複数の重みを保持した木構造データに対する類似度算出法の検討

池田 健人 †　　波多野 賢治 ‡

† 同志社大学大学院 文化情報学研究科　　‡ 同志社大学 文化情報学部

1 はじめに

近年，計算機の性能向上や一般ユーザへの普及により，個人が大量のデータを扱う時代になってきた．その中でも，XML や HTML をはじめ，様々なデータが木構造で表現されている．木構造は単純な構造でありながら，階層構造を用いることにより多様な表現が出来るため，今後もその利用は増え続けると考えられる．そのため，大量にある木構造データの中からユーザが望むデータを的確に検索するための類似度算出法が求められている．

木構造の類似度算出法としては，二つの文字列が存在した場合に最短何文字の挿入，削除，置換をすることにより同一の文字列になるか，という Edit Distance [1] の概念を木構造へ用いた Tree Edit Distance [2] が存在する．Tree Edit Distance とは，二つの木を最短何回のノードの挿入，削除，置換により同一の木に出来るか，という尺度で算出した値であり，距離が小さいほど二つの木は類似している，と判断出来る．しかし，Tree Edit Distance では二つの木の大きさが異なっている場合，そのノード数の差の分だけ挿入または削除を行わないといけなくなり距離が自然と大きくなる．そのため，ノード数の差が大きい場合，木に出現するノードの種類が同一であっても類似していないと判断されてしまう．

そこで，本稿では木の大きさによる影響を小さくし，出現するノードの種類等の構造的な距離を算出出来るよう Tree Edit Distance の拡張を行う．

2 Tree Edit Distance

Tree Edit Distance は，二つの木を最短何回の編集操作により同一の木に出来るか計算した Tree Edit Distance によって木の類似度を測ることが出来る．本節では Tree Edit Distance に用いる編集操作と Tree Edit Distance の算出法及び Tree Edit Distance の問題点について述べる．

2.1 編集操作

Tree Edit Distance で用いられる基本操作 E には，置換，削除，挿入の三種類存在する．各編集操作について以下で述べる．

置換 置換操作は，木の形は変化せずにラベルのみ別のラベルに置換する．ノード l_1 をノード l_2 へ置換する場合，$E\{l_1 \mapsto l_2\}$ と表記する．

削除 削除操作は，削除の対象が葉ノードの場合はそのままノードを削除し，内部ノードの場合は対象ノードの子ノードを対象ノードの親ノードの子ノードとする．ノード l_2 を削除する場合，$E\{l_2 \mapsto \varepsilon\}$ と表記する．

挿入 挿入操作は，葉ノードとして挿入する場合はそのままノードを付け足し，内部ノードとして挿入する場合は親ノードの子ノードとして付け足して元の子ノードを挿入したノードの子ノードとする．ノード l_1 の子ノードとしてノード l_2 を挿入する場合，$E\{\varepsilon \mapsto l_2\}$ と表記する．

2.2 Tree Edit Distance

木 T_1 と木 T_2 を 2.1 節の編集操作を用いて同一の木に変換する際のコストを編集コストと呼び $Cost(T_1, T_2)$ とする．Tree Edit Distance では，各編集操作に掛かる重みをすべて 1 としているため編集操作の回数がそのままコストとなる．そして，T_1 と T_2 の Tree Edit Distance $TED(T_1, T_2)$ は以下のように定義される．

$$TED(T_1, T_2) = \min\{Cost(T_1, T_2)\}$$

2.3 問題点

図 1 の各木間の TED を算出した場合，表 1 のような距離行列が得られる．さらに，この結果を群平均法を用いてクラスタリングすると図 2 (a) のようになる．

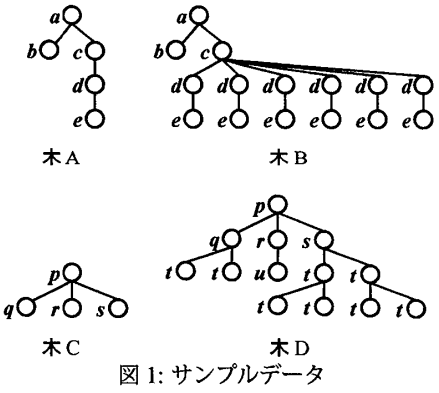

図 1: サンプルデータ

表 1: Tree Edit Distance による距離行列

	A	B	C	D
A	0	10	6	13
B	10	0	15	19
C	6	15	0	9
D	13	19	9	0

このサンプルデータは，各ノードを XML のタグ名と見立て，それぞれ木 A と B，C と D を XML の同じ DTD (Document Type Definition) に従うように作成した．そのため，理想的な結果は A と B，C と D の距離が小さくなり，同じクラスタに分析されることである．しかし，従来の Tree Edit Distance で距離を算出し，クラスタリングを行ったところ，図 2 (a) の様に同じ DTD に従った木よりも近いノード数を持った木同士の方が類似している，という結果となってしまう．

このような問題は，Tree Edit Distance が出現するノードの種類や共通する部分木の大きさを考慮せずにノードの個数の

A Similarity Calculation Method for Tree Structures with Several Weights
Kento IKEDA† and Kenji HATANO‡
†Graduate School of Culture and Information Science, Doshisha University
‡Faculty of Culture and Information Science, Doshisha University

