テキストの構造を考慮したテキスト間類似度の効率的計算法

鈴木 潤 † 平尾 努 † 佐々木 裕 † 前田 英作 †
† 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
〒619-0237 京都府相楽郡精華町光台 2-4
{jun,hirao,sasaki,maeda}@cslab.kecl.ntt.co.jp

1 はじめに

テキスト間の類似度計算は，機械翻訳，文書分類，文書要約，情報検索，質問応答等，テキスト処理の様々な応用分野で必須の要素技術である．従来手法として，テキストに出現した単語を要素としたベクトルによるベクトル空間法 [1] におけるコサイン距離や，テキスト間の編集距離 [2] により類似度を計算する方法が使われてきた．

本稿では、これらの手法とは異なり、テキスト内の様々な構造を反映した類似度を提案する。具体的には、テキストの構造を非循環有向グラフで記述し、非循環有向グラフ間の類似度を定義することで、テキストの構造を反映したテキスト間の類似度を計算する方法である。また同時に、Convolution Kernel[3] の枠組を利用して、非循環有向グラフカーネル (Directed Acyclic Graph Kernel: DAG Kernel) を定義し、テキスト間類似度の高速かつ効率的な計算方法を提案する。

本稿と同様に，構文構造を考慮したテキスト間の類似度をカーネル関数を用いて計算する方法が既に提案されている [6, 7]．これらの研究では、構文解析木用に提案された Tree Kernel[4] を拡張し、一般的な木構造に対して類似度を計算するアプローチを取っている。それに対して、本稿では、テキストの持つ構造を非循環有向グラフを用いてテキスト内の構造を表現する点が異なる。これにより、木構造に限らず、テキスト内の構造を柔軟に取り扱うことが可能となる。

2 Convolution Kernels

構造を持った離散データを扱うカーネルの基本的な概念として，Convolution Kernel が提案されている．Convolution Kernel では，比較対象となるオブジェクトが幾つかの部分によって構成されているとき，部分同士のカーネル (部分カーネル) を組合せて比較対象のカーネルを定義している．この方法を用いることにより，入力時及び計算時に比較対象を明示的にベクトル化せずに類似度を計算できるという利点があり、再帰式で定義される計算式から，高速かつ効率的に類似度を計算できる．$x \in X$ を複数の要素で構成されたオブジェクトであるとし，$\{\mathbf{x} = (x_1, \ldots, x_D) : x_d \in X_d (1 \leq d \leq D)\}$ をその構成部分とする．もし，$\mathbf{x}$ が x の構成要素である時，$R(\mathbf{x}, x) = 1$ とし，x を構成することが可能な $\mathbf{x}$ の集合を $R^{-1}(x) = \{\mathbf{x} : R(\mathbf{x}, x) = 1\}$ と定義する．

二つのオブジェクト x, y は式 (1) で定義される．

$$K(x,y) = \sum_{\mathbf{x} \in R^{-1}(x)} \sum_{\mathbf{y} \in R^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d) \qquad (1)$$

ただし，Convolution Kernel は抽象的な概念であり，部分カーネルの定義によってカーネルの実体が決定する．テキスト処理の分野では，Convolution kernels の実体として，Tree Kernel[4] や String (Subsequence) Kernel[5] 等が提案され実験結果が報告されている．

また，Convolution Kernels の枠組では，入力オブジェクトの構造の大きさが一定ではないため正規化の方法として式 (2) を用いることが多い．

$$\hat{K}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x) \cdot K(y,y)}} \qquad (2)$$

3 非循環有向グラフカーネル (DAG Kernel)

3.1 非循環有向グラフ間の類似度の定義

非循環有向グラフ内のノードはそれぞれ属性 (ラベル) を持つこととする．本稿で提案する非循環有向グラフ間の類似度は，2 つの非循環有向グラフ内の全部分パスから得られる属性列の一致度の総和と定義する．つまり，非循環有向グラフ間の類似度は，全部分パスから得られる属性列を要素，頻度を値としたベクトルの内積と等価である．

