2F3-04

木の近似照合に基づく共通構造の発見

Finding a Common Structure based on Approximate Tree Matching

久保山 哲二[*1] 申 吉浩[*2] 宮原 哲浩[*3]
Tetsuji Kuboyama Kilho Shin 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]東京大学 先端技術研究センター
Research Center for Advanced Science and Technology, University of Tokyo

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

Information extraction from semistructured data such as HTML documents gains importance with the growth of the Web. This paper proposes an efficient method for discovering a common tree structure from multiple trees based on the bottom-up subtree isomorphism algorithm by Valiente. The aim of this work is to extract Web contents from a set of HTML documents which are generated from a common template, but include a number of grammatical mistakes in HTML, redundant or missing fragments introduced by manual editing.

1. はじめに

インターネット上に蓄積されている膨大な HTML や HTML 文書等の情報を系統的に抽出し、利用する技術の開発は、大量の情報資源をデータベースとして有効利用するために必須である。Web 上で閲覧可能なデータには、オンラインショップの商品リストなどのように、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にプログラムにより自動生成されているために、テンプレートの抽出は比較的に容易である [1]。しかし、大学のシラバスデータ等にみられるように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は、効率のよい木の近似照合手法と共通構造の発見手法が求められている。

本稿では、木の近似照合に編集距離による手法を用いる。木の編集距離は、文字列の編集距離の自然な拡張であるが、どのような木構造の類似性に着目するかに応じて、様々な照合のクラスがある [2, 3, 4]。Zhang らによって提案された一般的な木の編集距離手法 [5, 6] が、もっとも一般的なクラスである。また、Jiang らにより提案された木のアラインメント [7] および、Lu らにより提案された less-constrained 編集距離 [8] は、Zhang [5, 6] による方法の部分クラスであり、2 つの木の結合を可能にする最も一般的なクラスであることが、久保山らにより示されている [4]。また、Zhang らにより提案された制約マッピングによる編集距離 [9] を用いた近似照合手法は、木のアラインメントのクラスに真に含まれることが知られている。

本研究では、XML や HTML 文書等の半構造データを一般的な木構造グラフとしてとらえる。そして、情報抽出に必要となる木構造グラフ上の問題を設定し、これを解決するための効率よいアルゴリズムを提案する。本稿では、とくに、次のような問題について考える。

- 木構造中に現れる部分木の繰り返し構造を発見する問題
- 3 つ以上の木構造について近似的に対応するノードを発見する問題 (木構造に対するマルチプルアラインメントの部分問題)

この 2 つの問題を解決するために、Valiente [10] により提案されている 2 つの木構造間の同型な部分木をボトムアップに求めるアルゴリズムを利用する。まず、2 つの木を対象にした Valiente のアルゴリズムを拡張して、複数の木に共通して存在する部分木をボトムアップに求める。これによって、複数の木構造間におけるノードの対応と、1 つの木の中に繰り返し現れる部分木を求めることができる。さらに、ノード間の対応だけでなく、対応する部分木を近似的に求める手法を与える。これらのアルゴリズムは、順序木と無順序木のいずれにも適用可能である。

以下、2 節で Valiente による 2 つの木構造間のボトムアップマッピングについて述べ、3 節で、複数の木に対する拡張、および上層構造のマッピングを求める問題への拡張を行い、それぞれアルゴリズムを示す。最後に 4 節でまとめを行う。

2. 準備

2.1 木

本稿では、木を半順序集合上の構造として表現する。**厳密半順序**(strict partial order) を表すために、標準的な表記 $<$ を用いる。すなわち、非空有限集合 V について次の条件を満たすものとする。

1. $\forall x, y, z \in V\ [x < y \ \wedge\ y < z \Rightarrow x < z]$ (推移性),
2. $\forall x \in V\ [x \not< x]$ (非対称性).

