3F2-05

木の編集距離を用いた半構造データからの情報抽出

Information Extraction from Semistructured Data Using Tree Edit Distance

久保山 哲二[*1] 宮原 哲浩[*2]
Tetsuji Kuboyama Tetsuhiro Miyahara

[*1]東京大学 国際・産学共同研究センター
Center for Collaborative Research, University of Tokyo

[*2]広島市立大学 情報科学部
Faculty of Information Sciences, Hiroshima City University

Recent research efforts on extracting information from Web pages have mainly focused on semi-automatic and automatic approaches to generating wrappers. In this paper, we present a structure-based approach to finding a common structured pattern from semistructured data such as HTML documents and XML documents through approximate tree matching, as defined by a tree edit distance metric. The common strucured pattern is generated by merging the set of trees according to a frequency measurement of nodes which appear in the tree structures of nested HTML tags. We investigate how to extract information by using the pattern.

1. はじめに

大量の Web ページから有用な情報を自動抽出する技術は、Web 上に蓄積されるデータが爆発的に増加しつづけている状況で、非常に重要性を増している。一方、Web 上で閲覧可能なデータには、シラバスやオンラインショップの商品などのように、同じ HTML のテンプレートをもとに記述された Web ページ群が多数存在する。このようなデータは一般的に構造化されているため自動化が容易である。しかし、テンプレートから手書きで作成された Web ページ群では、もとのテンプレートの構造を見出すことは容易ではない。

このような記述の揺れを吸収するために、本稿では、木の編集距離による近似マッチングを利用した Web ページ群からの共通パタン抽出手法を提案する。木の編集距離は、文字列の編集距離の概念を木構造データに拡張したものであり、10 年来、計算生物学、自然言語解析、画像解析をはじめとするさまざまな分野で研究が進められている [Bille 03]。木の編集距離計算により、2 つの木のノード間の近似的な対応が得られるため、不規則な部分構造を含んだ木構造間のマッチングに適している。本手法では、HTML データを構文解析した解析木 (タグ木構造) として扱い、木の編集距離で得られるノードの対応情報に基づき、ノードに重み付けを行いながら、複数のタグ木構造を結合してゆく。最終的に得られたタグ木構造と変数の重み付けから、タグ木構造の一部を変数化することにより、情報抽出のための木構造パタンを生成する。

同様の半構造データからの構造的特徴の発見手法としては、与えられた順序木集合に対する極小言語問題を解く MINL アルゴリズムに基づくタグ木パタン (tag tree pattern) の発見手法が提案されている [Miyahara 03]。タグ木パタンは、変数を持つ根付き順序木であり、変数には、木構造を埋め込むことができるため、半構造データを特徴付けるのに適した木構造パタンである。タグ木パタンと順序木のマッチングについては、効率的な判定アルゴリズムが知られている [Suzuki 02]。

以下、2 節では本手法の基礎となる木の編集距離について述べる。3 章で木の編集距離を用いた木構造パタンの生成手法を与え、4 節で情報抽出の計算についてのべる。最後に 5 節でまとめを行う。

連絡先: [*1]kuboyama@ccr.u-tokyo.ac.jp,
[*2]miyahara@its.hiroshima-cu.ac.jp

2. 木の編集距離

Zhang らによる木の編集距離の計算方法 [Shasha 89] をもとに、新たに定式化を行う。

定義 1. **木**(tree) は根ノードと互いに素な木の順序列から構成される。このような木の順序列を**森**(forest) という。木の列 $T_1, \ldots, T_n$ から構成される森を $F = (T_1 \circ (T_2 \circ (T_3 \circ \cdots \circ (T_n))))$ と表記し、根ノード v と森 F から構成される木を $v(F)$ と表記する。

ここで、木は順序木であり、ノードはラベル付けされているものとする。木はただ 1 つの要素からなる森であるとし、空の森は $\emptyset$ で表す。また、森 F と G を順に並べて得られる森を $(F \circ G)$ と略記する。

定義 2. F と G をそれぞれ森とする。**森 F と G の編集距離**(edit distance) は、F を G に変換するために必要な編集操作に要するコストの最小値であり、$\mathrm{FD}(F, G)$ と表記する。

編集距離の計算をする際には、通常次の 3 つの編集操作を用いる。(1) **代入**(substitution):ノード v のラベルを別のノード w のラベルに置き換える $(v \to w)$。(2) **挿入**(insertion):ノード w を挿入する $(\lambda \to w)$。(3) **削除**(deletion):ノード v を削除する $(v \to \lambda)$。また、それぞれの操作のコストを、$C_{\mathrm{sub}}(v, w) \equiv \gamma(v \to w)$, $C_{\mathrm{ins}}(w) \equiv \gamma(\lambda \to w)$, $C_{\mathrm{del}}(v) \equiv \gamma(v \to \lambda)$ と表記する。

