社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2003－FI－ 72　(6)
2003－NL－157　(6)
2003／9／29

素性選択機能をもったString Kernelの提案

鈴木 潤 †　平尾 努 †　磯崎 秀樹 †　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,isozaki,maeda}@cslab.kecl.ntt.co.jp

概要

本稿では，String Kernel に対する素性選択手法について議論する．String Kernel を含む Convolution Kernels の枠組では，全ての部分構造間の部分カーネルの総和を全体のカーネルの値と定義している．しかし，可能な全ての部分構造を素性として使用すると，素性空間の次元数が高くなり，データスパースネスの問題が起こることが実験的に示されている．このため String Kernel では，カーネル計算に使用する部分記号列を部分記号列のサイズに応じて選択する枠組が導入されている．本稿では，この従来手法による素性選択基準の問題点について議論し，その問題点を改善した素性選択手法を提案する．さらに，質問分類と文のモダリティ判定の二つのタスクを用いて，本稿で取り上げた従来手法の問題点の正当性を検証するとともに，従来手法との比較実験によって提案手法の有効性を示す．

キーワード: Convolution Kernels, String Kernel, 素性選択

String Kernel with Feature Selection Function

Jun Suzuki† Tsutomu Hirao† Hideki Isozaki† Eisaku Maeda†
†NTT Communication Science Laboratories, NTT Corp.
2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237 Japan
{jun,hirao,isozaki,maeda}@cslab.kecl.ntt.co.jp

Abstract

This paper discusses feature selection methods about String Kernel. The kernel value of Convolution Kernels is defined as a sum of all sub-kernels between all sub-structures of input objects. However, it is known experimentally that the data-sparseness problem arises if treating all sub-structures to calculate the kernel value. For this reason, the framework of a feature selection method based on the size of sub-sequences is introduced in String Kernel. In this paper, first, we discuss an issue about this kind of conventional feature selection method, and then we propose a new feature selection method. After that, we confirm the issue of conventional method experimentally and compare the performance between a conventional method and the proposed method by using using question classification task and sentence modality identification task

Keywords: Convolution Kernels, String Kernel, Feature Selection, Feature Mining

1　はじめに

近年，String Kernel[1] や Tree Kernel[2] など離散構造を扱うことが可能なカーネル関数が提案され，幾つかの自然言語処理タスクにおいてその有効性が示されている．

String Kernel や Tree Kernel は，より一般的に Convolution Kernels[3, 4] と呼ばれるカーネル関数の一種として定義される．この Convolution Kernels は，比較対象となる入力オブジェクトが幾つかの部分構造によって構成されているとき，その部分構造同士に定義された部分カーネルの総和を比較対象オブジェクトのカーネルの値と定義している．例えば，String Kernel では，入力オブジェクトが記号列 (例：abcd) の場合に用いられ，その部分構造は入力記号列中のギャップを許した部分記号列 (例：a,b,c,d,ab,ac,ad,bc,bd,cd,abc,abd,acd,bcd,abcd) と考える．このとき部分カーネルを部分記号列間の重み付き一致数とし，その総和を全体のカーネルの値と定義している．

この Convolution Kernels の枠組を用いることにより，テキストを数値ベクトルに変換することなくテキストの持つ離散構造を自然な形で扱うことができると同時に，これまで明示的に計算することが計算量的に困難であったテキストの構造的な情報を反映した効率的な計算が可能になる．

しかし，自然言語処理タスクに Convolution Kernels を用いる際，定義通り全ての部分構造を用いてカーネルの値を計算すると，データ数に対して素性空間の次元が高くなりすぎて，良い分類性能を得られない場合が多いことが知られている．

実際に，Convolution Kernels に関連する様々な文献でこの問題点を指摘しており，いくつかの対処法が考案されている [2, 5, 1, 6]．最も一般的な方法は，部分構造のサイズに基づく制約を与え，カーネル計算に使用する部分構造を選択する方法である．具体的には，部分構造のサイズに

対して閾値を与え，閾値以下のサイズの部分構造のみをカーネルの計算に使用する．しかしながら，部分構造のサイズとそこから得られる素性の有効性との間に相関があるとは言えず，与えた閾値より大きいサイズの部分構造の中にも重要な特徴が存在する可能性や，閾値以下の部分構造が性能に悪影響を与える可能性が考えられる．

そこで，本稿では二クラス分類問題に対する String Kernel を例に取り上げ，従来用いられている部分構造のサイズに基づく制約ではなく，各部分記号列の統計量に基づく素性選択方法を提案する．実データを用いた実験で，従来手法と提案手法を比較することにより，前述の，「与えた閾値より大きいサイズの部分構造の中にも重要な特徴が存在する可能性」と，「閾値以下の部分構造が性能に悪影響を与える可能性」について検証し，さらに，提案手法の有効性を示す．

以下本稿では，2 節で String Kernel について説明し，3 節で Convolution Kernels の問題点を示し，その対処法について述べる．4 節で従来手法と提案手法の比較実験おこない，5 章で実験結果に対する考察をおこなう．

2 String Kernel

Convolution Kernels の枠組の中で，記号列を入力とするカーネル関数として String Kernel が提案されている．また，その拡張形として，テキスト中の単語を一つの記号と見做して扱う，Word Sequence Kernel[6] がある．これ以外にも，対象タスクに特化した拡張等をおこなった String Kernel の派生形が存在する．