3.2 部分パスから属性列を得る方法

扱う非循環有向グラフのノードは複数の属性を持つことを許す。例えば，テキストの場合，ノードの属性として単語 (形態素) の原形，品詞、意味情報等である．図 1 に，非循環有向グラフに対するカーネルで扱う構造の例を示す．図中では，ノードの属性として，単語 (形態素) の原形，品詞を用いている．

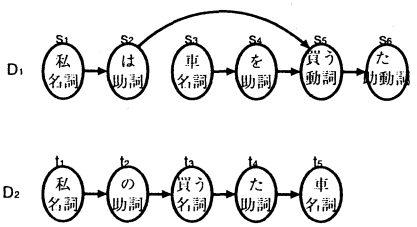

図 1: 非循環有向グラフに対するカーネルで扱う構造の例

次に，部分パスから属性列を得る際に以下の条件を用いる．

1. ペナルティ λ を与えてノードのスキップを許す
2. 組合せ数 n 以下の属性列のみを扱う

これらの条件は，String Kernel[5] で扱われている枠組と同じである*1．本稿では，便宜上「パス」という言葉を用いているが，本稿での「パス」にはノード単体も含むこ

*1 文献 [5] では，文字列長に依存したペナルティを与え，本稿では，スキップ数に依存したペナルティを与えるため，厳密には同じではない

表 1: D_1 の単語のみを使用した全部分パスから得られる属性列

属性列	値	属性列	値	属性列	値	属性列	値	属性列	値
私	1	私-は	1	私-は-買う-た	1	車-*-*-た	λ^2	を-買う	1
は	1	私-*-買う	λ	は-買う	1	車-を-買う	1	を-*-た	λ
車	1	私-*-*-た	λ^2	は-*-た	λ	車-を-*-た	λ	買う-た	1
を	1	私-は-買う	1	は-買う-た	1	車-*-買う-た	λ		
買う	1	私-は-*-た	λ	車-を	1	車-を-買う-た	1		
た	1	私-*-買う-た	λ	車-*-買う	λ	を-買う-た	1		

表 2: D_2 の単語のみを使用した全部分パスから得られる属性列

属性列	値	属性列	値	属性列	値	属性列	値	属性列	値
私	1	私-の-買う	1	私-の-買う-*-車	λ	の-買う	1	買う-*-車	λ
の	1	私-*-買う	λ	私-の-*-た-車	λ	の-買う-た	1	買う-た-車	1
買う	1	私-の-買う-た	1	私-の-*-*-車	λ^2	の-*-た	λ	た-車	1
た	1	私-の-*-た	λ	私-*-買う-た-車	λ	の-買う-た-車	1		
買う-た	1	私-*-買う-た	λ	私-*-買う-*-車	λ^2	の-買う-*-車	λ		
車	1	私-*-*-た	λ^2	私-*-*-た-車	λ^2	の-*-た-車	λ		
私-の	1	私-の-買う-た-車	1	私-*-*-*-車	λ^3	の-*-*-車	λ^2		

ととする．つまり，ノード単体から得られる属性も属性列に含める．

部分パスから得られる属性列として，例えば，図 1 の D_1 の $s_1 - s_2$ のパスからは，ノードの持つ属性の全組合せである「私-は」,「私-助詞」,「名詞-は」,「名詞-助詞」という 4 つの属性列を得ることとする．

次に，全てのノードは特殊な属性「$*$」を持っていると定義する．部分パスから属性列を得る際には，属性「$*$」も考慮して属性列を得る．ただし，始点および終点ノードの属性「$*$」は考慮しない．例えば，パス $s_3 - s_4 - s_5 - s_6$ に含まれる属性列は，「(車, 名詞)-(を, 助詞)-(買う, 動詞)-(た, 助動詞)」のそれぞれの組合せ 16 個，「(車, 名詞)-*-(買う, 動詞)-(た, 助動詞)」の組合せ 8 個，「(車, 名詞)-(を, 助詞)-*-(た, 助動詞)」の組合せ 8 個，「(車, 名詞)-*-*-(た, 助動詞)」の組合せ 4 個の計 36 の属性列となる．