編集操作の列 $S = s_1, \ldots, s_n$ が与えられたとき、その全コストは $\gamma(S) = \sum_{i=1}^{n} \gamma(s_i)$ とする。よって、木 T_1 と T_2 の編集距離 $\delta(T_1, T_2)$ は次のように定義できる。

$$\delta(T_1, T_2) = \min\{\gamma(S) | S \text{ は } T_1 \text{を } T_2 \text{に変換する編集操作列 }\}$$

定義 3. 木 T_1 と T_2 間の**編集マッピング**(edit mapping) は、次の条件を満たす $M(T_1, T_2)$ である。$V(T_1), V(T_2)$ を、それぞれ木 T_1, T_2 に含まれるノードの集合とすると、$M(T_1, T_2) \subseteq V(T_1) \times V(T_2)$ と、任意の $(v_1, w_1), (v_2, w_2) \in M(T_1, T_2)$ について、下記の条件を満たす。

1. $v_1 = v_2 \Leftrightarrow w_1 = w_2$
2. v_1 は v_2 の祖先 $\Leftrightarrow$ w_1 は w_2 の祖先
3. v_1 は v_2 の左側 $\Leftrightarrow$ w_1 は w_2 の左側

なお、文脈から明らかな場合は、編集マッピングを単にマッピングと呼び、M と表記する。

マッピング $M(T_1, T_2)$ において、$N_1 = V(T_1) \setminus \{v|\exists w(v,w) \in M\}$, $N_2 = V(T_2) \setminus \{w|\exists v(v,w) \in M\}$ とする。このとき、編集マッピングに次のような編集操作を対応づける。M は、木 T_1, T_2 間でラベルの書き換えを行うノードの対の集合 (同一のラベルに書き換えるものも含む)、N_1 は木 T_1 から削除されるノードの集合、N_2 は木 T_2 に挿入されるノードの集合をあらわす。すると、編集マッピング $M(T_1, T_2)$ から、木 T_1 を T_2 に書き換えるための編集コストが下記のように与えられる。

$$\gamma(M) = \sum_{(v,w)\in M} \gamma(v \to w) + \sum_{v \in N_1} \gamma(v \to \lambda) + \sum_{w \in N_2} \gamma(\lambda \to w)$$

編集マッピングにより、編集操作の適用順序の詳細に立ち入ることなく、木 $T_1, T2$ 間のノードの対応として編集距離を定義することができる。

$$\delta(T_1, T_2) = \min\{\gamma(M) | M \text{ は } T_1 \text{を } T_2 \text{に変換するマッピング }\}$$

木の編集距離の計算のため、動的計画法などを用いたさまざまなアルゴリズムが提案されている [Bille 03]。本稿でも、動的計画法による定式化を行う (F_1, F_2, G_1, G_2 は各々値として $\emptyset$ をとりうることに注意)。

アルゴリズム 木の編集距離計算

TreeEditDistance(T_1, T_2)
入力として与えられた木 T_1, T_2 に対して
関数 FD により森 $F = (T_1)$, G=(T_2) 間の距離を求める
return FD(F, G)

- $F = \emptyset$, $G = \emptyset$ のとき
  FD($\emptyset$, $\emptyset$) $= 0$
- $F = \emptyset$, $G = (w(G_1) \circ G_2)$ のとき
  FD($\emptyset$, $(w(G_1) \circ G_2)$) $=$ FD($\emptyset$, $(G_1 \circ G_2)$) $+ C_{\text{ins}}(w)$
- $G = \emptyset$, $F = (v(F_1) \circ F_2)$ のとき
  FD($(v(F_1) \circ F_2)$, $\emptyset$) $=$ FD($(F_1 \circ F_2)$, $\emptyset$) $+ C_{\text{del}}(v)$
- $F = (v(F_1) \circ F_2)$, $G = (v(G_l) \circ G_2)$ のとき
  FD($(F_1 \circ v(F_2))$, $\emptyset$)=

$$\min \left\{ \begin{array}{lcl} \text{FD}((F_1 \circ F_2), (w(G_1) \circ G_2)) & + & C_{\text{del}}(v), \\ \text{FD}((v(F_1) \circ F_2), (G_1 \circ G_2)) & + & C_{\text{ins}}(w), \\ \text{FD}(F_1, G_1) + \text{FD}(F_2, G_2) & + & C_{\text{sub}}(v, w) \end{array} \right\}$$

