固有表現抽出のためのSVMの高速化

磯崎秀樹[†] 賀沢秀人[†]

サポートベクトルマシン（SVM）は新しい高性能な学習手法である．しかし，従来手法より分類処理速度が桁違いに遅いことが知られている．本論文では，まずSVMを用いた固有表現抽出方法が既存手法より高精度であることを実験により示す．固有表現抽出は，地名・人名・組織名・日時などの固有表現を文書から抜き出す技術であり，情報抽出システムや質問応答システムなどの重要な基礎技術である．次に，固有表現抽出のデータの特徴を生かして，処理速度を大幅に改善するアルゴリズムを提案する．このアルゴリズムは，自然言語処理における他の様々なタスクに応用可能であると考えられる．

Speeding up Support Vector Machines for Named Entity Recognition

Hideki Isozaki[†] and Hideto Kazawa

The Support Vector Machine (SVM) is a powerful new machine learning method. However, it is well known that its classification speed is orders-of-magnitude slower than conventional systems. First, we show that a Named Entity (NE) recognizer based on SVMs gives better scores than conventional systems. Named Entity recognition is a task in which proper nouns and numerical information are extracted from documents and are classified into categories such as person, organization, and date. It is a key technology of Information Extraction and Open-Domain Question Answering. Then, we present an algorithm that makes the system substantially faster by exploiting characters of NE data. This algorithm will be applicable to other different tasks in Natural Language Processing.

1. はじめに

最近,「サポートベクトルマシン（SVM）」という高性能な機械学習手法が注目を集めている．SVMの特長は「マージン最大化」による高い汎化能力と「カーネル関数」を用いた「カーネルトリック」による非線形境界の効率的な処理である．

しかし，学習や処理に膨大な時間がかかるため，なかなか利用されていないのが実情である．本論文では,「固有表現抽出」という自然言語処理の1つのタスクを取り上げ，処理速度を大幅に改善できるアルゴリズムを提案する．本手法は固有表現抽出にとどまらず,様々な自然言語処理タスクの高速化に利用できると考えられる．

固有表現抽出とは，文書に書かれている情報の重要な要素である,「人名」「地名」「組織名」「日時」などの表現を抜き出すタスクであり，TREC-QA☆に代表されるオープンドメイン質問応答システム（ODQA）や情報抽出システムにおいて重要な技術である．日本語でも，ODQAの研究が最近行われており[1),29),34)~36)]，NTCIR-QAC☆☆が開催されている．また，イベントトラッキング[15)]，重要文抽出[10),11)]，大量文書のブラウジング[17)]などにおいても，固有表現抽出は有用である．

固有表現抽出システムは，**図1**のように，与えられた文章のどこからどこまでが固有表現か，そしてその種類は何かを判定する．情報検索と情報抽出を対象にしたIREX（Information Retrieval and Extraction Exercise☆☆☆）というワークショップでは，8つの固有表現が定義された．

固有表現抽出は，辞書を用意するだけで簡単に解決しそうだが，実際には辞書を作るだけでは不十分であり，前後の文脈を含めた判断が必要になることが多い．たとえば,「中野」のように人名でも地名でも使われ

† 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
NTT Communication Science Laboratories Nippon Telegraph and Telephone Corporation

☆ http://trec.nist.gov/presentations/TREC10/qa/
☆☆ http://www.nlp.cs.ritsumei.ac.jp/qac/
☆☆☆ http://nlp.cs.nyu.edu/irex/index-j.html

−入力−
1 日夕方 5 時，田中電機の田中純一郎社長は東京・大手町で記者会見を開いた．
−出力−
<DATE>1 日</DATE><TIME>夕方 5 時</TIME>，
<ORGANIZATION>田中電機</ORGANIZATION>の
<PERSON>田中純一郎</PERSON>社長は
<LOCATION>東京・大手町</LOCATION>で記者会見を開いた．

図 1 固有表現抽出の例
Fig. 1 An example of named entity recognition.

る単語や，「共同」「トンボ」「ゆず」のように，一般名詞か固有名詞か紛らわしい単語は少なくない．これらを正しく分類するには，前後の文脈を考慮する必要がある．「後ろに『さん』や『氏』などがあれば人名」というようなルールを人手で書いていく方法も使われるが，ある程度以上に精度を上げるのは，きわめて困難である．

これまでに，人手で辞書やルールを作成する方法[12),37)]，隠れマルコフモデル[3)]や最大エントロピーモデル[4),38)]などの統計的手法，ルール自動生成による手法[13),30),33)]，統計的手法とルール自動生成の出力の混合[43)]などが提案されている．

本論文では，サポートベクトルマシン（SVM）[8)]を用いる．SVM を用いた固有表現抽出はすでに山田ら[40)]により提案されているが，山田らは同一データで他の手法との比較を行っていないため，彼らの手法が他の手法より優れているのかどうか明確でなかった．本論文では，山田らの手法とは異なる手法で実装を行い，同一データで比較実験を行うことで，従来手法より好成績が達成できることを示す．

しかし，SVM は分類処理の速度が遅いという問題がある．山田らの実験[41)]によれば，Pentium III 933 MHz の計算機で 1,880 文を処理するのに 40 分と報告されている．つまり 0.8 文/秒という遅さである．我々の実験でも，Athlon 1.3 GHz の計算機で 92 バイト/秒と，従来手法に比べ，桁違いに遅いことが確認された．磯崎[13)]のルール生成法によるシステムは 7.9 キロバイト/秒なので，86 倍遅い．92 バイト/秒の処理速度では，現在購入可能な 11 年分の毎日新聞の記事を処理するのに 6 カ月かかってしまう計算になる．SVM の遅さは固有表現抽出に限らない．たとえば，Nakagawa らの品詞タグ付けプログラム[22)]は Alpha 21164A 500 MHz で 20 トークン/秒である．TreeTagger☆は SPARC10 でも 1 万トークン/秒なので，桁違いに遅いことが分かる．これらの報告のように，自然言語処理における SVM の遅さは次第に認識されてきている．そこで本論文では，分類にかかる時間を大幅に削減できる方法を提案する．