最後に，2 つのグラフから得られた全ての属性列間で一致する属性列の総和を，グラフ間の類似度とする．特殊な属性「$*$」は，属性列間の一致を判定する際には考慮せず，「A-*-B-C」と「A-B-C」，或は，「A-*-*-B-C」と「A-*-B-*-C」は一致すると判定する．ただし，属性「$*$」を含む数だけペナルティ $\lambda(0 < \lambda \leq 1)$ を与える．このような枠組で，条件 1. のノードのスキップを実現する．また，条件 2. で示したように，属性列の一致を判定する際に，全ての属性列ではなく，組合せ数 (属性数)n 以下の属性列の一致のみを扱う．これらの枠組により，柔軟なグラフ間の類似度を得ることができる．

表 1, 2 に図 1 の D_1, D_2 の全部分パスを展開した際に得られる属性列を示す。また，表 3 に属性列間の一致と最終的な類似度の値を示す．ただし，この例では，簡単のため，属性として単語の原形のみを用いている．

3.3 類似度の効率的な計算方法

本稿で定義した非循環有向グラフ間の類似度の効率的な計算方法を述べる．

式 (3)～(6) に，テキストに含まれる属性の全組合せを考慮した場合の式を示す．

$$K_{DAG}(D_1, D_2) = \sum_{s_i \in s} \sum_{t_j \in t} K(s_i, t_j) \tag{3}$$

$$K(s_i, t_j) = \begin{cases} 0 & \text{if} \quad s_i \neq t_j \\ \big(K'(s_i, t_j) + 1\big) \cdot val(s_i, t_j) \end{cases} \tag{4}$$

表 3: 図 1 中の D_1, D_2 の類似度

D_1		D_2		一致度
組合せ数 $n = 1$				
私	1	私	1	1
車	1	車	1	1
買う	1	買う	1	1
た	1	た	1	1
組合せ数 $n = 2$				
買う-た	1	買う-た	1	1
私-*-買う	λ	私-*-買う	λ	λ^2
私-*-*-た	λ^2	私-*-*-た	λ^2	λ^4
組合せ数 $n = 3$				
私-*-買う-た	λ	私-*-買う-た	λ	λ^2
全ての組合せを考慮した際の類似度				$5+2\lambda^2+\lambda^4$

$$K'(s_i, t_j) = \begin{cases} 0 & \text{if} \quad rel(t_j) = \phi \\ \displaystyle\sum_{t_q \in rel(t_j)} \big(\lambda K'(s_i, t_q) + K''(s_i, t_q)\big) \end{cases} \tag{5}$$

$$K''(s_i, t_j) = \begin{cases} 0 & \text{if} \quad rel(s_i) = \phi \\ \displaystyle\sum_{s_p \in rel(s_i)} \big(\lambda K''(s_p, t_j) + K(s_p, t_j)\big) \end{cases} \tag{6}$$

ノードが複数の属性を持つことを許したので，ノード s_i, t_j 内の属性で一致する属性の数を返す関数 $val(s_i, t_j)$ を定義する．例えば，$val(s_1, t_1) = 2$, $val(s_1, t_5) = 1$, $val(s_1, t_2) = 0$ となる．関数 $rel(s_i)$ は，ノード s_i に係るリンクを持つノードの集合を返す関数とする．$rel(s_i) = \phi$ は，ノード s_i に係るリンクを持つノードが無いことを意味する．例えば，$rel(s_5) = \{s_2, s_4\}$, $rel(s_1) = \phi$, $rel(s_3) = \phi$ となる．

次に，式 (7),(8) に，組合せ数 n を導入した場合の式を示す．

$$K_{DAG}(G_1, G_2) = \sum_{m=1}^{n} \sum_{s_i \in s} \sum_{t_j \in t} K_m(s_i, t_j) \tag{7}$$

$$K_m(s_i, t_j) = \begin{cases} 0 & \text{if} \quad s_i \neq t_j \\ val(s_i, t_j) & \text{if} \quad m = 1 \\ K'_{m-1}(s_i, t_j) \cdot val(s_i, t_j) \end{cases} \tag{8}$$