なお、上記の木の編集距離計算アルゴリズムと同様に木構造のトレースバックを行うことで 2 つの木の間のマッピングが得られる。

木の編集距離は、木 T_1 と T_2 について、下記の時間で計算できる [Zhang 94]。ここで、木 T のノードの総数を $|T|$、深さを depth(T)、葉ノードの総数を leaves(T) と表記する。

$$O(|T_1| \times |T_2| \times \begin{array}{l} \min(\text{depth}(T_1), \text{leaves}(T_1)) \times \\ \min(\text{depth}(T_2), \text{leaves}(T_2))) \end{array}$$

HTML や XML の構文木のでは、ノードにタグとデータをラベル付けする (それぞれ、タグノードとデータノードと呼ぶ)。編集距離の計算において、ノードがタグ同士のときは、一致するか否かでコストを計算し、データ同士の場合は、文字列の編集距離を用いてコストを計算する。

3. 木構造パタンの生成

本節では、木の編集距離のコスト計算に対する MDL 基準の適用研究 [Torsello 03] に基づいた木構造パタンの生成手法を示す。これは、木のサンプル集合 $D = \{T_1, T_2, \ldots, T_n\}$ から、木のパタン空間中における生成モデルを得るものであり、共通の構造を持つ部分のマッピングコストは低く、特殊な構造を持つ 部分のマッピングコストは高くなるようにノードの重み付けを計算する。

$T \in D$ の各ノード i に重み w_i があたえられた木構造パタン (基底木構造パタン) を $\mathcal{T}$ とする。このとき、ノード i と j のマッピングコストを $C_{\text{sub}}(i,j) = |w_i - w_j|$ とする。同様に、$C_{\text{del}}(i) = w_i$、$C_{\text{ins}}(j) = w_j$ とする。また、マッピング $M(\mathcal{T}_1, \mathcal{T}_2)$ について、$N_1 = V(\mathcal{T}_1) \setminus \{i|\exists j(i,j) \in M\}$, $N_2 = V(\mathcal{T}_2) \setminus \{j|\exists i(i,j) \in M\}$ とおき、木の編集距離を次のように与える。

$$\begin{aligned} \delta(\mathcal{T}_1, \mathcal{T}_2) &= \sum_{i \in N_1} w_i + \sum_{j \in N_2} w_j + \sum_{(i,j) \in M} |w_i - w_j| \\ &= \sum_{i \in V(\mathcal{T}_1)} w_i + \sum_{j \in V(\mathcal{T}_2)} w_j - 2 \sum_{(i,j) \in M} \min(w_i, w_j) \end{aligned}$$

各ノード $i \in V(\mathcal{T})$ の重み w_i をデータ D 中におけるノード i の出現確率とする。

$$w_i = \frac{p_i(\mathcal{M})}{|D|}$$

ここで、木 $T \in D$ について、$\mathcal{M}(\mathcal{T}, T)$ は $\mathcal{T}$ と T のマッピング、$p_i(\mathcal{M})$ はノード $i \in V(\mathcal{T})$ と対応するノード $j \in V(T)$ が $\mathcal{M}$ に存在するような木 $T \in D$ の数を表す。

この定義に従い、基底木構造パタンを次の手順で求める。

1. 木 $T \in D$ の全てのノード i の重みを $w_i = 1$ と置き、基底木構造パタンとする。
2. 全ての木 $T \in D$ 間の編集距離を求め、距離の近いものから順に統合を行い、木構造パタンを生成してゆく。
3. マッピングがとれたノードは統合し、挿入、削除したノードは、マッピングしたノードに付加することにより木構造パタンを生成する。この際、各ノードの重みを次のようにして更新する。
   - ノード i がマッピングにない場合
     $$w_i = \frac{p_i}{m_1 + m_2}$$
   - ノード j がマッピングにない場合
     $$w_j = \frac{p_j}{m_1 + m_2}$$
   - (i,j) がマッピングにある場合
     $$w_{ij} = \frac{p_i + p_j}{m_1 + m_2}$$

   ここで、m_1, m_2 は、それぞれ木構造パタン $\mathcal{T}_1$ と $\mathcal{T}_2$ の生成に用いられた木の数、p_i, p_j は、$\mathcal{T}_1$ と $\mathcal{T}_2$ 各々の生成に用いられた D 中の木の中で、ノード i, j とマッピングがとれる木の数である。