Σ を有限の記号の集合とし，Σ^n を Σ に属する n 個の記号の並びで構成される記号列の集合とする．S，T をそれぞれ Σ の要素によって作られる記号列とし，s_i，t_j を，それぞれ S の i 番目，T の j 番目の記号と定義し，$S = s_1 \ldots s_i \ldots s_{|S|}$ の様に表すものとする．S の任意の部分記号列 $S[i:j] = s_i \ldots s_j$ 中に記号列 u がギャップを許して含まれている場合 $(s_i = u_1, s_j = u_{|u|})$，$u$ を構成する記号の出現位置を $\mathbf{i} = (i_1 : i_{|u|})$ で表す．このとき，$S[\mathbf{i}]$ の記号列長は $len(\mathbf{i}) = i_{|u|} - i_1 + 1$ となる．

以上の表記を用いると，サイズ n の部分記号列に着目した String Kernel の値は以下のように定義されている．

$$K_{\mathrm{SK}}(S,T) = \sum_{u \in \Sigma^n} \sum_{\mathbf{i}|u=S[\mathbf{i}]} \lambda^{len(\mathbf{i})} \sum_{\mathbf{j}|u=T[\mathbf{j}]} \lambda^{len(\mathbf{j})} \tag{1}$$

ここで，便宜上「such that」を「|」で書く．これは，記号列 S，T 間のギャップを許した部分記号列の重み付き一致度の総和を計算していることになる．このときの重みとは，部分記号列の長さ $len(\mathbf{i})$ を減衰率 $\lambda (0 \leq \lambda \leq 1)$ に乗算した値 $\lambda^{len(\mathbf{i})}$ である．

ただし，String Kernel は，計算量の問題から実際には式 (1) ではなく，式 (2)〜(5) で定義される再帰式に基づいて計算される*1．共通部分記号列のサイズに関する変数を $m (1 \leq m \leq n)$ とし，S_i，T_j をそれぞれ S の i 番目の要素，T の j 番目の要素を表す変数とする．

$$K_{\mathrm{SK}}(S,T) = \sum_{1 \leq i \leq |S|} \sum_{1 \leq j \leq |T|} K_n(S_i, T_j) \tag{2}$$

$$K_m(S_i,T_j) = \begin{cases} 0, & \text{if} \quad s_i \neq t_j \\ \lambda^2 K''_{m-1}(S_i,T_j), & \text{otherwise} \end{cases} \tag{3}$$

$$K'_m(S_i,T_j) = \begin{cases} 0 \quad \text{if} \quad i = 0 \text{ or } j = 0, \\ \lambda K'_m(S_{i-1},T_j) + K_m(S_{i-1},T_j) \\ \qquad \text{otherwise} \end{cases} \tag{4}$$

$$K''_m(S_i,T_j) = \begin{cases} 0 \quad \text{if} \quad i = 0 \text{ or } j = 0, \\ 1 \quad \text{if} \quad m = 0, \\ \lambda K''_m(S_i,T_{j-1}) + K'_m(S_i,T_{j-1}) \\ \qquad \text{otherwise} \end{cases} \tag{5}$$

このカーネル計算は，動的計画法の要領で 2 次元のテーブルを用意し，左上から右下に向けて計算式にしたがって計算することで容易に計算することができる．図 1 に，動的計画法と同様の計算手法によって得られた実際の計算結果の例を示す．空欄は 0 を表す．例えば，$K''_1(s_3,t_4)$ は，$K''_1(s_3,t_3)$ と $K'_1(s_3,t_3)$ の値から計算されるが，この値は事前に計算されているため，テーブルから容易に得ることができる．最終的なカーネルの値は図中の斜線が入っている場所の値の総和で得ることができる．

String Kernel も含めて，Convolution Kernel の枠組では，入力オブジェクトのサイズが一定ではないため，最終的なカーネルの値として式 (6) を用いて正規化する．入力オブジェクトを x，y とする．

$$\hat{K}(x,y) = \frac{K(x,y)}{\sqrt{K(x,x) \cdot K(y,y)}} \tag{6}$$

x, y の自分自身に対するカーネルの値で正規化することとなり，$x = y$ の時 $\hat{K}(x,y) = 1$ となる．また，$\hat{K}(x,y)$ の値は，オブジェクト x，y 間の類似度と定義される．

2.1 本稿で扱う String Kernel

ここで，本稿で取り上げる問題をより明確に議論するために，文献 [1] で提案されたオリジナルの String Kernel の定義を若干変更する．以下に，オリジナルの String Kernel との差分を示す．

1. 任意のサイズ n の部分記号列のみをカーネルの計算に使用する枠組から，n 以下の部分記号列全てを使用する枠組に変更
2. 部分記号列の重みを記号列の長さに基づく減衰からギャップ数に基づく減衰に変更

まず，一つ目の変更の理由を述べる．単語を単位とした記号列を扱う場合，記号数は数万のオーダーになるため，オリジナルの String Kernel では，n が大きいと記号列の平均出現確率が非常に低くなり，多くの素性値が 0 付近に集中するという問題が発生する．そこで，Word Sequence