☆ http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

2. SVM を用いた固有表現抽出

ここでは，最大エントロピーモデルに基づく固有表現抽出システム[4),13),38)]の中の最大エントロピーモデルを最小限の変更で SVM に置き換えたシステムの実装について述べる．このシステムでは，固有表現抽出を形態素の分類問題としてとらえる．1 つの固有表現は 2 つ以上の形態素からなることがあるので，各形態素を IREX 固有表現の 8 種類に対して先頭，途中，末尾，単独の 4 種類に分け，合計 $4 \times 8 + 1 = 33$ 種類に分類する．たとえば，「そこで，大阪中央銀行の」の場合，「大阪」が「組織名の先頭」で「中央」が「組織名の途中」，「銀行」が「組織名の末尾」，「そこで」「，」「の」が「固有表現以外」となる．「日銀総裁」の場合「日銀」が「組織名単独」で「総裁」が「固有表現以外」となる．

SVM でこれらの単語を分類していけばいいのだが，各単語を独立に分類することはできない．たとえば，「組織名の先頭」の次には「組織名の途中」か「組織名の末尾」しか来ることができない．このような制約を満たすすべての組合せの中で，最ももっともらしい分類を見つける必要がある．そこで，Viterbi アルゴリズム（たとえば文献 16) の p.105）を用いて，文全体でベストになる組合せを選ぶ．

さて，SVM は実数ベクトル $\mathbf{x}$ を 2 つのクラスに分類するための機械学習手法であり，学習データを $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ で表す．ただし，y_i は $\mathbf{x}_i$ が正例であれば $+1$，負例であれば -1 をとる．SVM は，以下の判別関数の符号により 2 クラス（正例，負例）の分類を行う．

$$g(\mathbf{x}) = \sum_{j=1}^{N} y_j \alpha_j K(\mathbf{x}, \mathbf{x}_j) + b.$$

ここで α_j と b は学習によって決まる定数である．SVM の学習は，次の式を最大化する問題で表せる[31)]．

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

ただし

$$0 \leq \alpha_i \leq C, i = 1, \ldots, N, \text{ かつ } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

ここで，C はマージン最大化と分類誤りのバランスを調整するためのパラメータである．$K(\mathbf{x}, \mathbf{x}_j)$ はカーネル関数と呼ばれ，言語処理では d 次の多項式 $K(\mathbf{x}, \mathbf{x}_j) = (1 + \mathbf{x} \cdot \mathbf{x}_j)^d$ がよく利用される[19),22),39)]．詳細は文献 8)，31) を参照されたい．

関数 $g(\mathbf{x})$ において，$\alpha_j = 0$ の $\mathbf{x}_j$ は結果に寄与しないので，$\alpha_j = 0$ を除くと，次のように書き換えることができる．

$$g(\mathbf{x}) = \sum_{i=1}^{S} w_i K(\mathbf{x}, \mathbf{z}_i) + b. \qquad (1)$$

ここで w_i，$\mathbf{z}_i$ はいずれかの $y_j \alpha_j$，$\mathbf{x}_j$ に対応する．$\mathbf{z}_i$ をサポートベクトルと呼ぶ．S $(\leq N)$ はサポートベクトルの数である．

前述のように，固有表現抽出では 33 クラスへの分類が必要であるが，SVM は 2 クラスの分類しかできない．そこで SVM を複数組み合わせて使う．one-versus-rest，pairwise[18)]，2 分木[32)]，2 分 DAG[26)]，誤り訂正符号[2)] などの方法が利用されているが，本論文では，各クラスごとに分類器を用意する，いわゆる one-versus-rest を採用する．

また，Viterbi アルゴリズムではコストが必要となり，通常，コストとして確率が用いられるが，SVM は確率を出力できないので，$g(\mathbf{x})$ の値にシグモイド関数を適用することにより，確率類似量を計算する．ただし，データに合うようにシグモイド関数のパラメータを決定する[25)] と処理が煩雑になるので，ここでは全クラスに同じシグモイド関数 $s(x) = 1/(1 + \exp(-\beta x))$ を用いる．つまり，まず 33 クラスの各クラス c に対して，c か否かを判定する分類器 $g_c(\mathbf{x})$ を作り，次に $s(g_c(\mathbf{x}))$ により出力を 0〜1 の値に正規化してから Viterbi アルゴリズムを適用する．このように，前後の文脈を見て単語の分類を決定しているので，厳密な意味での one-versus-rest ではない．

なお，SVM を適用するにあたり，ルールベースの前処理・後処理[13)] を利用した．以下はその説明である．

分類の際の素性としては，分類対象の形態素とその前後 2 つずつの計 5 形態素に対し，出現形，ChaSen 2.2.1[21)] による品詞，形態素を構成する文字種という 3 種類の素性を用いる．つまり，5×3 で計 15 素性となる．文字種は 17 種類に分類した．数字は「小数」「1 万未満の正の整数」「1 万以上の整数」「ゼロから始まる 2 桁以上の整数」に分けた．それ以外の形態素はカタカナのみ，漢字のみ，ひらがなのみ，大文字のみ，小文字のみ，キャピタライズ（大文字 1 文字＋小文字列），その他の英数，ギリシャ文字，記号，「その他」とし，漢字 1 文字，大文字 1 文字，カタカナ 1 文字の場合をさらに分けた．なお，ChaSen は数字を 1 文字ずつに分割してしまうので，連続する数字をまとめて 1 形態素にしている．また，「3 月」などの形態素は「3」と「月」に分けている．また，カタカナ未知語の過分割を避けるため，未知語コストを 7000 に下げた．さらに，未知語はサ変名詞ではなく固有名詞一般として出力されるようにしている．詳細は磯崎[13)] を参照されたい．

表 1 形態素を分類するための素性
Table 1 Features for classification of morphemes.

