情報処理学会論文誌　Vol. 52　No. 7　2291–2307 (July 2011)

Boostingに基づく系列ラベリングにおける効率的規則表現方法による高速化

岩倉友哉[†1]

本稿では，品詞タグ付け，Text Chunkingといった系列ラベリングにおける高速化手法を提案する．まず，系列ラベリング向けの規則表現方法を提案する．通常の系列ラベリング問題では，判別対象の単語から得られる素性に加えて，その前後の単語から得られる素性を用いる．そのため，判別対象の単語からの相対位置により，各単語から得られる素性が，異なる素性として，複数回処理されるという問題がある．本稿では，この相対位置による違いを吸収可能な規則表現方法による判別方法を提案する．また，素性の組合せを規則として学習するBoostingアルゴリズムにおいて，素性の組合せ生成を制御する手法を提案する．素性の組合せを考慮することで，精度改善に貢献するが，処理速度が低下するという問題がある．本手法では，組合せに考慮される素性とされない素性の2種類に分けることで，組合せ生成を調整し，高速化を行う．本手法を，英語品詞タグ付け，Text Chunkingで評価した結果，精度を保持したまま，速度改善が得られることを確認した．

An Efficient Rule Representation for Fast Boosting-based Sequential Labeling

Tomoya Iwakura[†1]

This paper proposes two techniques to improve processing speed for sequential labeling problems such as part-of-speech (POS) tagging, text chunking, and so on. The first technique is a rule representation. Usual parsers for sequential labeling decide the class of each word by using the features generated from the word and its surrounding words. As a result, each word is evaluated a number of times. We propose a rule representation that allow us to avoid unnecessary such evaluation. The other is a boosting-based algorithm learning rules represented by combination of features. Although combination of features contributes accuracy, there is a problem that considering combination degrades processing speed. To avoid time-consuming evaluation of combination, we specify features never considered for combination. We evaluate our methods with English POS tagging and text chunking. The experimental results show that the processing time obtained with our methods is shorter than the processing time of taggers and chunkers without our techniques while maintaining competitive accuracy.

1. はじめに

系列ラベリングとは，観測された系列に対し，ラベル列を付与することにあたり，自然言語処理であれば，単語列の各単語に対してラベルを付与する，品詞タグ付与，名詞句や動詞句などの基本フレーズを判別するText Chunking，人名などの固有名詞や日付などの数値表現を抽出する固有表現抽出といった技術が含まれる．

これら系列ラベリングに含まれる技術は，種々の自然言語処理を扱うアプリケーションで利用されている．固有表現抽出であれば，個人名の匿名化[7),17)]といったアプリケーションに応用されている．また，品詞タグ付けとText Chunkingは固有表現抽出の前処理として利用されるなど，数多くの自然言語処理タスクにおいて利用されている技術である．このような自然言語処理を扱うアプリケーションは，Webページ，社内文書，電子メールといった日々増加するテキストデータを対象とすることも多いことから，高速な処理が求められており，これらのアプリケーションで利用される系列ラベリングに含まれる自然言語処理技術の高速化が必要であるといえる．

本稿では，系列ラベリングのための高速化手法を提案し，英語品詞タグ付けとText Chunkingという2つの系列ラベリングタスクにおける評価結果を報告する．

まず，本稿では，高速化のために，系列ラベリング向けの規則表現方法とその表現方法に基づく抽出手法を提案する．近年，品詞タグ付けやText Chunking，固有表現抽出などの系列ラベリング問題に対しては，Support Vector Machines（SVMs）[10)]，最大エントロピー法[24)]，Boostingに基づく規則学習手法[9)]，Conditional Random Fields（CRF）[14)]，Structured Perceptron[2)]のような手法が適用され，高い精度が報告されている．これらの機械学習手法に基づく手法による品詞タグ付けやText Chunkingでは，通常，各単語のラベルを決定する際に，現在の判別対象の単語に加えて，前後の単語から得られる素性を用いて判別を行う．そのため，判別対象の単語からの相対位置により，各単語の表記やその文字種別などの属性が異なる素性として複数回処理される．本稿では，この相対位置による違い

†1 株式会社富士通研究所
Fujitsu Laboratories Ltd.