組合せ数 n を考慮する際の関数 K'_m, K''_m は，式 (5)，式 (6) の K', K'' をそれぞれ，K'_m, K''_m に変えた式となるため，ここでは省略する．

```
Algorithm DAG Kernel
for (i = 1; i ≤ |D1|; i++)
  for (j = 1; j ≤ |D2|; j++)

    foreach sp ∈ rel(si)
      K''[si, tj] += λK''[sp, tj] + K[sp, tj]
    end

    foreach tq ∈ rel(tj)
      K'[si, tj] += λK'[si, tq] + K''[si, tq]
    end

    if (val(si, tj) > 0)
      K[si, tj] = (K'[si, tj] + 1) · val(si, tj)
    end
  end
end
return  Σ_{si∈s} Σ_{tj∈t} K[si, tj]
```

図 2: 非循環有向グラフカーネルのアルゴリズム (全組合せ)

```
Algorithm DAG Kernel n_comb
for (i = 1; i ≤ |D1|; i++)
  for (j = 1; j ≤ |D2|; j++)

    foreach sp ∈ rel(si)
      for (m = 0; m < n; m++)
        K''_m[si, tj] += λK''_m[sp, tj] + K_m[sp, tj]
      end
    end

    foreach tq ∈ rel(tj)
      for (m = 0; m < n; m++)
        K'_m[si, tj] += λK'_m[si, tq] + K''_m[si, tq]
      end
    end

    if (val(si, tj) > 0)
      K0[si, tj] = val(si, tj)
      for (m = 1; m < n; m++)
        Km[si, tj] = K'_{m-1}[si, tj] · val(si, tj)
      end
    end
  end
end
return  Σ_{m=1}^{n} Σ_{si∈s} Σ_{tj∈t} Km[si, tj]
```

図 3: 非循環有向グラフカーネルのアルゴリズム (組合せ数 n 以下)

3.4 実装上の高速計算法

実際の計算では、ノードの計算順序を事前に決定することで、効率良く計算することができる。任意のノードを計算するとき、そのノードを終端とするパスを持つ全てのノードの計算が事前に終了しているように順序付けを行なえばよい。対象が非循環有向グラフなので、この順序は一意には決定しないが、必ず一通り以上得ることができる。D_1 の場合、例えば $\langle n_1, n_2, n_3, n_4, n_5, n_6\rangle$ や $\langle n_3, n_4, n_1, n_2, n_5, n_6\rangle$ のような順序になる。

類似度は、動的計画法と同様にテーブルを用いて容易に計算できる。テーブルの左上から右下に向かって再帰式の定義に従って順次計算すれば良い。任意のノード対 (s_i, t_j) の値を計算する際には、その計算に必要な値は全てテーブル上に保持されているので、簡単かつ高速に計算することが可能である。

図 2 に，全ての属性の組合せを考慮する場合，図 3 に属性の組合せ n 以下の場合の類似度の効率的な計算アルゴリズムを示す．ここで，i，j は，前述のソートを行った後の順序で番号付けされたノード番号とする．

このように計算することで，全ての組合せを考慮した類似度を計算する際には，計算量 $O(|D_1||D_2|)$ となり，比較対象のノード数の積で計算することができる．また，組合せ数 n 以下を用いる際には，計算量 $O(n|D_1||D_2|)$ となる．

3.5 非循環有向グラフの利点

非循環有向グラフの枠組では，形態素解析や依存構造解析の n-best 結果のような入力でも扱うことができる。実際の応用では、冗長な解析結果を用いた方が性能の向上を期待できる場合がある。図 4 に形態素解析結果を非循環有向グラフで表した例を示す。