2 つ前の形態素	=「昨日」
2 つ前の品詞	= 副詞可能名詞
2 つ前の文字種	= 複数漢字
直前の形態素	=「，」
直前の品詞	= 読点
直前の文字種	= 記号
中心の形態素	=「ビレンドラ」
中心の品詞	= 固有名詞一般
中心の文字種	= 複数カタカナ
直後の形態素	=「国王」
直後の品詞	= 一般名詞
直後の文字種	= 複数漢字
2 つ後の形態素	=「は」
2 つ後の品詞	= 係助詞
2 つ後の文字種	= ひらがな

以上の素性を 0/1 の 2 値で表現し，これらを成分に持つベクトルに変換して SVM で学習を行う．たとえば，「昨日，ビレンドラ国王は …」という文の「ビレンドラ」という未知語は，**表 1** の 15 個の素性で表される．SVM で扱えるようにするため，これらの素性を**表 2** のような 0/1 のバイナリベクトルに変換する．

形態素そのものが素性に含まれているため，ベクトルの次元は万のオーダになる．しかし，元の素性が 15 個しかないので，値が 1 の成分は 15 個だけであることに注意されたい．

後処理[13)] は，学習データにおける固有表現抽出の境界の変更を行う．たとえば，ChaSen は「奈良市内」を「奈良」と「市内」に分割するが，学習データでは「奈良市」が地名とされているという問題がある．そこでこの問題に対処するため，以下のような後処理ルールを自動生成する．まず，固有表現の境界が形態素の途中にある場合，その形態素が固有表現の中に完全に含まれるようにタグ位置を仮想的にずらす．これで境界の問題はなくなったので，形態素を分類する問題として上記の学習を行う．固有表現の境界を含んでいた形態素は単独・先頭・末尾のいずれかに分類されるので，そのクラスとずらした文字数を記録しておく．対象文書に学習結果を適用し，Viterbi アルゴリズムで

表 2 形態素の素性を表すバイナリベクトル
Table 2 A binary vector that represents features of a morpheme.

2 つ前の形態素が「昨日」	= 1
2 つ前の形態素が「，」	= 0
2 つ前の形態素が「ビレンドラ」	= 0
2 つ前の形態素が「国王」	= 0
2 つ前の形態素が「は」	= 0
：	
2 つ前の品詞が副詞可能名詞	= 1
2 つ前の品詞が読点	= 0
2 つ前の品詞が固有名詞一般	= 0
2 つ前の品詞が一般名詞	= 0
2 つ前の品詞が係助詞	= 0
：	
2 つ前の文字種が複数漢字	= 1
2 つ前の文字種が記号	= 0
2 つ前の文字種が複数カタカナ	= 0
2 つ前の文字種がひらがな	= 0
：	
直前の形態素が「昨日」	= 0
直前の形態素が「，」	= 1
直前の形態素が「ビレンドラ」	= 0
直前の形態素が「国王」	= 0
直前の形態素が「は」	= 0
：	

最良解が得られると，単独・最初・末尾に属する各形態素をさきほどの記録と照らし合わせてタグを移動する．ただし，「米国」などの一部の形態素は，形態素解析の失敗などにより，固有名詞の境界を含んだり含まなかったりする．そこで，学習データでこの書き換えの正解率を調べておき，50%以下のものは適用しない．

ちなみに山田らは，単独と最初，途中と末尾を区別せず，$2 \times 8 + 1 = 17$ 種類に分類している．また，分類の決定においては，Viterbi アルゴリズムを使わず，文の先頭あるいは末尾から逐次的に決定している．文字種は 6 種類で，形態素境界の後処理方法も異なる．

3. 従来手法との成績の比較

学習用のデータとしては，IREX で作成された CRL 固有表現データ 1,174 記事☆を用いる．さらに，データが増えた場合の挙動を調べるため，分野特定課題用学習データ 23 記事と佐々木の拡張タグデータ[28] を IREX の最終的な定義に即して修正した 4,890 記事も用いる．いずれのデータも毎日新聞 CD-ROM の 94，95 年版に基づいている．採点には IREX の GENERAL（一般課題）という公式テストデータ（99 年の記事）に対する F 値を用いる．ここでいう F 値とは，システムの出力した固有表現のうち，文書中に出現した箇所

表 3 パラメータ C，d による成績の違い（$\beta = 5$）
Table 3 Difference in scores with parameters C, d.

$d \backslash C$	0.0001	0.001	0.01	0.1	1	10
1	—	65.22	73.41	81.18	82.96	80.57
2	—	76.66	84.43	**85.10**	**85.10**	**85.10**
3	78.68	83.79	83.79	83.79	83.79	83.79

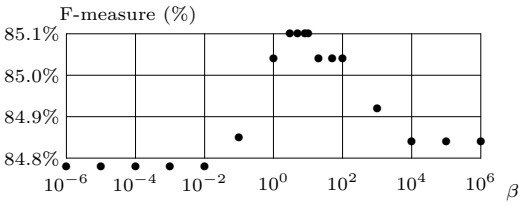

図 2 シグモイド関数の傾き β による成績の変化
Fig. 2 Difference in scores with a parameter β.