また、任意の $x, y \in V$ について、表記 $x \leq y$ により $x < y$ または $x = y$ であることを表す。

定義 1 (根つき木)**. 根つき木** $T = (V, <)$ は、**根**という最大要素 $r(T) \in V$ をもつ非空の有限厳密半順序集合で、かつ $\{y \in V | x \leq y\}$ が任意の $x \in V$ について、全順序集合となるものである。

以降、根つき木を、単に木という。集合 V の要素を T の**ノード**と呼び、T のすべてのノードからなる集合を $V(T)$ で

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

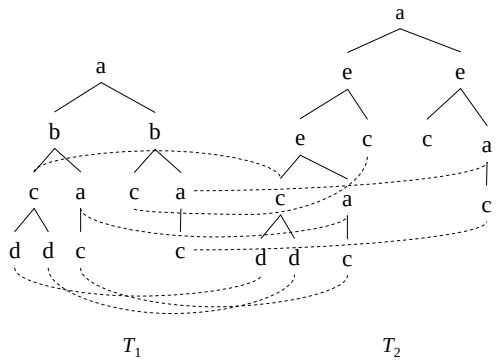

図 1: 例: 木 T_1 と T_2 のボトムアップマッピング

表す。木 T の**辺集合**を $E(T) = \{(x, y) \in V(T) \times V(T) | x < y \wedge \nexists z \in V(T)\ [x < z < y]\}$ により定義し、辺集合の要素を辺という。ノード x の**祖先**は、$x \leq y$ となるようなノード y である。とくに、$x < y$ のとき、y を**真の祖先**という。ノード x の**親**とは、x の真の祖先のなかで最小のノードであり、$p(x)$ で表す。ノード x の **子集合**は、集合 $\{y|(y, x) \in E(T)\}$ であり、$\mathrm{ch}(x)$ で表す。子集合 $\mathrm{ch}(x)$ の要素を**子**という。また、T の極小ノードを**葉**という。木 T に含まれるノード x について、$y \leq x$ となるようなすべてのノード y を含む部分木を $T(x)$ と表記する。木 T に含まれるノードの数を、木の大きさと定義し、$|T|$ で表す。2 項関係 M について、$M|_1 = \{x|(x, y) \in M\}$、$M|_2 = \{y|(x, y) \in M\}$ とする。

定義 2. 木 $T = (V, <)$ と集合 $V' \subseteq V$ が与えられたとき、任意のノード $y \in V'$ について $y \leq x$ となるようなノード $x \in V$ を V' の**共通祖先**(common ancestor) という。V' の共通祖先 x が、任意の V' の共通祖先の中で最小であるとき、**最小共通祖先**(least common ancestor) という。

$\mathrm{lca}(V')$ により、V' の最小共通祖先、$x \smile y$ により、$\mathrm{lca}(\{x, y\})$ をそれぞれ表す。

2.2 木の近似照合とマッピング

2 つの木の間での近似照合の表現に、**マッピング**(tree mapping) という概念を用いる。木マッピングとは、2 つの木の間のノード間の対応関係を表す集合であり、次のように定義する [4]。

定義 3 (マッピング)**.** 木 T_1 から T_2 への**マッピング** M とは、$V(T_1) \times V(T_2)$ の部分集合であり、次の条件を満たす集合である。

$\forall (x_1, x_2), (y_1, y_2) \in M\ [x_1 \leq y_1 \Leftrightarrow x_2 \leq y_2]$

この定義は、直感的には、ノードの対応が一対一であること、および、ノードの上下関係が保存されることを示している。順序木の場合は、ノードの左右関係の保存条件が加わる。

次の 制約マッピングは Zhang により導入された。

定義 4 (Zhang [6])**.** マッピング M は、次の条件を満たすとき **制約マッピング**(constrained mapping) であるという。

$\forall (x_1, x_2), (y_1, y_2), (z_1, z_2) \in M$
$[z_1 < x_1 \smile y_1 \Leftrightarrow z_2 < x_2 \smile y_2]$

制約マッピングの特徴は、マッピングに含まれるノードの対応だけでなく、ノードを含む部分木がどのように 2 つの木の間で対応が取れているのかを考慮しており、対応がとれる部分木同士は、互いに素であることを保証している。

定理 1 (久保山ら [4])**.** マッピング M が、次の条件を満たすとき、かつそのときに限り、木の結合が存在する。

$\forall (x_1, x_2), (y_1, y_2), (z_1, z_2) \in M$
$[x_1 \smile y_1 < x_1 \smile z_1 \Rightarrow y_2 \smile z_2 = x_2 \smile z_2]$.