*1 文献 [1] で紹介されている式と違う形であるが，本稿の以後の式展開のために書き直した式を用いている．計算しているものは同じである

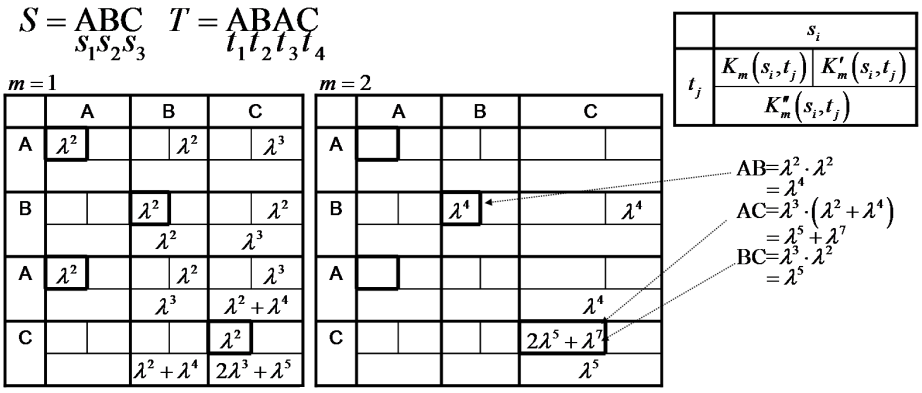

図 1: 実際の String Kernel の計算方法

Kernel[6] に記述されているように，閾値以下のサイズを持つ部分記号列を全て扱う枠組に変更する．Convolution Kernels では，全ての部分構造をカーネルの計算に使用するように定義されているので，この方法の方がより一般化された枠組であり，オリジナルの String Kernel の方が特殊形であると考えることができる．実際，使用する部分記号列中の閾値を最も長い入力記号列のサイズに設定すれば，全ての部分記号列を用いてカーネルの計算をしていることと等価になる．

二つ目は，部分記号列の長さによる重みを公平にするための変更である．String Kernel の枠組では，部分記号列の長さに基づいて減衰率 λ を与えていた．つまり，部分記号列のサイズ n を λ の乗数に持った値 λ^n が，それぞれの部分記号列の最大値となるため，部分記号列のサイズが大きくなればなる程，値は小さくなり，それらの一致度は相対的にカーネルの値に反映されなくなる．そこで，これを部分記号列中のギャップ数に基づいて値を減衰させるように変更する．例えば，これまでは部分記号列 abcd 間の一致による重み λ^8，に対し変更により重み $1(\lambda^0)$ が与えられる．このように変更することで，部分記号列のサイズが大きく程，カーネルの値に反映されなくなる問題を解決することができる．

以降の議論では，これらの変更をおこなった String Kernel のことを便宜上，拡張 String Kernel とよぶ．

拡張 String Kernel の定義式は，式 (1) を基に，$\gamma(\mathbf{i}) = len(\mathbf{i}) - n$ を用いて以下のようにかける．

$$K_{\mathrm{ESK}}(S,T) = \sum_{m=1}^{n} \sum_{u\in\Sigma^m} \sum_{\mathbf{i}|u=S[\mathbf{i}]} \lambda^{\gamma(\mathbf{i})} \sum_{\mathbf{j}|u=T[\mathbf{j}]} \lambda^{\gamma(\mathbf{j})} \qquad (7)$$

また，同様に，式 (2)，(3) をそれぞれ式 (8)，(9) のように変更することで，拡張 String Kernel の計算式が得られる．

$$K_{\mathrm{ESK}}(S,T) = \sum_{m=1}^{n} \sum_{1\le i\le |S|} \sum_{1\le j\le |T|} K_m(S_i,T_j) \qquad (8)$$

$$K_m(S_i,T_j) = \begin{cases} 0, & \text{if } s_i \neq t_j \\ K''_{m-1}(S_i,T_j), & \text{otherwise} \end{cases} \qquad (9)$$

実際の計算方法の枠組は図 1 と同じである．

3 Convolution Kernels の使用する際の注意点およびその対処法

まず，前提条件として，本稿では，Support Vector Machine(SVM)[7] 等のカーネル法に基づく 2 クラスの学習分類問題を扱うものとして議論を進める．本節では，その際に Convolution kernels に属するカーネル関数を使用する場合の注意点について議論する．

まず，Convolution Kernels の定義では，カーネルの計算に使用する入力オブジェクトの構造のサイズが大きくなると，事実上の素性数は指数関数的に増加するため，データスパースネスの問題が起こり，効率的な学習ができない可能性が高くなることが経験的に知られている．例えば，拡張 String Kernel では，理論上の素性空間の次元数は $\sum_{i=1}^{n} |\Sigma|^i$ であり，この問題の可能性は容易に推察できる．

以下に，String Kernel を用いて実際にどのような現象が起こるかを述べる．ある入力記号列 $x, y (x \neq y)$ があるとき，扱う構造のサイズを大きくすると，自分自身に対するカーネルの値 $K(x,x), K(y,y)$ は非常に大きくなるが，x, y 間のカーネルの値 $K(x,y)$ は $K(x,x), K(y,y)$ と比較すると，それほど大きくならないことが多い．これは，鋭いピークを持ったガウシアンと同様で，分布の中心から少しでも外れた場合の値が 0 に等しくなることと同じ現象である．つまりこれを，ガウシアンカーネル $K(x,y) = \exp(-||x-y||^2/2\sigma^2)$ のアナロジーで考えると，分散 σ^2 の非常に小さい大きなピークを持ったカーネルとなっているといえる．実際，式 (6) を変形すると

$$\hat{K}(x,y) = \exp\left(\log K(x,y) - \frac{1}{2}(\log K(x,x) + \log K(y,y))\right) \qquad (10)$$

のように，ガウシアンカーネルと同型の指数関数に書き直すことができる．

表 1 に具体例として，あるデータを与えた際の Gram Matrix を示す．この例からも，使用する構造のサイズ n を大きくしていくと，Gram Matrix は単位行列に徐々に近付いていくことが見てとれる．