の左右の境界の位置と固有表現の分類が，正解と完全に一致するものの数を a としたとき，以下で定義される再現率と適合率の調和平均である．

- 再現率 $= a$/正解データにおける NE の数
- 適合率 $= a$/システムの出力した NE の数

学習には，工藤の TinySVM-0.07☆☆を用いた．

SVM による学習では，カーネル関数の選択と C の設定が問題となる．CRL データを学習データとしたときの $\beta = 5$ における GENERAL の F 値は，**表 3** に示すとおりであり，これによれば $d = 2$ の成績が一番良い（山田らの実験でも 2 次が一番良いと報告されている）．表によると，$d = 2$ のとき，C は 0.1 以上で成績に差がなく，C が大きいほど学習に時間がかかるので，以下の実験ではすべて $d = 2$，$C = 0.1$ を用いる．

図 2 に示すように，β の値は成績にあまり影響しない．図 2 の左右両端はフラットになっているが，これは次のような理由による．$|\beta x| \ll 1$ のとき，$\log s(x) = \beta x/2 - \log 2$ で近似できる．Viterbi アルゴリズムは文中の各形態素の $\log s(g_c(\mathbf{x}))$ の和を最大化するように働くので，$\beta \ll 1$ のときに Viterbi アルゴリズムで得られる最良なクラスの組合せ（解）は，$\log s(x) = x$ の場合の解に近づいていく．一方，$|\beta x| \gg 1$ のとき，$\log s(x) = \beta(x)_-$ で近似できる．ここで $(x)_-$ は $x < 0$ のとき x で，$x \geq 0$ のとき 0 の関数である．したがって，$\beta \gg 1$ のときに Viterbi アルゴリズムで得られる解は，$\log s(x) = (x)_-$ の場合の解に近づいていく．

SVM による固有表現システムの F 値を従来手法と比べたものを**図 3** に示す．RG+DT と ME は磯崎[13]

☆ http://nlp.cs.nyu.edu/irex/NE/から入手可能.

☆☆ http://cl.aist-nara.ac.jp/~taku-ku/software/TinySVM

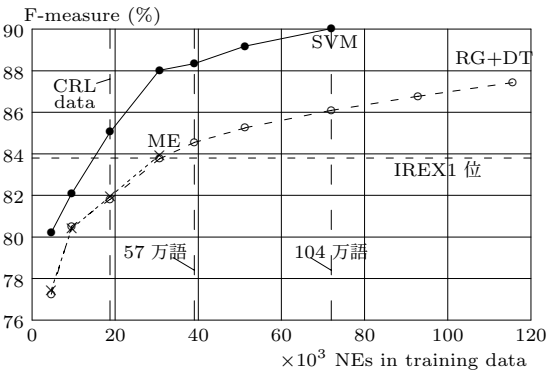

図 3 SVM に基づく手法と他の手法の比較
Fig. 3 Comparison of the SVM-based method with other methods.

の実験結果の再掲であり，前者はルール自動生成法によるシステム，後者は最大エントロピー法によるシステムの成績である．前処理はいずれも共通で，後処理は SVM と ME で共通である．最大エントロピー法のツール☆は大量データでクラッシュしたため，途中までの成績しか分からない．IREX で 1 位だったシステムの成績は 83.86%である．内元ら[38] は最大エントロピー法により 80.17%を得ている．宇津呂ら[43] は最大エントロピー法と決定リストとの組合せにより 84.07%を得たと報告しているが，単語の内部に固有表現の境界がある場合を除いて計算しているため，単純には比較できない．SVM による 85.10%という成績は，これらに比べても良いことが分かる．また，山田ら[40] は CRL データの 5 分割交差検定による F 値の平均として 83.2%を得ているが，我々の結果は 86.77%で，3 ポイント以上高い．

学習データを 104 万語まで増やすと，GENERAL の F 値は 90.03%まで向上した．CRL だけのときと 104 万語のときの内訳を**表 4** に示す．104 万語の場合，人工物名の成績が 50%前後でかなり低い点と，組織名が 80%程度であることを除けば，いずれのクラスも適合率，再現率ともに 90%を超えていることが分かる．

本手法は，最大エントロピー法のための実装方法をそのまま転用したものなので，マルチクラスのうえに確率のようなコストが必要となり，SVM にとっては必ずしも有利な条件ではない．それでも最大エントロピー法を上回る成績が達成できたことになる．シグモイド関数や C などのパラメータをクラスごとに調整することで，さらに精度が向上する可能性はあるが，本論文の主題は高速化であり，ここでは SVM を用いることで，従来手法をしのぐ成績が出せることを示すにとどめる．

☆ E.S. Ristad によるもの．このツールは現在公開されていない．

表 4 成績の内訳
Table 4 Details of the scores.

学習データ	CRL		104 万語	
	再現率	適合率	再現率	適合率
組織名	72.85	82.45	81.16	82.77
人名	88.46	86.92	95.27	93.60
地名	84.02	88.97	90.31	91.20
人工物名	33.33	43.24	41.67	60.61
日付	93.85	94.21	97.31	98.06
時刻	94.44	94.44	98.15	96.36
金額	100.00	100.00	100.00	100.00
割合	90.48	100.00	100.00	100.00
平均	83.05	87.27	89.40	90.66
F 値	85.10		90.03	

4. SVM による分類の高速化

実験により，SVM による固有表現抽出は既存手法より高精度であることが明らかになったが，はじめに述べたように，処理速度があまりに遅く，大量の文書を処理するのには膨大な時間がかかることも分かった．ここでは，その理由を考え，解決策を提案する．

ほとんどの SVM プログラムでは，ゼロ要素の多いベクトルどうしの内積計算 $\mathbf{x}_i \cdot \mathbf{x}_j$ を高速に行うため，非ゼロ要素だけのデータ構造を利用した sparse dot product というアルゴリズム[24] を採用している．しかし，式 (1) の $g(\mathbf{x})$ の式に従って計算しているため，サポートベクトルの数 S に比例した計算量が必要になる．学習データが 57 万語（原文で 2 MB）の場合，33 クラスのサポートベクトルの数の総和は 23 万なので，$g(\mathbf{x})$ の式に従えば，1 語につき 23 万回の内積計算を行わなければならない．したがって，まず式 (1) に従うことをやめなければ，高速化は望めない．