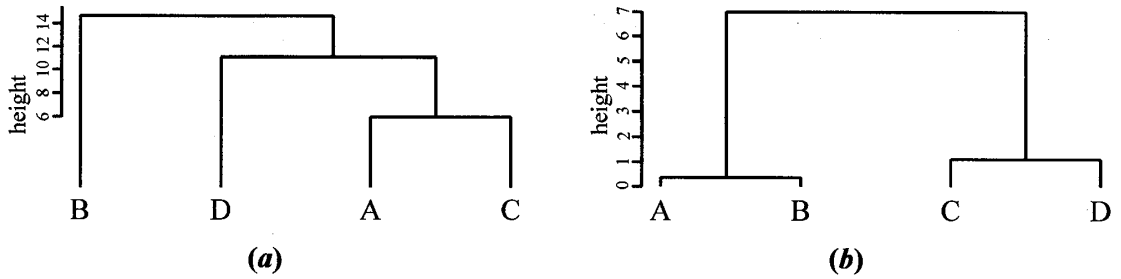

図 2: 距離行列によるクラスタリング結果

みを考慮していることにより引き起こされるため，それらを考慮した拡張を行う．

3 提案手法

本稿では，前述の Tree Edit Distance の問題点を改善するため，編集操作の対象となるノードの種類数と比較する二つの木に共通する最大部分木の大きさを考慮した重み付けを行う．以下では，木 T に含まれるノード n の個数を $N(T,n)$，木 T_1 と T_2 に共通する最大の部分木のノード数を $N(\max\{T_1 \cap T_2\})$ と表記する．なお，本稿で取り扱う木はラベル付き順序木である．

2.3 節で述べたように，ノード数の差による影響を小さくし，DTD の様な同じ規則に従っている木同士の場合に重みを大きくしたいと考えられる．そこで，各編集操作に対して以下の α_E，β_E という重み付けを行う．

重み α_E α_E は，ノードの個数よりもノードの出現の有無を重視する重み付けである．同じ規則に従った木同士では，出現するノードの種類が定められていると考えられる．そのため，そのノードが出現する回数で割ることにより，ノードの出現有無を重視しようと考え，α_E を以下のように定義する．編集操作が挿入または削除の場合には式 (1) を用いる．この式では，挿入または削除の対象となるノードの個数で割っている．また，置換の場合はに式 (2) を用いる．置換は，挿入や削除と異なり，対象となるノードは置換するノードとされるノードの二種類となる．そのため，置換の対象となるノードとされるノードの二種類の個数を足し，2 を乗じることにより挿入・削除の場合と同等に扱えるようにする．

- $E\{\varepsilon \mapsto n\}$，$E\{n \mapsto \varepsilon\}$ の場合

$$\alpha_E = \frac{1}{N(T_1,n) + N(T_2,n)} \tag{1}$$

- $E\{n \mapsto m\}$ の場合

$$\alpha_E = \frac{2}{(N(T_1,n)+(T_2,n))+(N(T_1,m)+N(T_2,m))} \tag{2}$$

重み β_E β_E は，共通する部分木の大きさを重視する重み付けである．同じ規則に従った木同士では，その規則により共通した部分木が生成されると考えられる．そのため，共通する最大の部分木のノード数で割ることにより，共通する部分木の大きさを重視しようと考え，β_E を以下のように定義する．対象の二つの木に共通する部分木が存在する場合には式 (3) を，共通する部分木が存在しない場合には 0 で割ることになるため，式 (4) を用いて算出する．

- $N(\max\{T_1 \cap T_2\}) \geq 1$ の場合

表 2: 提案手法による距離行列

	A	B	C	D
A	0.000	1.429	6.000	8.317
B	1.429	0.000	5.357	8.924
C	6.000	5.357	0.000	4.000
D	8.317	8.924	4.000	0.000

$$\beta_E = \frac{1}{N(\max\{T_1 \cap T_2\})} \tag{3}$$

- $N(\max\{T_1 \cap T_2\}) = 0$ の場合

$$\beta_E = 1 \tag{4}$$

α_E，β_E による距離の算出 既存の Tree Edit Distance における各編集操作の編集コスト 1 に α_E と β_E を乗じ，その合計値を距離として用いる．

図 1 のサンプルデータに対して α_E，β_E の重みを用いて距離の算出及びクラスタリングを行ったところ，得られた距離行列は表 2，クラスタリング結果は図 2 (b) のようになった．従来の Tree Edit Distance ではノード数の差により木 A と B，C と D はそれぞれ違うクラスタに属していたが，提案手法では同じクラスタに属することが出来た．

4 おわりに

本稿では，Tree Edit Distance への重み付けによる拡張を行った．同じ種類のノードが出現しており，その個数だけが違う場合，共通する部分木が存在する場合に有効な重み付けを行えていると考えられる．今後は，様々な木の組合せを対象とした評価実験を行い，本提案手法の有用性の確認を行うと同時に，更なる重みを組み合わせることにより理想的な距離を算出可能な手法へと改良していく予定である．

謝辞

本研究の一部は，財団法人 日揮・実吉奨学会 研究助成金 (研助第 2144 号) によるものである．ここに記して謝意を表す．

参考文献

[1] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, Vol. 10, No. 8, pp. 707–710, 1966.

[2] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, Vol. 18, No. 6, pp. 1245–1262, Dec. 1989.