤのポイントの差が大きい. これは, Simple ではプレイヤは1つの属性値に対して直接入力もしくは「値なし」という2つの選択肢しかなく, プレイヤ間の入力速度に大きな違いがないからと考えられる. 図 8 から Simple ではポイントの内訳も互いに似ていることがわかる. 一方, Full ではポイントを得る手段に抽出規則を作成するという方法が加わるため, プレイヤによって行動戦略が異なった. これは, 各プレイヤのポイントの内訳がプレイヤによって違いが見られることからもわかる.

今回の実験においては, 選択肢の値を入力することによるポイント

がもっとも大きなプレイヤが存在した(p8). 実験終了後に調査したところによると, 本プレイヤは, 表示されたツイートは見ずにそれらしい選

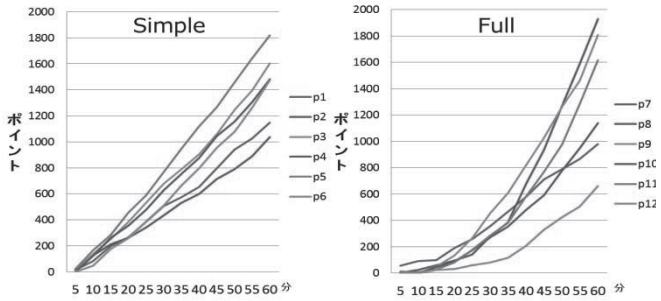

図 7. プレイヤのポイントの変化
Fig.7 Growth of player's points

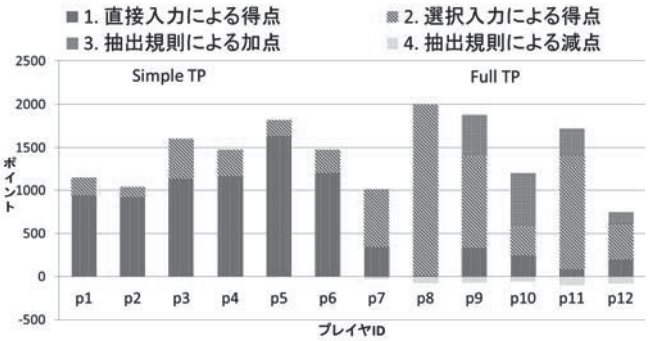

図 8. 各プレイヤのポイントの内訳
Fig.8 Breakdowns of player's points

択肢を選んでいたことがわかった. これは, 他のプレイヤはほとんど正しい値を入力することを前提として成立するフリーライダーとなっている. フリーライダーを排除するためには, 間違った選択肢を選んだ場合の減点をより大きくする必要がある.

6.2 実験 2: 抽出したデータの品質

TweetPecker で抽出したデータの品質について調査を行った. TweetPecker で抽出された属性値の内, 正誤の判定が可能なものを正しい値と誤った値に分類した. 地域属性の値はおよそ 99%の値が正しい値であり, 天気属性は 98%, 日付属性は 97%, 時間帯属性は 62%であった. 時間帯属性に誤りが多かった理由は, 今回の実験で使用したハッシュタグ「#天気」によって取得したツイートではその時点の天気に関するツイートと天気予報に関するツイートが存在しており, 天気予報に関するツイートでは, 時間帯が明記されていないものが多々存在したことである. このとき, プレイヤはツイートされた時刻を入力する傾向があった. このように人手でも困難なデータ抽出の場合には, 間違った値を入力するのを防ぎ, かつシステムがその作業が困難であることを発見する仕組みが必要と考えられる.

7. まとめと今後の課題

本論文では, 非構造データから構造データを抽出する方法として GWAP を用いる手法を提案した. 提案手法を実現したシステム TweetPecker は, Twitter 上の発言からリレーションを抽出するシステムである. TweetPecker には事前にリレーションのスキーマを登録し, 次に抽出する各タプルの属性値を複数プレイヤによる同調ゲームによって決定する. TweetPecker の特徴は, プレイヤに対して属性値の入力をさせるだけでなく, データ抽出規則の入力を許すことである. このため, 最初は人手によって属性値が抽出されるが, 抽出規則が増えると属性値の抽出が徐々に計算機によって行われるようになる. 本論文ではこのゲームが合理的なプレイヤの元で正しく動作し, かつ必ず終了することを示した. また, 実験により, プレイヤにデータ抽出規則の入力を許可した場合の振る舞いを検証し, ゲームの進行につれて計算機による抽出の割合が増加する事を確認した.

今後の課題としては, 1 ツイートにつき複数のタプルを生成することを可能にするなど, 提案手法が扱える対象をより拡大する事である.

[謝辞]

本研究の一部は JST さきがけ「情報環境と人」および科学研究費補助金(#21240005)による.

[文献]

[1] Wei Han, David Buttler, Calton Pu. "Wrapping web data into XML". Newsletter ACM SIGMOD Record. 2001, 30(3), pp.33-38.

[2] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran Soares da Silva, Juliana S. Teixeira: A Brief Survey of Web Data Extraction Tools. SIGMOD Record 31(2): 84-93 (2002)

[3] Luis von Ahn, Laura Dabbish. "Designing Games With A Purpose".Communication of the ACM. 2008, 51(8), pp.58-67.

[4] Luis von Ahn , Laura Dabbish, "Labeling Images with a Computer Game". Proceedings of the SIGCHI conference on Human factors in computing systems. 2004, pp.319-326.

[5] Hector Garcia-Molina, Jeffrey D Ullman, Jennifer Wisdom. "Database Systems: the Complete Book". Prentice Hall. 2002, pp.463-502.

[6] Luis von Ahn, Mihir Kedia, Manuel Blum: Verbosity: a game for collecting common-sense facts. CHI 2006: 75-78

[7] 白川 真澄, 中山 浩太郎, 荒牧 英治, 原 隆浩, 西尾 章治郎. "Wikipedia と Web の情報を組み合わせたオントロジー構築の試み". 電子情報通信学会論文誌. D, 情報・システム. 2011, J94-D(3), pp.525-539.

[8] Atsuyuki Morishima, Norihide Shinagawa, Shoji Mochizuki. ``The Power of Integrated Abstraction for Data-centric Human/Machine Computations". VLDS 2011. 2011, pp.5-9.

[9] F. Vega-Redondo. Economics and Theory of Games. Cambridge University Press, 2003.

福角 駿 Shun FUKUSUMI
筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻在学中. 日本データベース学会学生会員.

森嶋 厚行 Atsuyuki MORISHIMA
筑波大学図書館情報メディア系/知的コミュニティ基盤研究センター准教授. 1998 年筑波大学大学院工学研究科修了. 博士(工学). ACM, IEEE-CS, 情報処理学会, 電子情報通信学会, 日本データベース学会各正会員.

品川 徳秀 Norihide SHINAGAWA
筑波大学図書館情報メディア系/知的コミュニティ基盤研究センター主任研究員. 筑波大学大学院工学研究科修了. 博士(工学). ACM, IEEE-CS, 情報処理学会, 日本データベース学会各正会員.

北川 博之 Hiroyuki KITAGAWA
筑波大学システム情報系教授. 1980 年東京大学大学院理学系研究科修了. 理学博士. 情報処理学会フェロー, 電子情報通信学会フェロー. ACM, IEEE-CS, 日本データベース学会各正会員.