すでに述べたように，固有表現抽出では 2 次の多項式カーネルの成績が良いので，この場合だけ高速になればよい．カーネルが $\mathbf{x}$ の 2 次式ならば，$g(\mathbf{x})$ も $\mathbf{x}$ の 2 次式なので，以下のように書き直せる．

$$g(\mathbf{x}) = W_0 + \sum_{h=1}^{D}(W_1[h]x[h] + W_2[h]x[h]^2) + \sum_{h=1}^{D-1}\sum_{k=h+1}^{D} W_3[h,k]x[h]x[k],$$

ただし，

$$W_0 \stackrel{\text{def}}{=} b + \textstyle\sum_{i=1}^{S} w_i,$$
$$W_1[h] \stackrel{\text{def}}{=} 2\textstyle\sum_{i=1}^{S} w_i z_i[h],$$
$$W_2[h] \stackrel{\text{def}}{=} \textstyle\sum_{i=1}^{S} w_i z_i[h]^2,$$

$$W_3[h,k] \stackrel{\text{def}}{=} 2\sum_{i=1}^{S} w_i z_i[h] z_i[k].$$

ここで $\sum_{i=1}^{S} w_i = \sum_{j=1}^{N} y_j \alpha_j = 0$ なので $W_0 = b$ である．次元数 D は学習データが 57 万語の場合で 15 万なので，単純に計算すると $W_3[h,k]$ の数は百億を超えることになる．カーネル関数は，このような高い次元の計算を元の次元での計算で抑えることで高速な処理を可能にしており，「カーネルトリック」と呼ばれている．つまり，一般的には成分で展開するのは，計算上不利である．

しかし，今の場合，学習データに出現する素性のペアの数が限られているため，実際に計算の必要な $W_3[h,k]$ の数は 57 万語の場合で 660 万にとどまる．これらの定数は，$\mathbf{x}$ にかかわらず，あらかじめ計算しておくことができる．残りは計算するまでもなく，すべてゼロである．ここで，ベクトル $\mathbf{x}$ を，値が 1 である成分の番号のリスト $L(\mathbf{x})$ で表す．たとえば $\mathbf{x} = (0,1,0,0,1,0,1)$ のとき，$L(\mathbf{x}) = [2,5,7]$ である．その要素を $2 \in L(\mathbf{x})$ のように表すと，$g(\mathbf{x})$ は以下のように単純化できる．

$$g(\mathbf{x}) = W_0 + \sum_{h \in L(\mathbf{x})} (W_1'[h] + \sum_{k(>h) \in L(\mathbf{x})} W_3[h,k])$$

ただし

$$W_1'[h] \stackrel{\text{def}}{=} W_1[h] + W_2[h] \quad = \quad 3\sum_{i\,:\,h \in L(\mathbf{z}_i)} w_i,$$

$$W_3[h,k] = 2\sum_{i\,:\,h,k \in L(\mathbf{z}_i)} w_i.$$

つまり，$g(\mathbf{x})$ を計算するには，$L(\mathbf{x})$ の要素 h に対する係数 $W_1'[h]$ と要素ペア (h,k) に対する係数 $W_3[h,k]$ と定数 W_0 を加えればよい．非ゼロ成分は各ベクトルにつき 15 個しかないので，各クラスにつき $15 + {}_{15}C_2 + 1 = 121$ 個の定数の足し算を行うだけでよい．この数はサポートベクトル数 S に関係ないので，本論文のように S が大きいときには有効であると考えられる．33 クラス全部を合わせても，1 形態素の分類の計算に用いられる係数の総数は 3,993 個であり，23 万回の内積計算を行うのにくらべれば，はるかに少ない計算量であると期待される．ただし，ベクトルの次元が高いため，$W_3[h,k]$ をそのまま配列で記憶することはできず，係数データベースから値を取り出すのに若干時間がかかる可能性がある．

以上の考え方は，2 次以外の多項式カーネルやその他のカーネルの多項式近似にも適用可能であるが，とくに 2 次式の場合を XQK（eXpand the Quadratic Kernel）法と呼ぶことにする．値が 0/1 以外の成分があっても，ゼロの成分がほとんどであれば，同様に高速化できる．

```
function g(x1) {
  g = W0;
  i = 0;
  while (x1[i] > 0) { // x1[i] = h
    g += get_w1(x1[i]); // W1'[h]
    j = i + 1;
    while (x1[j] > 0) { // x1[j] = k
      g += get_w3(x1[i], x1[j]); // W3[h,k]
      j++;
    }
    i++;
  }
  return g;
}
```

図 4 展開係数を用いた分類アルゴリズム
Fig. 4 A classification algorithm based on weights given by the expansion.

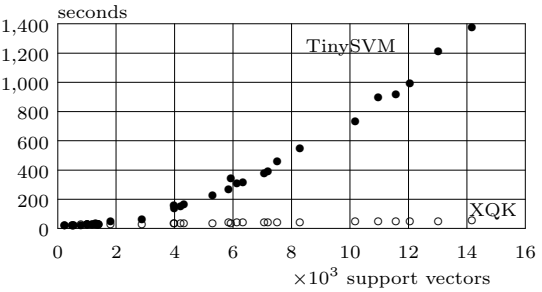

図 5 サポートベクトル数と処理時間
Fig. 5 The number of support vectors versus processing time.

ベクトル $\mathbf{x}$ を $\texttt{x1} = L(\mathbf{x})$ で表したときに $g(\mathbf{x})$ を計算するためのアルゴリズムを**図 4** に示す．なお，$\texttt{x1}[i]$ はリスト $L(\mathbf{x})$ の i 番目の要素を表し，対応する要素がないときは $\texttt{x1}[i] = -1$ とする．図中の $\texttt{get\_w1}(h)$，$\texttt{get\_w3}(h,k)$ は，それぞれ係数 $W_1'[h]$，$W_3[h,k]$ の値をデータベースから取り出す関数である．

ここでは，$W_1'[h]$ はそのまま配列とし，$W_3[h,k]$ はゼロでないもののリストを h ごとに記録しておき，k に関する 2 分探索で $W_3[h,k]$ を探すという方法を用いた．学習データが 57 万語の場合で比較したところ，学習データ自体を分類するのに要する時間は，**表 5** のようになった．XQK の 33 クラス合計の CPU 時間は，TinySVM の 21 倍，SVM-Light [14] の 102 倍高速という結果が得られた．とくに「固有表現以外」のクラスでは，TinySVM の 225 倍，SVM-Light の 587 倍も高速になっている．その他の 32 クラスのサポートベクトル数と CPU 時間の関係を**図 5** に示す．サポートベクトルの数が多いほど高速化の効果があることが分かる．

表 5 を見ると，実際には XQK の処理時間もサポートベクトルの数に若干依存していることが分かる．こ

表 5 処理時間の削減
Table 5 Reduction of processing time.