を吸収することでの速度改善を目的に，相対位置による違いを吸収可能な規則表現方法とその規則表現による判別方法を提案する．たとえば，Kernel に基づく手法向けには，処理速度改善のための手法が提案され，高速化に成功している[6),11),25)]．しかし，従来手法は機械学習手法に直接関連するものである．これに対し，本規則表現方法は，前後の単語から得られる情報を用いる系列ラベリングにおいて利用可能な系列ラベリングのための高速化の一手法となる．この手法は，系列ラベリングにおいて，各クラスの判別の条件が素性あるいは素性の組合せで定義される場合に適用できる．本手法は，たとえば，CRF，Structured Perceptron，Boosting に基づく規則学習器などの手法による学習結果に利用可能である．また，SVMs のような Kernel を用いる学習手法であっても，今までに提案されてきた高速化手法[6),11)] と組み合わせることで，この手法は利用可能となる．

続いて，本稿では，素性の組合せの生成が調節可能な Boosting に基づく規則学習手法を提案する．本稿では，多くの従来手法と同様に，機械学習手法を用いて品詞タグ付けと Text Chunking を実現する．機械学習手法はいくつか存在するが，次の理由から Boosting に基づく規則学習手法[9)] が今回の評価に適していると考え利用することにした．まず，他の機械学習手法と同様に，Boosting に基づく手法は，文書分類[20)]，品詞タグ付け[9)]，Text Chunking[9),13)] などに適用され，高い精度が報告されている．処理速度の面においても，文献 12) にあるように，Boosting に基づく規則学習器は，小さいサイズの規則集合を生成する傾向にあり，高速な分類が行えることが報告されている．また，今回利用する学習手法[9)] は，素性の組合せを自動で生成する Boosting に基づく手法である．素性の組合せを人手で与えている手法[3),21),22)] と比較し，この手法では，単一の素性から構成される規則に加えて，素性の組合せから構成される規則を自動で学習する．さらに，この Boosting 手法による学習結果は，自然言語処理で広くられている Polynomial Kernel を用いて暗黙的に素性の組合せを考慮する SVMs[6),11)] や Perceptron のようなオンライン学習手法[18)] による学習結果とは異なり，本稿で提案する規則表現方法に直接変換可能という特徴もある．また，この学習手法を用いた英語品詞タグ付けと Text Chunking は，2 章で説明する系列ラベリングで利用される 2 種類の素性を導入可能である．

ただ，今回利用する Boosting 学習手法のように，すべての素性の組合せを自動で考慮する手法を，素性の種類数が多い自然言語処理タスクに適用する場合，生成される組合せ数も膨大となるため，処理時間が問題となる．これに対し，従来手法[3),21),22)] では，素性の組合せ選択を人手で行う必要があるものの，処理速度の改善という観点から，素性組合せの生成を調整可能であるという利点がある．

本稿では，組合せ数の増加を抑えることを目的に，Boosting 手法を対象とした素性組合せの生成手法を提案する．従来手法が素性の組合せを人手で指定していたのに対し，本手法では，逆に，組合せに利用しない素性を指定するアプローチを用いる．素性の組合せは，組合せに利用しないと指定された素性以外から学習する．このような方法を用いることで，素性の組合せ生成の調整を可能とし，処理時間の改善を目指す．

本稿の構成は次のとおりである．まず，2 章で従来の系列ラベリング手法について述べ，3 章で系列ラベリングのための規則表現方法およびその規則表現を用いた判別方法について述べる．続いて，4 章で素性組合せ生成調整可能な Boosting アルゴリズムについて述べる．5 章で本手法の評価に用いる英語品詞タグ付けタスクと Text Chunking タスクについて述べ，6 章で評価結果について述べる．7 章で先行研究を紹介し，8 章で今後の課題を述べる．最後に 9 章で本稿をまとめる．

2. 従来手法による系列ラベリング

本章では，従来手法の概要と速度低下に関連する点について説明する．ここでは，系列ラベリング問題の 1 つである英語品詞タグ付け問題を例に説明する．

通常，英語品詞タグ付けや Text Chunking などの系列ラベリングでは，各単語のクラスラベルを判別するために，現在の対象の単語に加えて，その前後に出現する単語から得られる素性を用いる．英語品詞タグ付けであれば，クラスラベルは品詞となる．ここでは，“I am happy.” 中の単語の品詞を判別する例で説明を行う．

今回の判別手法では，文頭から文末，あるいは，文末から文頭の方向で順次各単語がどのクラスラベルになりうるかを計算し，その結果をもとに，入力単語列に付与する最終的なクラスラベル列を決定する．解析の方向は，後に説明する動的素性の利用によって決まる．文末から文頭の方向で解析するとすると，“.”，“happy”，“am”，“I” という順番で処理することになる．

ここでの説明では，判別対象である現在位置の単語および前後 1 単語の表記を素性として用いて判別を行うとする．この条件で，“am” の品詞タグを判別する場合は，1 つ前の “I”，現在位置の “am”，1 つ後の “happy” の 3 単語の表記が素性として利用される．実際には，同一の表記であっても異なる出現位置の単語であることを区別するために，現在位置を基点に，“I:−1”，“am:0”，“happy:1”，のように素性を表現する．以降，素性の “:” の後の −1，0，1 は，現在の判別対象の単語からの相対位置を示す**相対位置情報**と呼ぶことにする．また，“I”，“am” のように，単語表記や，単語を構成する文字種別といった，相対位置情報