すなわち、この条件を満たすマッピングがアラインメントである。

アラインメントは、低制約 (less-constrained) マッピングともよばれる。次にボトムアップマッピングの定義を示す。

定義 5 (Valiente [10])**.** マッピング M は、次の条件を満たすとき**ボトムアップ**であるという。

$\forall (x_1, x_2) \in M\ [|T(x_1)| = |T(x_2)| \wedge$
$y_1 \in V(T_1(x_1)) \Rightarrow y_2 \in V(T_2(x_2))$ s.t. $(y_1, y_2) \in M]$

この定義は、Valiente による定義 [10] と表現は違うが等価である。なお、Valiente [10] がボトムアップマッピングをアラインメント [7] の部分クラスとしているのは間違いである。詳しくは、久保山らによる編集距離に基づく木の近似照合の意味論に関する厳密な議論 [4] を参照のこと。図 1 に、ボトムアップマッピングの例を示す。図において、破線部分がマッピングを表している。

ボトムアップマッピング問題とは、2 つの木構造間で、最大の要素含むボトムアップマッピングを求める問題である。

3. 複数の木構造間のボトムアップマッピング

本節では、このアルゴリズムを複数の木構造間のボトムアップマッピング問題を解くアルゴリズムに拡張する。次に、ボトムアップマッピングを拡張して、明示的には対応がとれていなかったノードについても近似的に照合を行うアルゴリズムを示す。

3.1 複数の木の簡約表現

複数の木構造間のボトムアップマッピングを求めるために、まず、複数の木を 1 つの森とみなして、ボトムアップに同型な部分木をまとめあげた非巡回グラフ (DAG) 表現を作る。

定義 6. 森 $F = \{T_1, \ldots, T_n\}$ の**簡約**表現とは、次の条件を満たす非巡回グラフ G のことである。

1. G 中の各ノード v は、森 F のノードの同値類 $[v]$ に対応する。ここで、ノード u と v が等しいとき、かつそのときに限り、u を根とする F の部分木と、v を根とする F の部分木が同型であると定義する。
2. G 中にノード u から v への有向辺があるとき、かつそのときに限り、ノード $x \in [u]$ を含む木 T が F 中に存在し、T において、$y \in [v]$ となる x の子ノード y が存在する。

図 2 に、F の簡約表現を求めるアルゴリズムを示す。また、図 3 に、F の簡約表現の例を示す。このアルゴリズムの時間計算量は、F 中に含まれる全ノードの数を $|F|$ と表記すると、$O(|F|)$ となる。

このようにして F の簡約表現の DAG を求めることにより、F 中に現れる共通の部分木を効率よく見つけることができる。

3.2 ボトムアップマッピング問題

次に、F の DAG から、実際にボトムアップマッピングを求めるアルゴリズムを、図 4 に示す。一般に、F のボトムアップマッピングは F の DAG から一意に決まらない。とくに、順序木の場合は、左側で最初に照合する部分木から、マッピングの対象にしてゆくことにより、マッピング条件を満たす結果が得られる。

アルゴリズム 木 $F = \{T_1, \ldots, T_n\}$ の DAG 記述

```
procedure compact(F: trees, G: DAG, K: mapping)
  (K is a mapping from nodes in F to nodes in G)
  let L be an empty hash of node labels to nodes of G
  foreach leaf label l in F do
    add a new node v to G
    label[v] ← l; L[l] = v
  end foreach
  let Q be a queue containing all leaf nodes in F
  foreach node v in F do
    rest[v]← number of children of v
  end foreach
  while Q is empty do
    dequeue node v from Q
    if v is leaf then K[v] ← L[label[v]]
    else
      foreach node w in G from the latest added do
        if v and w have the same
                    label, height, and number of children
          and
          {K[u]|u is a child of node v} =
                    {u|u is a child of node w}
        then K[v] ← w
        else
          add a new node w to G
          K[v] ← w; label[w] ← label[v]
        end if
      end foreach
    end if
    if node v is not the root of a tree in F then
      rest[parent[v]]--
      enqueue parent[v] into Q if rest[parent[v]] = 0
    end if
  end while
```