形態素クラス	サポートベクトル数	TinySVM（秒）	XQK（秒）	高速化率	SVM-Light（秒）
固有表現以外	64,970	11,488.13	**51.11**	*224.8*	29,986.52
人工物名 途中	14,171	1,372.85	**41.32**	*33.2*	6,666.26
場所 単独	13,019	1,209.29	**38.24**	*31.6*	6,100.54
組織名 途中	12,050	987.39	**37.93**	*26.0*	5,570.82
:	:	:	:	:	:
合計	228,306	21,754.23	**1,019.20**	*21.3*	104,466.31
全クラス一括処理			**349.53**	*62.2*	

れは，サポートベクトルの数が増えるほど，ゼロでない係数が増えるので，必要な係数の値を取り出すのに手間がかかるためである．とはいえ，2 分探索にかかる時間はデータ数の対数オーダなので，処理時間の差は既存手法に比べ少ない．

以上の結果は，クラスごとに独立に処理した時間の合計であるが，実際には各形態素ごとに全クラスの結果が必要である．以下のような理由により，形態素ごとに全クラスの $g_c(\mathbf{x})$ をまとめて計算した方が速い．

> 全クラスの $W_3[h,k]$ を 1 つの大きなリストにして辞書式順序でソートすると，同じ $[h,k]$ の係数が 1 カ所に集まる．したがって，1 回の 2 分探索ですべてのクラスの係数を求めることができる．そして，2 分探索の時間計算量は対数オーダなので，クラスごとに 2 分探索を繰り返すより，全クラスの係数をまとめて 1 度の 2 分探索ですませた方が速い．たとえば，10 万個の W_3 の中から特定の $W_3[h,k]$ を取り出すことを 3 クラスで行っているとき，別々に実行すれば計算量は $3\log_2(100,000) = 34.5$ であるが，3 つをまとめると $\log_2(300,000) = 12.6$ ですむ．

この考え方に従って全クラスの $g_c(\mathbf{x})$ を一括して計算するように変更したところ，さらに 3 倍速くなって 349.53 秒となり，TinySVM の 62 倍高速になった．形態素解析などの前処理や，Viterbi アルゴリズム・形態素境界などの後処理を含めても 415.08 秒で済み，4,806 バイト/秒の処理速度が達成できた．ここでは実装が容易な 2 分探索を用いたが，アクセスの速いデータ構造を利用することで，さらに高速化できる可能性がある．

5. 考察と関連研究

2 次の多項式カーネルの展開により，処理速度は向上したが，メモリ使用量は増加する．$\mathbf{z}_i$ の非ゼロ要素の数を m_i (= 15) とすると，従来手法では，各サポートベクトルにつき m_i 個の非ゼロ要素の素性番号と w_i を記憶するだけでよかったのが，m_i 個の係数 $W_1'[h]$ と $m_i(m_i-1)/2$ 個の係数 $W_3[h,k]$ を記憶しなければならなくなる．現在の実装では，h，k に 4 バイトずつ，$W_3[h,k]$ に 8 バイト使っていて，学習データが 57 万語のとき 130 MB，104 万語のとき 190 MB のメモリを使用する．最近の計算機はメモリが大きいので，この程度ではとくに問題ないが，精度に関係のない無駄なデータはない方が高速になる．我々は，これらの係数を利用して素性選択する方法を現在検討中であり，別の機会に報告したい．

また，本論文で提案した XQK 法は，分類の実行を高速化する手法であるが，学習の高速化も切実な問題である．そこで，同じ展開による手法が学習の高速化にも適用できるのではないかと期待される．素朴な SMO アルゴリズム[24] では，$\boldsymbol{\alpha}$ が変わるたびに $g(\mathbf{x}_1),\ldots,g(\mathbf{x}_N)$ の値を計算するので，分類実行とほぼ同じ方法で高速化できる．しかし，TinySVM や libsvm[6] などは，$g(\mathbf{x})$ を計算せず，$W(\boldsymbol{\alpha})$ の勾配の値を記録・更新している．勾配の変化を計算する手間はわずかなので，これ以上速くなることはあまり期待できない．

SVM の処理速度を向上させる方法として，Burges[5] はサポートベクトルの代わりに，Reduced Set Vectors と呼ばれる数の少ない別のベクトル集合で $g(\mathbf{x})$ を近似することで高速化する方法を提案している．しかし，近似手法なので，速度を優先すれば精度は落ちる．また，このベクトル集合を求めるのには複雑な計算を必要とする．Support Vector Regression を用いた近似方法[23] も，同様の問題がある．Romdhani ら[27] は，粗い近似の Reduced Set で可能性のない部分をふるい落とし，精度の高い Reduced Set で残った部分を精査する方法により，精度をあまり落とさずに顔認識を高速化する方法を提案している．

我々の高速化手法は近似手法に比べて単純であり，厳密な値が計算できる．2 次の多項式カーネルで，非ゼロ要素の数が少なく，サポートベクトルの数が多い

という条件が満たされていなければならない☆が，チャンキング[20] や品詞タグ付け[22]，専門用語抽出[39]，構文解析[42] などのタスクは，いずれもこれらの条件を満たすと思われるので，高速化の効果が期待できる．