2 つの木を統合する方法としては、アラインメントが考えられる。木の編集距離においては、文字列の編集距離のように、マッピングから直ちにアラインメントを得ることができない。このため、木の場合は、木のアラインメント [Jiang 03] がよく知られている。しかし、木のアラインメントによって木を統合すると、次のような問題が発生する。すなわち、木のアラインメントにおいては、2 つの木の上方の構造が一致しなければ、下方の構造が一致することはない。この性質のため、木のアラインメントによる統合を行うと、下部に同じ部分構造を重複して生成してしまう可能性が高い。つまり、木の編集距離では対応のとれていたノードが、別々のノードとして扱われてしまう。

一方、木の編集距離に基づくマッピングを用いてナイーブに木を統合すると、複数の親ノードを持つノードが発生し木構造を保てなくなる (非循環有向グラフになる)。そこで、本手法では、2 節で示した木の編集距離よりも、構造を保つ制約を加えた**低制約編集距離**(less-constrained editdistance)[Lu 01] から得られる編集マッピングを用いて木を統合する (図 1)。

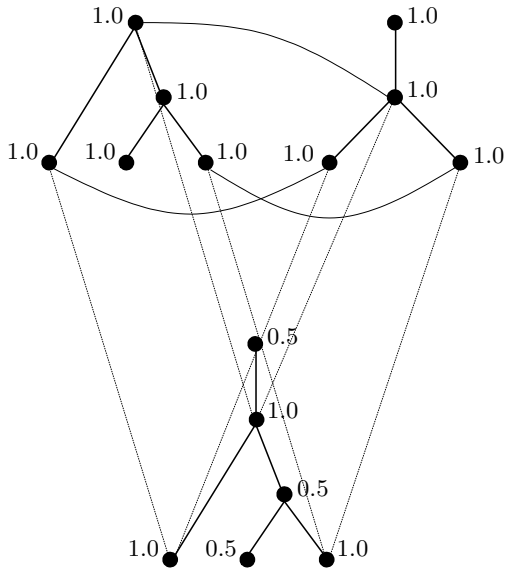

図 1 木の統合例

4. 木構造パタンの生成

前節で得られた変数を含まない基底木構造パタンを、葉ノードから順に重み付けに応じて変数を含む木構造パタンに変換する。

ここではその概要を述べる。

1. まず、重み付けの閾値を定める (たとえば、$\theta = 0.7$)。
2. 閾値に満たない基底木構造パタンのノードに印をつける。(このノードを弱ノードと呼ぶ。)
3. 葉ノードの兄弟間に連続した弱ノードがある場合、1 ノードに束ね、anchor-VLDC にする (次節参照)。
4. 兄弟のない葉ノードで、親ノードとともに弱ノードのとき、
   (a) その葉ノードが anchor-VLDC なら親ノードとともに、1 つのノードに束ね anchor-VLDC にする。
   (b) それ以外では、親ノードとともに 1 つの path-VLDC にする。
5. (3) と (4) を新たにノードを束ねることができなくなるまで繰り返す。
6. 上記以外で、隣接するノードに弱ノードがない場合、そのノードをラベル変数とする。

5. 情報抽出

本節では、前節で生成した木構造パタンからの情報抽出について述べる。

ある形状の部分木のマッチングコストを無視するために、Zhang らにより 2 種類の VLDC、path-VLDC(図 3) と umbrella-VLDC(図 4) をノードのラベルに含む木を用いたマッチング手法が提案されている [Zhang 94]。本稿では、VLDC を変数とみなし、前節で得られた木構造パタン中の変数にマッチングした部分の情報を抽出することがねらいである。そのために、Zhang らの VLDC に加えて部分木を埋め込むことができる新たな変数 (△) を導入し (図 2)、変数を含まない順序木とのマッピングと情報抽出の計算方法を示す。ここでは、情報抽出を、変数に代入される木構造の同定とする。任意のノード v について、$\gamma(\triangle \to v) = 0$ とする。

- 根のラベルが △ の場合:

$$\mathrm{FD}(\triangle(F), v(G)) = \min \left\{ \begin{array}{ll} \mathrm{FD}(F, v(G)) & + \ \gamma(\triangle \to \lambda), \\ \mathrm{FD}(\triangle(F), G) & + \ \gamma(\lambda \to v), \\ \mathrm{FD}(F, G) & + \ \gamma(\triangle \to v), \\ \mathrm{FD}(\emptyset, G) & + \\ \quad \min_{1 \le k \le n} \{\mathrm{FD}(\triangle(F), T_k) - \mathrm{FD}(\emptyset, T_k)\}, & \\ \min_{1 \le k \le n} \{\mathrm{FD}(F, G - T_k)\} & \end{array} \right\}$$