図 2: 木 $F = \{T_1, \ldots, T_n\}$ の DAG 記述を求めるアルゴリズム

3.3 上層構造マッピングへの拡張

ボトムアップマッピングは、同型な部分木を葉の側から調べてゆくことにより、木のノード間の対応を求めるものであった。そのため、ボトムアップマッピングは、木の下層構造の対応は得られるが、上部構造がどのように対応しているのかがわからない。そこで、上層構造の対応を求める手法について述べる。簡単のため、ここでは、2 つの木に関する方法のみを述べるが、3 つ以上の場合も自然に拡張可能である。

木 T_1 と T_2 の間のマッピングに含まれるノードに注目し、T_1 と T_2 の間でノードを含む部分木の対応を考える。ここで、ノード $x, y \in V(T_1)$ を含む部分木とは $T_1(x \smile y)$ である。

まず、マッピングに含まれるすべてのノード間で、部分木の対応を求め、新たなマッピングを作る。

定義 7. 木 T_1 から T_2 へのマッピング M について、$M^* = \{(x \smile y, \bar{x} \smile \bar{y}) | (x, \bar{x}), (y, \bar{y}) \in M\} \cup \{(r(T_1), r(T_2))\}$ とおく。

しかし、M^* のような単純なマッピングの拡張では、マッピングの整合性が崩れてしまうことを示す。

Valiente [10] は、ボトムアップマッピングはアラインメントに真に含まれるクラスであるという定義をしている。しかし、あきらかにこの定義が実際のボトムアップマッピングとは違うことが図 1 からも明らかである。

命題 2. ボトムアップマッピングは、必ずしもアラインメントではない。また、アラインメントは、必ずしもボトムアップマッピングではない。

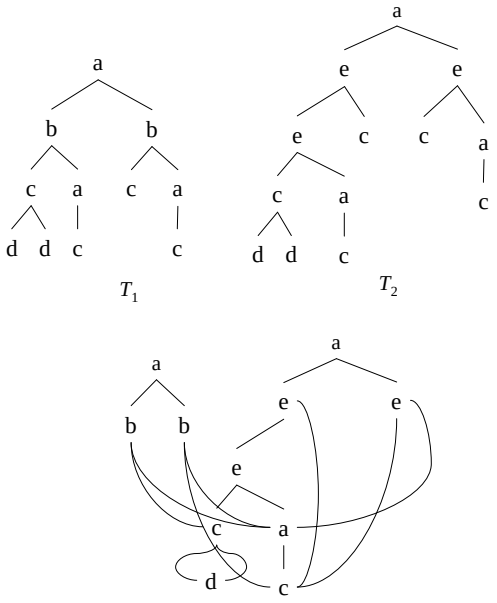

図 3: 例: 木 T_1 と T_2; および、$F = \{T_1, T_2\}$ の簡約表現 G

いずれも、図 1 が、Valiente による定義の反例になっていることから自明なので証明は省略する。修正されたマッピングのクラス階層図を図 5 に示す。

補題 3. ボトムアップマッピング M について、M^* は必ずしもマッピング条件を満たさない。

Proof. M^* がマッピング条件を満たすためには、M が制約マッピングの部分クラスでなくてはならない。ところが、ボトムアップマッピングは、アラインメントのクラスにも含まれないことから、M^* は、必ずしもマッピング条件を満たさない。 □

そこで、与えられたボトムアップマッピング M から、制約マッピングを満たすような部分マッピング $M' \subset M$ を選ぶ。このマッピング M' について、拡張を行い、M'^* を得る。図 6 に、拡張の例を示す。円で囲ったノードが拡張されたマッピングに含まれるノードである。

4. むすび

本研究では、XML や HTML 文書等の半構造データを一般的な木構造グラフとしてとらえ、木構造グラフ中に現れる部分木の繰り返し構造を発見する問題、および、3 つ以上の木構造について近似的に対応するノードを発見する問題について、効率のよいアルゴリズムを提案した。また、その過程で、Valiente によるボトムアップマッピングの意味論に関する誤りを指摘した。

今後、本手法を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

[1] D. C. Reis, P. B. Golgher, A. S. Silva, and A. H. F. Laender. Automatic web news extraction using tree edit distance. In *WWW2004*, pages 502–511, 2004.