Downs ら[9] は線形従属なサポートベクトルを除くことで高速化する方法を提案しているが，線形従属なサポートベクトルの数が少なければ高速にはならない．小規模な実験結果が報告されているが，ほとんど削減できない場合もかなり見られる．

今回は多項式カーネルを利用したが，固有表現抽出にとってより良いカーネル関数の研究も今後の課題である．たとえば Collins ら[7] は，タギングカーネルを提案し，パーセプトロンの一種に適用している．

なぜ 2 次のカーネルが良いのかの理由の解明についても，今後の課題である．$d = 1$ より $d = 2$ が良いことから，このタスクについては，素性の組合せを見ることが重要であることが分かる．$d = 3$ で成績が悪いのは，素性 3 つの組合せが過大に評価されるからと思われるが，さらに様々な実験が必要である．

6. おわりに

SVM は高性能な学習手法として最近注目されてきている．しかし同時に，学習速度・処理速度ともに遅くて実用的ではないという問題が指摘されてきた．

本論文では，まず SVM に基づく固有表現抽出システムが，従来手法より好成績であることを実験により示した．CRL データの 5 分割交差検定では F = 86.77%が得られ，IREX GENERAL に対する成績では，CRL データだけを学習に用いた場合でも，F = 85.10%が得られた．また，学習データを増やすことで F = 90.03%という好成績が達成できた．これらの成績は，いずれも従来手法を上回る結果である．

次に処理速度の遅さを改善する計算方法を提案した．この方法は，処理時間がサポートベクトルの数にほとんど依存しないという特長を持ち，サポートベクトル数が 6.5 万個の SVM では，標準的な sparse dot product を用いている SVM-Light の 587 倍，より高速な TinySVM と比べても 225 倍という結果が得られた．この結果，全クラスの合計処理時間は TinySVM の 21 倍になった．さらに複数の SVM の処理をまとめることにより，TinySVM の 62 倍高速になった．この方法は，固有表現抽出にとどまらず，他の様々な自然言語処理タスクに応用可能であると考えられる．

謝辞 TinySVM を公開されている工藤拓氏，アドバイスをいただいた内元清貴氏と山田寛康氏，学習データを提供してくれた佐々木裕氏に感謝します．また，日頃，有益な助言と討論をいただく知識処理研究グループのメンバーと，研究を支援して下さっている片桐滋部長・石井健一郎所長に感謝します．

☆ すでに述べたように，値が 0/1 であることは必要条件ではない．

参考文献

1) 秋葉友良，伊藤克亘，藤井 敦，石川徹也：音声入力による質問応答システムのための音声認識用言語モデルの検討，言語処理学会第 8 回年次大会発表論文集，pp.244–247 (2002).
2) Allwein, E.L., Schapire, R.E. and Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, *Journal of Machine Learning Research*, Vol.1, pp.113–141 (2000).
3) Bikel, D.M., Schwartz, R. and Weischedel, R.M.: An Algorithm that Learns What's in a Name, *Machine Learning*, Vol.34, No.1-3, pp.211–231 (1999).
4) Borthwick, A.: A Maximum Entropy Approach to Named Entity Recognition, Ph.D. Thesis, New York University (1999).
5) Burges, C.J.C.: Simplified support vector decision rules, *Proc. International Conference on Machine Learning*, pp.71–77 (1996).
6) Chang, C.-C. and Lin, C.-J.: LIBSVM: A Library for Support Vector Machines, National Taiwan University (2002). http://www.csie.ntu.edu.tw/~cjlin/libsvm.html.
7) Collins, M. and Duffy, N.: New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perception, *Proc. 40th Annual Meeting of the Association for Computational Linguistics* (*ACL*), pp.263–270 (2002).
8) Cristianini, N. and Shawe-Taylor, J.: *An Introduction to Support Vector Machines*, Cambridge University Press (2000).
9) Downs, T., Gates, K.E. and Masters, A.: Exact Simplification of Support Vector Solutions, *Journal of Machine Learning Research*, Vol.2, pp.293–297 (2001).
10) Hirao, T., Isozaki, H. and Maeda, E.: Extracting Important Sentences with Support Vector Machines, *Proc. International Conference on Computational Linguistics*, pp.342–348 (2002).
11) 平尾 努，賀沢秀人，磯崎秀樹，前田英作，松本裕治：機械学習を用いた複数文書要約，情報科学技術フォーラム，pp.91–92 (2002).
12) 磯崎秀樹：辞書式優先順位に基づく日本語固有表現抽出，情報処理学会研究報告 NL-135-5, pp.33–