テキストのような本来スパースな実データでは，更にこの傾向は顕著となり，サイズの大きい構造を扱う場合には，

表 1: 部分記号列のサイズの閾値 n を変化させた時のカーネルの値 $\hat{K}(x,y)$ の変化

S_1:ababb, S_2:acadbba, S_3:cbabdb, S_4:aba

n=1

	S_1	S_2	S_3	S_4
S_1	1.000	0.859	0.881	0.868
S_2		1.000	0.820	0.924
S_3			1.000	0.645
S_4				1.000

n=2

	S_1	S_2	S_3	S_4
S_1	1.000	0.709	0.765	0.817
S_2		1.000	0.778	0.785
S_3			1.000	0.593
S_4				1.000

n=3

	S_1	S_2	S_3	S_4
S_1	1.000	0.575	0.669	0.728
S_2		1.000	0.631	0.647
S_3			1.000	0.476
S_4				1.000

n=∞

	S_1	S_2	S_3	S_4
S_1	1.000	0.424	0.554	0.681
S_2		1.000	0.438	0.507
S_3			1.000	0.412
S_4				1.000

自分自身に対するカーネルの値以外はほぼ 0 と等しくなるといったことがしばしば起こる．つまり，学習データに出現したサンプルとほぼ同じか，或は，全く同じデータに対してしか信頼性の無い学習しかできない．

このように，Convolution Kernels を使用する際には，扱う構造のサイズを考慮する必要がある．これは，多項式カーネルの次数や，ガウシアンカーネルの分散のといったパラメタと同じことである．

3.1 従来手法

前節で示した問題に対応するための方法を述べる．これまで提案されているのは，主に，カーネルの値を平滑化させる方法と構造のサイズに基づく平滑化をおこなう方法である．

まず，最初に最終的なカーネルの値を平滑化させる方法について述べる．文献 [3] では，式 (10) で，$d^2(x,y) = \frac{1}{2}(\log K(x,x) + \log K(y,y)) - \log K(x,y))$ とおき，ガウシアンカーネルの分散と同形の平滑化の項を導入する方法を述べている．

$$\hat{K}(x,y) = \exp(-d^2(x,y)/\sigma^2) \tag{11}$$

しかし，この方法は，文献 [2] で実験的には実データに対してそれほど有効でないと述べられている．

次に，部分構造のサイズに基づく平滑化をおこなう方法について述べる．(拡張)String Kernel では，定義自体が既にこの問題に対応するように作成されており，部分構造のサイズ n 以下の部分記号列しか扱わない枠組がこれにあたる．つまり，全部分記号列中でサイズに対する閾値により素性選択を行なっていることに等しい．Tree Kernel の場合でも，例えば，木の階層数をパラメタとし，その階層数以上を扱わないようにする方法 [2] や，大きい部分木にはペナルティを与える等 [5] の方法を取っている．

このように，これまでの手法では，構造のサイズに依存して閾値を設ける方法が主流である．しかし，この素性選択基準は，仮に閾値 n 以上のサイズの部分構造の中に非常に有効な特徴が存在しても，それらはカーネルの値には反映されないという問題点を含んでいるためである．また，そのサイズの大きい部分構造をカーネルの値に反映させるために閾値の値を大きくしたとしても，データスパースネスの問題の影響で，逆に性能を劣化させることがほとんどである．このように，サイズに基づく素性選択手法は，適用は簡単であるが，仮にサイズの大きい部分構造の中に有効な特徴が存在することがわかっていても，サイズの閾値 n を大きくすることができないという問題がある．

そこで，部分構造のサイズに基づく粗い選択方法ではなく，個々の部分構造の有効性に基づいて素性を選択できる枠組が望まれる．

3.2 提案手法

本節では，前節までの議論を前提に，部分構造のサイズに依存しない素性選択手法として，String Kernel が計算する全ての部分記号列から，各部分記号列の統計量に基づいて分類に有効であると判定された部分記号列のみを素性として扱う方法を提案する．具体的には，全ての部分記号列に対して，全学習データ内での正例・負例間の出現頻度の偏りを検定し，出現傾向に偏りがみられる部分記号列のみを扱う．つまり，正例・負例間の出現頻度に偏りがある程，それぞれのクラスを特徴付ける素性として有効であると考える．

これにより，個々の部分記号列に対して統計量に基づく重要性のランクを与えることができるため，適切な閾値を与えることで，不要に素性数を増加させず，有効な素性のみを用いて学習・分類が可能となる．

表 2: カイ二乗値の計算

	C	$\bar{C}$	行合計
I	$O_{IC} = y$	$O_{I\bar{C}} = x - y$	$O_I = x$
$\bar{I}$	$O_{\bar{I}C} = M - y$	$O_{\bar{I}\bar{C}} = N - M - x + y$	$O_{\bar{I}} = N - x$
列合計	$O_C = M$	$O_C = N - M$	N

本稿では，パターンマイニング手法の一つである Apriori SMP[8] を参考に，カイ二乗 (χ^2) 値を用いて部分記号列の検定をおこなう．表 2 の分割表を用いて χ^2 値の計算方法を説明する．表中の C は正例のクラス，$\bar{C}$ は負例のクラスを表し，I は任意の部分記号列 I が出現したことを意味し，$\bar{I}$ は出現しない場合を意味する．O_{ij} を $i,j \in \{I,\bar{I}\} \times \{C,\bar{C}\}$ 両方の条件を満たした場合の観測値とする．例えば，O_{IC} は正例中の部分記号列が出現した数を意味する．同様に，O_i を $i \in \{I,\bar{I},C,\bar{C}\}$ の条件を満たした場合の観測値とする．例えば，O_C は正例数を意味する．次に，部分記号列 I が出現するデータ数を x，そのうち正例に出現するデータ数を y とし，同様に，サンプル数を N，正例数を M とおく．

部分記号列 I の χ^2 値 $\chi(I)$ を，$x(I), y(I)$ の関数 $chi(x(I), y(I))$ と考える．$x(I), y(I)$ は，それぞれ部分記号列 I に対する x, y の値である．χ^2 値は，$i,j \in \{I,\bar{I}\} \times \{C,\bar{C}\}$ のペアがそれぞれ独立であると仮定し