ここで、$G = (T_1 \circ \cdots \circ T_n)$ とする
また、$F - T$ は、森 F から木 T を削除した森を表す

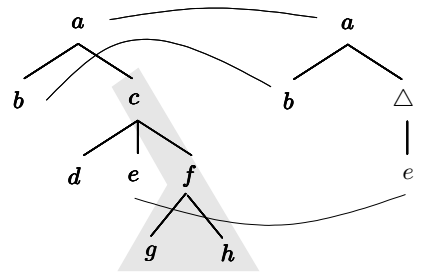

図 2 マッチングの例 (anchor-VLDC)

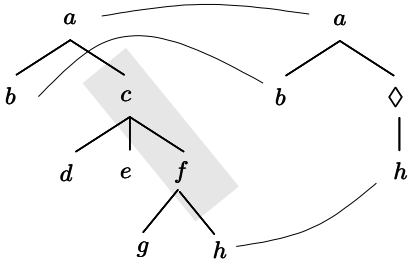

図 3 マッチングの例 (path-VLDC)

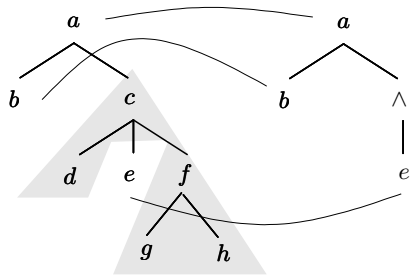

図 4 マッチングの例 (umbrella-VLDC)

抽出情報は、トレースバックによるマッピング計算の過程で変数にマッチした部分の構造を取り出せばよい (他の変数の場合も同様に計算できる)。

6. まとめ

一定の共通構造を持つ Web ページ群から情報抽出のためのパタンを生成し、このパタンを用いて情報抽出を行う手法を示した。本手法では木の編集距離を用いた木構造の近似マッチングに基づき、木構造の確率分布を考慮したパタンを生成するため、Web ページの記述の揺れにへの耐性が期待できる。また、パタン中の変数は単体のデータのみでなく、データの部分構造を抽出できるため、共通構造に埋め込まれた変則的な構造も自然に抽出することが可能である。

今後、HTML のタグ知識などを用いたより緻密なデータレコード抽出を組み込んだ実装を行い、シラバスや商品仕様などの Web ページ群に対して本手法を適用し、検証を行う。

参考文献

[Miyahara 03] Tetsuhiro Miyahara, Yusuke Suzuki, Takayoshi Shoudai, Tomoyuki Uchida, Sachio Hirokawa, Kenichi Takahashi, Hiroaki Ueda, “Extraction of Tag Tree Patterns with Contractible Variables from Irregular Semistructured Data,” PAKDD 2003, Springer LNAI 2637, pp. 430-436.

[Suzuki 02] Yusuke Suzuki, Kohtaro Inomae, Takayoshi Shoudai, Tetsuhiro Miyahara, Tomoyuki Uchida, “A Polynomial Time Matching Algorithm of Structured Ordered Tree Patterns for Data Mining from Semistructured Data,” ILP 2002, Springer LNAI 2583, pp. 270-284.

[Bille 03] Philip Bille, “Tree Edit Distance, Alignment Distance and Inclusion,” Technical report TR-2003-23, IT University of Copenhagen, March 2003.

[Shasha 89] Kaizhong Zhang, Dennis Shasha, “Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems,” SIAM J. Comput. 18(6), 1989, pp. 1245-1262.

[Zhang 94] Kaizhong Zhang, Dennis Shasha, Jason T. L. Wang, “Approximate Tree Matching in the Presence of Variable Length Don’t Cares,” Journal of Algorithms, Vol. 16, No. 1, January 1994, pp. 33-66.

[Torsello 03] Andrea Torsello, Edwin R. Hancock, “Tree Edit Distance from Information Theory,” IAPR Workshop GbRPR 2003, LNCS 2726, pp.71-82.

[Jiang 03] Tao Jiang, Lusheng Wang, Kaizhong Zhang, “Alignment of trees – an alternative to tree edit,” Theoretical Computer Science(TCS), 143, 1995, pp.137-148.

[Lu 01] Chin Lung Lu, Zheng-Yao Su, Chuan Yi Tang, “A New Measure of Edit Distance between Labeled Trees,” COCOON 2001, LNCS 2108, 2001, pp.338-348.