40 (2000).
13) 磯崎秀樹：メタルールと決定木学習を用いた日本語固有表現抽出，情報処理学会論文誌，Vol.43, No.5, pp.1481–1491 (2002).
14) Joachims, T.: Making Large-Scale Support Vector Machine Learning Practical, *Advances in Kernel Methods*, Schölkopf, B., Burges, C.J.C. and Smola, A.J. (Eds.), chapter 16, pp.170–184, MIT Press (1999).
15) 賀沢秀人，平尾　努：固有表現に着目した双方向イベントトラッキングとその文書要約への応用，情報処理学会研究報告 NL-143 (2001).
16) 北　研二：確率的言語モデル，東京大学出版会 (1999).
17) 国領弘治，佐々木裕，前田英作：固有表現を利用した大量文書の時系列ブラウジング，情報科学技術フォーラム，pp.93–94 (2002).
18) Kreßel, U.H.-G.: Pairwise Classification and Support Vector Machines, *Advances in Kernel Methods*, Schölkopf, B., Burges, C.J.C. and Smola, A.J. (Eds.), chapter 15, pp.255–268, MIT Press (1999).
19) 工藤　拓，松本裕治：Support Vector Machine による日本語係り受け解析，情報処理学会研究会報告 NL-138 (2000).
20) Kudo, T. and Matsumoto, Y.: Chunking with Support Vector Machines, *Proc. 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pp.192–199 (2001).
21) 松本裕治，北内　啓，山下達雄，平野善隆，松田寛，高岡一馬，浅原正幸：形態素解析システム『茶筌』version 2.2.1 使用説明書，奈良先端科学技術大学院大学 (2000).
22) Nakagawa, T., Kudoh, T. and Matsumoto, Y.: Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines, *Proc. 6th Natural Language Processing Pacific Rim Symposium*, pp.325–331 (2001).
23) Osuna, E.E. and Girosi, F.: Reducing the Run-time Complexity in Support Vector Machines, *Advances in Kernel Methods*, Schölkopf, B., Burges, C.J.C. and Smola, A.J. (Eds.), chapter 16, pp.271–283, MIT Press (1999).
24) Platt, J.C.: Fast Training of Support Vector Machines Using Sequential Minimal Optimization, *Advances in Kernel Methods*, Schölkopf, B., Burges, C.J.C. and Smola, A.J. (Eds.), chapter 12, pp.185–208, MIT Press (1999).
25) Platt, J.C.: Probabilities for SV Machines, *Advances in Large Margin Classifiers*, Smola, A.J., Bartlett, P.L., Schölkopf, B. and Schuurmans, D. (Eds.), chapter 5, pp.61–71, MIT Press (2000).
26) Platt, J.C., Cristianini, N. and Shawe-Taylor, J.: Large margin DAGs for multiclass classification, *Advances in Neural Information Processing Systems 12*, pp.547–553, MIT Press (2000).
27) Romdhani, S., Torr, P., Schölkopf, B. and Blake, A.: Computationally Efficient Face Detection, *Proc. International Conference on Computer Vision*, pp.695–700 (2001).
28) 佐々木裕：トランスデューサによる日本語固有表現抽出，言語処理学会第 5 回年次大会発表論文集，pp.108–111 (1999).
29) 佐々木裕，磯崎秀樹，平　博順，平尾　努，賀沢秀人，鈴木　潤，国領弘治，前田英作：SAIQA：大量文書に基づく質問応答システム，情報処理学会研究報告，pp.77–82 (2001). FI-64-12, NL-145-12.
30) 颯々野学，宇津呂武仁：統計的日本語固有表現抽出における固有表現まとめ上げ手法とその評価，情報処理学会研究報告 NL-139-1 (2000).
31) Schölkopf, B., Burges, C.J.C. and Smola, A.J. (Eds.): *Advances in Kernel Methods*, MIT Press (1999).
32) Schwenker, F.: Solving Multi-class Pattern Recognition Problems with Tree-Structured Support Vector Machines, *Pattern Recognition, Proc. 23rd Symposium*, Radig, B. and Florczyk, S. (Eds.), LNCS No.2191, pp.283–290, Springer (2001).
33) Sekine, S., Grishman, R. and Shinnou, H.: A Decision Tree Method for Finding and Classifying Names in Japanese Texts, *Proc. 6th Workshop on Very Large Corpora* (1998).
34) 志賀正裕，太田知宏，藤畑勝之，公文隆太郎，森辰則：実時間質問応答のための探索制御付き命題照合，言語処理学会第 8 回年次大会発表論文集，pp.267–270 (2002).
35) Suzuki, J., Sasaki, Y. and Maeda, E.: SVM Answer Selection for Open-domain Question Answering, *Proc. International Conference on Computational Linguistics*, pp.974–980 (2002).
36) 鈴木　潤，佐々木裕，前田英作：統計的機械学習を用いた質問タイプ同定，情報科学技術フォーラム，pp.89–90 (2002).
37) 竹元義美，福島俊一，山田洋志：辞書およびパターンマッチルールの増強と品質強化に基づく日本語固有表現抽出，情報処理学会論文誌，Vol.42, No.6, pp.1580–1591 (2001).
38) 内元清貴，馬　青，村田真樹，小作浩美，内山将夫，井佐原均：最大エントロピーモデルと書き換え規則に基づく固有表現抽出，自然言語処理，Vol.7, No.2, pp.63–90 (2000).
39) 山田寛康，工藤　拓，松本裕治：単語の部分文字列を考慮した専門用語抽出と分類，情報処理学会研究報告 NL-140-11 (2000).

40) 山田寛康，工藤 拓，松本裕治：Support Vector Machines を用いた日本語固有表現抽出，情報処理学会論文誌，Vol.43, No.1, pp.44–53 (2002).
41) 山田寛康，松本裕治：Support Vector Machine の多値分類問題への適用法について，情報処理学会研究報告 NL-146-6 (2001).
42) 山田寛康，松本裕治：Support Vector Machine を用いた決定性上昇型構文解析，情報処理学会研究報告 NL-149-9 (2002).
43) 宇津呂武仁，颯々野学，内元清貴：正誤判別規則学習を用いた複数の日本語固有表現抽出システムの出力の混合，自然言語処理，Vol.9, No.1, pp.65–100 (2002).

(平成 14 年 10 月 8 日受付)
(平成 15 年 1 月 7 日採録)

磯崎 秀樹（正会員）

1983 年東京大学工学部計数工学科卒業．1986 年同工学系大学院修士課程修了．同年，日本電信電話（株）入社．1990 年〜1991 年スタンフォード大学ロボティクス研究所客員研究員．現在，NTT コミュニケーション科学基礎研究所特別研究員．博士（工学）．人工知能・自然言語処理の研究に従事．電子情報通信学会，人工知能学会，言語処理学会，AAAI，ACL 各会員．

賀沢 秀人（正会員）

1995 年東京大学理学部物理学科卒業．1997 年同大学院理学系研究科修士課程修了．同年，日本電信電話（株）入社．現在，NTT コミュニケーション科学基礎研究所に所属．主として自然言語処理，機械学習の研究に従事．ACM，IEEE 各会員．