アルゴリズム 木 $F = \{T_1, \ldots, T_n\}$ 間のマッピング

```
procedure mapping(F: trees, G: DAG, K: mapping,
                                    M_ij: mapping)
  for i = 1 to n do
    foreach node v in T_i in reverse <_{T_i}-order do
      for j = i + 1 to n do
        if M_ij[v] is undefined then
          w ← the root of T_j
          foreach node u in T_j with K[u] = K[v] do
            w ← u if u is the left to w
          end foreach
          if K[v] = K[w] then
            M_ij ← M_ij ∪ {(x, y)|corresponding pair of nodes
                    x ∈ V(T_i(v)) and y ∈ V(T_i(w))}
          end if
        end if
      end for
    end foreach
  end for
```

図 4: 木 $F = \{T_1, \ldots, T_n\}$ のボトムアップマッピングを求めるアルゴリズム

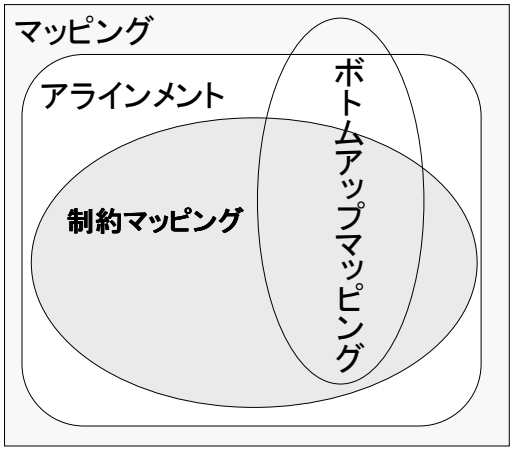

図 5: マッピングクラスの階層構造

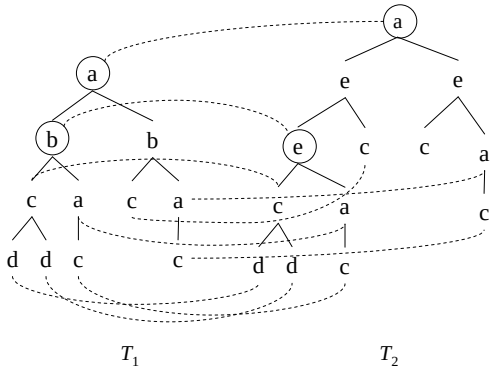

図 6: 例: ボトムアップマッピングの拡張

Science, 143:137–148, 1995.

[8] C. L. Lu, Z.-Y. Su, and G. Y. Tang. A new measure of edit distance between labeled trees. *LNCS*, 2108:pp. 338–348, 2001. COCOON 2001.

[9] K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15:205–222, 1996.

[10] G. Valiente. An efficient bottom-up distance between trees. In *Proc. 8th Int. Symposium on String Processing and Information Retrieval*, pages 212–219. IEEE Computer Science Press, 2001.

[2] J.T.-L. Wang and K. Zhang. Finding similar consensus between trees: an algorithm and a distance hierarchy. *Pattern Recognition*, 34:127–137, 2001.

[3] G. Valiente. Tree edit distance and common subtrees. Technical Report LSI-02-20-R, Universitat Politecnica de Catalunya, Barcelona, Spain, 2002.

[4] T. Kuboyama, S. Kilho, and T. Miyahara. A theoretical analysis of tree edit distance measures. *Information Processing Society of Japan, Transactions on Mathematical Modeling and Its Applications*. to appear.

[5] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, December 1989.

[6] K. Zhang and T. Jiang. Some max snp-hard results concerning unordered labeled trees. *Information Processing Letters*, 49:249–254, 1994.

[7] T. Jiang, L. Wang, and K. Zhang. Alignment of trees — an alternative to tree edit. *Theoretical Computer*