た場合の期待値 $E_{ij} = n \cdot \frac{O_i}{n} \cdot \frac{O_j}{n}$ と観測値 O_{ij} の正規化した偏差であり，以下のように書ける．

$$chi(x(I), y(I)) = \sum_{i \in \{I, \bar{I}\}, j \in \{C, \bar{C}\}} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad (12)$$

提案手法では，まず，χ^2 値の閾値 τ をあたえ，String Kernel を計算する際に出現した任意の部分記号列 u が $\tau \leq u$ なら通常通りカーネルの値に加算し，$u < \tau$ の場合は部分記号列 u の値を 0 と見做してカーネルの値を計算する．

このときの素性選択機能付き String Kernel の定義式は，式 (7) を変形し以下のようにかける．

$$K_{\mathrm{FSSK}}(S,T) = \sum_{\tau \leq \chi^2(u) | u \in \Sigma^*} \sum_{\mathbf{i} | u = S[\mathbf{i}]} \lambda^{\gamma(\mathbf{i})} \sum_{\mathbf{j} | u = T[\mathbf{j}]} \lambda^{\gamma(\mathbf{j})} \quad (13)$$

ここで，$\Sigma^* = \bigcup_{i=1}^{\infty} \Sigma^i$ とする．

以上，素性選択方法の基本的なアイディアを述べた．次節で，ここで述べた統計量を計算する枠組と String Kernel の枠組を効率的に結合する方法について述べる．

3.3 素性選択アルゴリズム

基本的には提案手法を用いた場合でも，通常の String Kernel と同様の手順で逐次計算をおこなっていく．その手順に，入力記号列間の共通記号が存在した際に，そこで計算された部分カーネルの値を最終的なカーネルの値に含めるかを χ^2 値で判定する部分が付加される．

まず，前準備として，ある部分記号列を u とし，u に対する χ^2 値は式 (12) から $\chi^2(u) = chi(x(u), y(u))$ である．次に，任意の部分記号列 v に対し，u を先頭に持つ全ての部分記号列 uv の χ^2 値の上限 $\chi^2_{\mathrm{up}}(u) = chi_{\mathrm{up}}(x(u), y(u))$ は

$$chi_{\mathrm{up}}(x(u), y(u)) = \max\left(chi(y(u), y(u)), chi(x(u) - y(u), 0)\right) \quad (14)$$

で与えられる．

そこで，u に対して，以下の 3 つの条件で場合分けをおこなう．

1. $\tau \leq chi(x(u), y(u))$
2. $\tau > chi(x(u), y(u))$，かつ，$\tau \leq chi_{\mathrm{up}}(x(u), y(u))$
3. $\tau > chi(x(u), y(u))$，かつ，$\tau > chi_{\mathrm{up}}(x(u), y(u))$

このとき，1. の場合には u の χ^2 値は閾値 τ 以上なので，素性として採用し，最終的なカーネルの値に加算される．2. の場合は，u の χ^2 値は閾値 τ 未満なので，最終的なカーネルの値には加算されないが，uv の χ^2 値が閾値 τ を上回る可能性があるので，内部的には計算値を保持しておく．3. の場合には，u と uv の χ^2 値は閾値 τ 未満なので，対象の部分カーネルの値を 0 とし，以降の計算から削除する．

以上の処理を式で書くと以下のようになる．

$$K_{\mathrm{FSSK}}(S,T) = \sum_{m=1}^{n} \sum_{1 \leq i \leq |S|} \sum_{1 \leq j \leq |T|} K_m(S_i, T_j) \quad (15)$$

$$K_m(S_i, T_j) = \sum_{u \in U_m(S_i, T_j)} K_{m,u}(S_i, T_j) \quad (16)$$

$U_m(S_i, T_j)$ は，部分記号列の集合であり，前述の条件 1. を満たす $w | s_i = t_j = w$ を最後の要素とする任意の部分属性列 uw で構成される．ここで w を任意の記号とする．

$$K_{m,uw}(S_i, T_j) = \begin{cases} K''_{m-1,u}(S_i, T_j), \\ \quad \text{if } s_i = t_j = w \text{ and } \tau \leq \chi_{\mathrm{up}}(uw) \\ 0, \quad \text{otherwise} \end{cases} \quad (17)$$

式 (17) で，条件 2. を実現している．つまり，$\tau \leq \chi_{\mathrm{up}}(uw)$ を満たす部分記号列のみ uw を計算し，それ以外は 0 とする．

$$K'_{m,u}(S_i, T_j) = \begin{cases} 0 \quad \text{if} \quad i = 0 \text{ or } j = 0, \\ \lambda K'_{m,u}(S_{i-1}, T_j) + K_{m,u}(S_{i-1}, T_j) \\ \qquad \text{otherwise} \end{cases} \quad (18)$$

$$K''_{m,u}(S_i, T_j) = \begin{cases} 0 \quad \text{if} \quad i = 0 \text{ or } j = 0, \\ 1 \quad \text{if} \quad m = 0 \\ \lambda K''_{m,u}(S_i, T_{j-1}) + K'_{m,u}(S_i, T_{j-1}) \\ \qquad \text{otherwise} \end{cases} \quad (19)$$