S: すももももももものうち

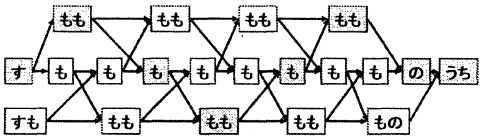

図 4: 形態素解析の n-best 結果を非循環有向グラフで表現した例

このように，非循環有向グラフで表現することさえできれば、自然言語処理解析の結果のどのような入力に対してもテキストの類似度を計算することが可能である。

4 実験

本稿で提案した手法の有効性を検証するため，実際のテキストを用いて実験をおこなった．

本稿で提案する非循環有向グラフ (DAG) の比較手法として，文節依存構造を用いた Tree Kernel(TK)，単語単位の String Subsequence Kernel(SSK)，出現単語全てを要素とし頻度を値としたベクトルによるコサイン距離 (BOW1)，名詞及び未知語のみを要素としたベクトルによるコサイン距離 (BOW2) を用いた．Tree Kernel は，文献 [6, 7] を参考に子ノードの順序及び組合せ数を考慮して類似度を計算できるように改良を加えた．また，ノードは複数の属性を持つことを許した．

ノードの属性には，単語 (形態素の原形) と意味情報を用いた．BOW1，BOW2 も同様に単語と意味情報を要素として用いた．なお，形態素には chasen[*2] の出力，意味情報には日本語語彙大系 [8] に記述された意味カテゴリ，文節及び依存構造には cabocha[*3] の出力を用いた．

パラメタ λ は，0.1 から 0.9 まで 0.1 づつ変化させ，最も良い結果を得た λ の値を用いた．

4.1 実験 1：質問タイプによる類似性評価

本稿では，同一の質問タイプに分類される質問文は類似度が高いと考え，提案した類似度の有効性を検証するため，以下の手順で実験をおこなった．

1. テストセットから質問文を一つ抽出

[*2] http://chasen.aist-nara.ac.jp/index.html.ja

[*3] http://cl.aist-nara.ac.jp/~taku-ku/software/cabocha/

表 4: 質問タイプに基づく質問文の類似度判定結果

n	2	3	4	5	6
DAG	**0.430**	0.420	0.411	0.398	0.382
SSK	0.421	**0.423**	0.418	0.408	0.396
TK	**0.311**	0.274	0.250	0.245	0.244
BOW1	0.374				
BOW2	0.401				

表 5: 要約文の対応付けに基づく文類似度判定結果

n	2	3	4	5	6
DAG	**0.502**	0.459	0.387	0.353	0.337
SSK	**0.475**	0.432	0.373	0.339	0.313
TK	**0.174**	0.083	0.035	0.020	0.021
BOW1	0.394				
BOW2	0.451				

2. 抽出された質問文とそれ以外の質問文の類似度を全て計算
3. 類似度の高い順で順位付け
4. 抽出された質問文と同じ質問タイプに属する質問文の mean average precision で性能評価

質問文には，NTCIR-QAC1[*4]の formal-run, dry-run と追加で発表された質問文を合わせた計 1011 文を用いた．また，質問タイプは CRL-QA データ [*5]の分類に基づいてタイプを付加した．

表 4 に結果を示す．

4.2 実験 2：要約文の類似性評価

文献 [9] で用いられた毎日新聞での複数文書要約を対象とした重要文データに対して読売新聞の文を用いて対応付けを行い，提案手法の有効性を検証した．なお，文献 [9] で用いられた 12 トピックに関しては，毎日新聞と読売新聞との間で文書間の対応付け，文間の対応付けを人手により行い正解データを作成している．

実験には，毎日新聞と読売新聞で一対一対応がついている 173 文を用いた．

1. 毎日新聞に基づき作成されたあるトピックに対する重要文データより対応する読売新聞の記事集合に含まれる一文と一対一で対応している文を抽出する．
2. 1. で抽出した全ての文に対して，対応する読売新聞の記事集合に含まれる各文との類似度を計算する．
3. 類似度の高い順に順位付けを行う．
4. 上位 10 件までを対象として Mean Reciprocal Rank (MRR) を計算する．