つぎに，条件 1. の部分記号列集合を得るために必要な演算を定義する．

$$U_m(S_i, T_j) = \begin{cases} \bigcup_{u \in U''_{m-1}(S_i, T_j)} \Psi(uw), \\ \qquad \text{if} \quad s_i = t_j = w \\ \emptyset, \quad \text{otherwise} \end{cases} \quad (20)$$

$$\Psi(u) = \begin{cases} u \quad \text{if} \quad \tau \leq \chi^2(u) \\ \emptyset, \quad \text{otherwise} \end{cases} \quad (21)$$

$$U'_m(S_i, T_j) = \begin{cases} \emptyset \quad \text{if} \quad i = 0 \text{ or } j = 0, \\ U'_m(S_{i-1}, T_j) \cup U_m(S_{i-1}, T_j) \\ \qquad \text{otherwise} \end{cases} \quad (22)$$

$$U''_m(S_i, T_j) = \begin{cases} \emptyset \quad \text{if} \quad i = 0 \text{ or } j = 0 \text{ or } m = 0, \\ U''_m(S_i, T_{j-1}) \cup U'_m(S_i, T_{j-1}) \\ \qquad \text{otherwise} \end{cases} \quad (23)$$

実際には，図 1 と同様の 2 次元テーブルを用いて容易に計算することができる．

次に，χ^2 値を計算するために必要な u の出現数 $x(u), y(u)$ を効率的に得る方法について述べる．基本的なアイディアは文献 [9] に基づいて，パターンマイニング手法の一つである PrefixSpan[10] のアルゴリズムを活用する．ただし，通常の PrefixSpan の頻度に基づく閾値により抽出パターンの制約を与えている方法とは異なり，ここでは，前述の Apriori SMP の方法に従って χ^2 値に基づいて抽出するパターンに制約を与える．

具体的には，式 (17) で χ^2 を計算する際に，PrefixSpan のアルゴリズムを用いて全学習データに対する共通部分記号列 uw の出現数を得る．このとき，u までの PrefixSpan の過程は事前の計算で得られているので，実際には，u から uw の部分記号列を抽出する計算量だけで計算することが可能である．さらに，実装上では PrefixSpan により得られた部分記号列を Trie 状のキャッシュに保持しておくことで，以前に出現した同一の部分記号列の判定を高速におこなうことが可能である．

4 実験

実データを用いて素性選択機能付き String Kernel (FSSK)，拡張 String Kernel (ESK)，BOW Kernel (BOW-K)[11] の比較実験をおこなった．BOW-K は，出現単語

(属性) を素性ベクトルとした多項式カーネルのことであり，ベースラインの手法として用いた．ESK 及び FSSK でも同様に扱う記号の単位は単語とし，ESK は式 (8), FSSK は前節のアルゴリズムを用いてカーネルの値を計算した．

前提条件である二クラスの学習・分類タスクとして，文のモダリティ判定タスク (実験 1)，質問分類タスク (実験 2) を用いた比較実験をおこなった．学習器には SVM を用い，各手法の評価尺度には，5fold の交差検定により得られた F 値の平均を用いた．

表 3 に SVM およびカーネルのパラメタを示す．

表 3: SVM 及びカーネルのパラメタ

パラメタ	実験 1	実験 2
ソフトマージン C	1000	
減衰率 λ	0.5	
χ^2 の閾値 (α)	2.7055(0.1), 3.8415(0.05)	
頻度の閾値	5	10

提案手法 FSSK では，素性選択の閾値パラメタの違いによる 3 つの方法で実験をおこなった．χ^2 分布の自由度 1 の 10%有意水準での棄却点 2.7055 を閾値にした場合を FSSK1，5%有意水準での棄却点 3.8415 を閾値にした場合を FSSK2，χ^2 値ではなく，部分記号列の頻度を閾値として素性選択をおこなった場合を FSSK3 とする．

4.1 実験 1：文のモダリティ

被験者 1 名が田村ら [12] の大分類に従い文のモダリティを決定した 99 年毎日新聞の社説からランダムに選んだ 60 記事を実験データをもちいた．田村らが用いたモダリティの大分類は，「意見」，「断定」，「叙述」であり，データ数はそれぞれ 123，326，686 文である．

まず，前処理として，実験データを chasen[*2]を用いて形態素解析し，その後，磯崎の手法 [13] を用いて固有表現抽出をおこなった．基本的には形態素の表層形を一つの記号として扱うが，固有表現が抽出された場合には，形態素の区切りを固有表現の区切りで置き換えて使用した．この解析済テキストを入力データとした．

図 2 に，実際に実験に用いたデータの例を示す．図中，括弧内が一つの記号に相当する．

4.2 実験 1 の結果

表 4 に文のモダリティ判定の実験結果を示す．実験結果は，(a)「意見」とそれ以外，(b)「断定」とそれ以外，(c)「叙述」とそれ以外の二クラス分類を行なった結果である．

4.3 実験 2：質問分類

実験用データには，NTCIR-QAC1[*3]の 1011 問，CRL-QA データ [*4]の 2000 問，NTT2000 の 2000 問を合わせた計 5011 問を用いた．また，このデータには，CRL-QA データの分類に基づいた 150 の質問タイプが付与されている．

実験には，与えられた質問タイプから代表的な質問タイプ LOCATION，NUMEX，ORGANIZATION，PERSON，TIME_TOP の 5 つを選出した．このとき，CRL-QA データでは質問タイプが階層的に定義されているので，対象となる質問タイプの下位に位置の下位に位置する質問タイプも対象質問タイプとして扱った．このとき，それぞれの質問タイプの質問数は，LOCATION 752 問，NUMEX 882 問，ORGANIZATION 733 問，PRESON 824 問，TIME_TOP 779 問であった．

前処理として，実験 1 と同様に形態素解析と固有表現抽出をおこなった．固有表現が抽出された場合には，形態素の区切りを固有表現の区切りで置き換えて使用した．質問分類では，実験 1 と異なり，属性として単語の表層ではなく，単語の基本形，意味情報，固有表現タグを用いた．意味情報は，日本語語彙大系の意味カテゴリ [14] を使用した．

図 3 に，実際に実験に用いたデータの例を示す．図中で，括弧内のカンマで区切られた一つ一つが記号となる．つまり，ここでは記号セットを入力とした String Kernel を用いている．

4.4 実験 2 の実験結果

表 5 に質問分類の実験結果を示す．実験結果は，(a)LOCATION とそれ以外，(b)NUMEX とそれ以外，(c)ORGANIZATION とそれ以外，(d)PERSON とそれ以外，(e)TIME_TOP とそれ以外の二クラス分類を行なった結果である．

5 考察

5.1 String Kernel の有効性

まず，それぞれの実験で，BOW-K と ESK の結果を比較すると，ほぼ全ての実験で ESK の方がはるかに良い結果を得ている．この結果から，本稿で扱ったタスクでは，部分記号列である単語の並びが有効な素性となっていることがわかる．例えば，文のモダリティ判別タスクでは，文末表現が有効な特徴であると考えられ，質問分類では，疑問詞とその前後の単語から成る表現が有効な特徴であると考えられる．

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

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

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

クラス	入力オブジェクト（記号列）
意見	(議員)(立法)(で)(短期)(決着)(を)(目指す)(テーマ)(と)(は)(思わ)(ない)(．)
断定	(２１世紀)(の)(アジア)(に)(は)(，)(覇権)(国家)(の)(出現)(を)(許さ)(ない)(仕組み)(が)(必要)(だ)(．)
叙述	(暗証)(番号)(で)(アクセス)(し)(情報)(を)(伝達)(する)(．)

図 2: 文のモダリティ判定タスクの実験データの例

表 4: 文のモダリティの分類実験の結果

(a) 意見

n	1	2	3	4	∞
FSSK1	-	.734	.743	.746	.751
FSSK2	-	.740	.748	.750	.750
FSSK3	-	.735	.751	.749	.747
ESK	-	.706	.672	.577	.058
BOW-K	.507	.531	.438	.368	-

(b) 断定

n	1	2	3	4	∞
FSSK1	-	.828	.858	.854	.857
FSSK2	-	.824	.855	.859	.860
FSSK3	-	.832	.853	.861	.857
ESK	-	.816	.834	.830	.339
BOW-K	.652	.708	.686	.665	-

(c) 叙述

n	1	2	3	4	∞
FSSK1	-	.896	.906	.910	.910
FSSK2	-	.894	.903	.909	.909
FSSK3	-	.896	.910	.909	.911
ESK	-	.902	.913	.910	.808
BOW-K	.819	.839	.826	.793	-

5.2 ESK の部分記号列のサイズの閾値に関する分類精度

ESK では，全ての実験で部分記号列のサイズの閾値 n が 2 或いは 3 の場合に分類性能が最大となり，4 では分類性能が急激に劣化する傾向が見られる．これは，n を大きくすると，本稿で問題として取り上げたデータスパースネスの問題が起こっていることを示している．また，全ての部分記号列 $(n = \infty)$ をカーネル計算に使用すると，分類精度は最高値の半分以下か悪くすると 1 割程度まで悪化することもあった．このとき，適合率は非常に高くなるが，逆に再現率は非常に低くなるといった現象が起こっている．表 6 に本稿の実験での実際の再現率・適合率の例を示す．

表 6: $n = \infty$ の時の再現率と適合率

	再現率	適合率	F 値
実験 1：意見	.917	.209	.339
実験 2：LOCATION	.896	.093	.168

これは，学習データに含まれていた正例と非常に似たデータのみ正しく正例と判定し，それ以外は全て負例と判定するような分類器を学習してたことになる．学習データに含まれていた正例と非常に似たデータ以外を全て負例と分類されるのは，データの比率が正例に対して負例が多いデータであるので，デフォルトの分類規則が負例となるような分類器が学習されているためである．

このように，テキストの場合は $n = 2$ あたりが最も良い性能を出す閾値となる．これは，単語を記号としていることから $n = 2$ で既に素性空間の次元数は非常に高くなっており，部分構造のサイズに基づく素性選択手法では，$n = 2$ あたりが限界であることを示している．

5.3 $n = 2$ の場合の素性選択の性能

ここではサイズが 2 以下の部分記号列を扱った実験結果 $(n = 2)$ に焦点をあてる．この場合，ESK に比べ提案手法の概ね良い結果を得ている．

この結果は，部分構造のサイズによる素性選択では，分類に悪影響を与える素性を排除できていないことを意味し，逆に，統計量に基づく素性選択をおこなうことで，悪影響を与える素性を排除できていることを示している．

5.4 提案手法の有効性

提案手法を用いることにより，扱う部分記号列のサイズを大きくしても，或いは，全ての部分記号列を扱う場合でも，その中から有効な素性を選択することで，従来手法より良い分類精度が得られた．

サイズに基づく素性選択手法のように全ての部分記号列を扱う場合 $(n = \infty)$ でも，分類精度は大幅に劣化せず，むしろ向上することもあった．これは，サイズが大きい部分構造にも分類に有効な素性が存在したことを意味している．ある n の素性空間は，それより n が大きい素性空間の部分空間であるため，n が大きい方が素性空間の次元数は必ず高くなっている．それにも関わらず分類性能が向上していることは，n を増加した影響で分類に有効な素性が加わったことを示していからである．

特に，$n = 4$ から $n = \infty$ に変更したことで，性能が向上した場合は，その分類には $n = 5$ 以上の特徴が分類に有効であったことを示唆している．