最終的な性能は，1〜4 の手順を 12 トピックに関して行い，MRR の平均で評価した．表 5 に結果を示す．

4.3 考察

2 種類の実験で，提案手法は，代表的な従来手法である BOW1，BOW2 による類似度よりも良い結果を得ることができた．質問文タイプによる質問文の類似性評価実験は，質問文の意図によるタイプ分類なので，テキストの持つ意味的な要素が重要な特徴になる．また，要約文間の類似性評価も，テキストの内容的な類似性を評価する問題なので，テキストの構造的な情報が類似度を評価する上で重要と考えられる．このような問題に対しては，提案手法を用いることにより性能を向上させることができることが確かめられた．

Tree Kernel も同様にテキストの構造を反映した類似度であるが，木構造の一致を類似度と定義しているため，提案手法と比べて比較的制約が厳しくなる．本稿で扱った実験では，より柔軟なテキスト間のマッチングを必要としたため，良い成績を得られなかったと考えられる．

5 今後の課題

本稿で提案した類似度はカーネル法に基づいているため、近年、機械学習の分野で注目を集めている Support Vector Machine にそのまま適用することができる。今後は、テキスト間の類似度の評価をおこなうとともに、テキスト処理タスク中の分類問題に対しても、本手法を適用し性能の評価をおこなっていく予定である。

今後は、より多くの構造を取り入れて実験をおこない、有効性を検証していく予定である [10]。

6 まとめ

本稿では，テキストの構造を非循環有向グラフで表現し，非循環有向グラフ間の類似度を定義することで，テキスト内の構造を反映した類似度を提案した．また，実際の計算方法として，非循環有向グラフカーネルを定義し，非循環有向グラフ間の高速かつ効率的な類似度計算方法を提案した。提案手法を，実データを用いて有効性を検証し、構造を考慮しない手法より良好な成績を得ることができた。

参考文献

[1] Salton, G., Wong, A. and Yang, C.: A Vector Space Model for Automatic Indexing, *Communicatoion of the ACM*, Vol. 11, No. 18, pp. 613–620 (1975).

[2] Needleman, S. B. and Wunsch, C. D.: A General method applicable to the Search for Similarities in the Amino-Acid Sequence of Two Proteins, *Journal of Molecular Biology*, Vol. 3, No. 48, pp. 443–453 (1970).

[3] Haussler, D.: Convolution Kernels on Discrete Structures, *Technical Report UCS-CRL-99-10*, UC Santa Crus (1999).

[4] Collins, M. and Duffy, N.: Convolution Kernels for Natural Language, *Proc. of NIPS'2001* (2001).

[5] Lodhi, H., Saunders, C., Taylor, J.-., Cristianini, N. and Watkins, C.: Text Classification using String Kernel, *Journal of Machine Learning Research*, Vol. 2, pp. 419–444 (2002).

[6] 高橋哲朗, 乾健太郎, 松本裕治: テキストの構文的類似度の評価方法について, 情報処理学会 自然言語処理研究会 NL-150-24, pp. 163–170 (2002).

[7] 鹿島久嗣, 小柳光生: 半構造データへのサポートベクターマシンの適用, 人工知能学会 人工知能基礎論研究会 SIG-FAI48, pp. 57–62 (2002).

[8] 池原悟, 宮崎正弘, 白井諭, 横尾昭男, 中岩浩巳, 小倉健太郎, 大山芳史, 林良彦: 日本語語彙大系, 岩波書店 (1997).

[9] 平尾努, 賀沢秀人, 磯崎秀樹, 前田英作, 松本裕治: 機械学習による複数文書からの重要文抽出, 自然言語処理, Vol. 10, No. 1, pp. 81–108 (2002).

[10] 鈴木潤, 平尾努, 佐々木裕, 前田英作: 階層構造を利用したテキスト間類似度の効率的計算法, 情報処理学会自然言語処理研究会 NL-150 (2002).

[*4] http://www.nlp.cs.ritsumei.ac.jp/qac/

[*5] http://www.cs.nyu.edu/~sekine/PROJECT/CRLQA/