このように，使用する分類問題によって違いはあるものの，この結果は，サイズの大きい部分構造が分類の特徴に有効である可能性を示している．よって，学習データのサイズにも依存するが，基本的には，全ての部分構造から提案手法を用いて有効な素性を選択する方法が理想的な方法であると考えられる．

5.5 統計量の閾値の違いによる分類精度

提案手法を用いる際に最も重要なことは，統計量の閾値をいくつに設定するかである．これにより，分類性能は大きく変わってくる．実験結果からもわかるように，最適な閾値は扱う問題にも依存するし，扱う部分構造のサイズによっても最適な値は違う．実験によって FSSK1 が良い場合と FSSK2 が良い場合があることがこれを示している．

本実験では，χ^2 検定の考えに基づいて，独立性の仮説の

クラス	入力オブジェクト（記号列）
LOCATION	(９９年,DATE)(に)(:サミット,[会議])(が)(開く)(れる)(た)(の)(は)(どこ,[地域:人間活動],[位置])(です)(か)
ORGANIZATION	(９９年６月,DATE)(に)(破綻,[不成立],[失敗],[貧],[絶交])(する)(た)(生保,[財政])(は) (どこ,[地域:人間活動],[位置])(です)(か)
PERSON	(日本自動車工業会,ORGANIZATION)(会長,[役員],[長],[人:管理職])(は)(誰,[不定称])(です)(か)

図 3: 質問分類タスクの実験データの例

表 5: 質問分類の実験結果

(a) LOCATION

n	1	2	3	4	∞
FSSK1	-	.795	.793	.798	.792
FSSK2	-	.788	.799	.804	.800
FSSK3	-	.792	.788	.798	.792
ESK	-	.791	.775	.732	.169
BOW-K	.744	.768	.756	.747	-

(b) NUMEX

n	1	2	3	4	∞
FSSK1	-	.912	.915	.908	.908
FSSK2	-	.913	.916	.911	.913
FSSK3	-	.912	.909	.909	.905
ESK	-	.912	.885	.817	.036
BOW-K	.842	.852	.807	.726	-

(b) ORGANIZATION

n	1	2	3	4	∞
FSSK1	-	.709	.720	.720	.723
FSSK2	-	.703	.710	.716	.720
FSSK3	-	.718	.717	.713	.717
ESK	-	.705	.668	.594	.035
BOW-K	.641	690	.636	.572	-

(d) PERSON

n	1	2	3	4	∞
FSSK1	-	.954	.956	.955	.955
FSSK2	-	.954	.955	.951	.954
FSSK3	-	.952	.954	.956	.957
ESK	-	.949	.937	.914	.311
BOW-K	.950	.952	.941	.928	-

(e) TIME_TOP

n	1	2	3	4	∞
FSSK1	-	.961	.958	.957	.956
FSSK2	-	.961	.956	.957	.956
FSSK3	-	.953	.953	.955	.955
ESK	-	.946	.910	.866	.223
BOW-K	.902	.909	.886	.855	-

棄却域 10%と 5%に相当する閾値 χ^2 値を用いた．ただし，これは最適な値というわけではなく，実際には最適な値は実験的に求める必要がある．

しかし，逆に考えると，本稿で示した実験結果は，最適な閾値を求めることにより分類精度が向上する可能性がまだ残されていることを意味する．

6 まとめ

本稿では，従来の String Kernel の素性選択手法である部分構造のサイズに基づく方法の問題点を取り上げ，その問題点を解決する一つの方法として，統計量を基準とした素性選択手法を提案した．

質問分類と文のモダリティ判定の二つのタスクを用いて，従来手法の問題点を検証するとともに，従来手法との比較によって提案手法の有効性を示した．

参考文献

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

[2] Collins, M. and Duffy, N.: Convolution Kernels for Natural Language, *Proc. of Neural Information Processing Systems (NIPS'2001)* (2001).

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

[4] Watkins, C.: Dynamic Alignment Kernels, *Technical Report CSD-TR-98-11*, Royal Holloway, University of London Computer Science Department (1999).

[5] Collins, M. and Duffy, N.: New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron, *Proc. of ACL'2002*, pp. 263–270 (2002).

[6] Cancedda, N., Gaussier, E., Goutte, C. and Renders, J.-M.: Word-Sequence Kernels, *Journal of Machine Leraning Research*, Vol. 3, pp. 1059–1082 (2003).

[7] Cortes, C. and Vapnik, V. N.: Support Vector Networks, *Machine Learning*, Vol. 20, pp. 273–297 (1995).

[8] Morishita, S. and Sese, J.: Traversing Itemset Lattices with Statistical Metric Pruning, *Proc. of ACM SIGACT-SIGMOD-SIGART Symp. on Database Systems (PODS'00)*, pp. 226–236 (2000).

[9] Kudo, T. and Matsumoto, Y.: Fast Methods for Kernel-based Text Analysis, *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pp. 24–31 (2003).

[10] Pei, J., Han, J., Mortazavi-Asl, B. and Pinto, H.: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, *Proc. of the 17th International Conference on Data Engineering (ICDE 2001)*, pp. 215–224 (2001).

[11] Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features, *Proc. of European Conference on Machine Learning(ECML '98)*, pp. 137–142 (1998).

[12] 田村直良, 和田啓二: セグメントの分割と統合による文章の構造解析, 自然言語処理, Vol. 5, No. 1, pp. 59–78 (1996).

[13] Isozaki, H. and Kazawa, H.: Efficient Support Vector Classifiers for Named Entity Recognition, *Proc. of the 19th International Conference on Computational Linguistics (COLING 2002)*, pp. 390–396 (2002).

[14] 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).