をともなわない単語の属性情報を**属性**と呼ぶことにする．

“happy” の品詞を判別する場合は，“am” は “am:−1”，“happy” は “happy:0”，“.” は “.:1”，となる．このように，判別対象の単語からの相対位置によって同一の単語であっても異なる素性として扱われるため同一の単語が複数回処理されることになる．“am” を例に見てみると “am:−1”，“am:0” と異なる素性となっている．こちらの例では，前後 1 単語を用いるため，単語の表記が最大で 3 回処理される．

素性抽出後は，それらの素性を使って，各単語がどのクラスラベルになりうるかを計算する．クラスラベルの判別は，たとえば，“am:0” が出現したらクラスラベルは VBP，といった素性を条件として判別するような規則を用いて判別を行う．すべての単語を処理した後に，入力列に対して付与される最適なクラスラベル列を決定する．

この例では，単語の表記だけを利用しているが，実際には，様々な素性が利用される．たとえば，英語の品詞タグ付けや Text Chunking であれば，単語の先頭および最後の数文字，各単語をすべて小文字（あるいは大文字）に変換した結果，各単語が大文字で始まっているかなどの単語から得られる様々な属性を基にした素性が用いられている[3),9),23)]．本稿では，このような入力の単語列から得られる素性を，**静的素性**と呼ぶことにする．

さらに，系列ラベリングでは，静的素性に加え，解析時に動的に決定される**動的素性**を利用する方法も提案されている[9),10)]．動的素性を利用する場合は，解析方向を決定する必要がある．たとえば，英語品詞タグ付けで，解析方向を文末から文頭とした場合，現在位置から文末側にある指定された範囲の単語に付与された品詞を動的素性として用いる．動的素性を利用しない場合や CRF などを用いて first-order Markov モデルを学習するような場合は，解析方向については考慮せずに，静的素性の処理だけとなる．

相対位置により異なる素性として扱われるという問題は，静的素性と動的素性の両方において起きうる．そこで，本稿では，先行研究[9),10)] で用いられている，静的素性および動的素性という 2 種類の素性を用いて系列ラベリングを実施する手法を基に評価を行う．また，文献 10) で文末から文頭方向の解析で動的素性を利用することで高い精度が得られることが報告されていることから，本稿でも同様の解析方向で評価を行う．

3. 系列ラベリングのための効率的規則表現方法による判別

本章では系列ラベリングのための規則表現方法およびその表現方法による判別方法を説明する．本稿では，従来手法に基づいて学習した規則を提案手法の規則表現に変換し利用するアプローチを用いる．まず，3.1 節で系列ラベリングのための規則表現方法について説明し，3.2 節で従来の規則表現からの変換方法について説明する．続いて，規則適用の前提について 3.3 節で説明し，3.4 節で規則適用方法を，3.5 節で規則適用例を説明する．最後に 3.6 節で本提案手法の利点と欠点について述べる．

3.1 系列ラベリングのための効率的規則表現方法

本稿で提案する系列ラベリングのための規則表現手法である**圧縮規則表現**について説明する．圧縮規則表現は，2 章で説明した，同一の単語から得られる属性が処理対象の単語からの相対位置によって異なる素性として扱かわれるという問題を解決するために用いる．

まず，従来の規則表現方法について説明する．多くの学習手法による学習結果は ⟨ 規則，クラスラベル，確信度 ⟩ の集合で表現でき，規則（**rule**）を中心にまとめると次のようになる．

$$\langle \mathbf{rule}, \{(cl_1, c_1), \ldots, (cl_q, c_q)\}\rangle,$$

本稿では，この形式を**従来規則表現**と呼ぶことにする．ここで，**rule** は，素性あるいは素性の組合せから構成される規則である．ここでの素性は，静的素性および動的素性の両方を含む．cl_p（$1 \leq p \leq q$）はクラスラベルであり，c_p は cl_p の確信度である．学習アルゴリズムによって，確信度に相当する個所の学習結果は，確率やスコアなど，それぞれのアルゴリズムによって異なるが，今回の説明では，確信度をそれらを意味する語として用いることにする．従来規則表現は，本稿で用いる Boosting 手法に限らず，最大エントロピー法，CRF，Structured Perceptron などの学習結果にもあてまはる[⋆1]．また，Polynomial Kernel を用いた SVMs による学習結果であれば，今までに提案されてきた手法[6),11)] による変換した結果は，Polynomial Kernel で暗黙的に展開していた素性の組合せを明示的に展開するので，従来規則表現にあてはまるといえる．

従来規則表現では，規則適用後に確信度が付与される対象は現在位置の単語と一意に決まる．これに対し，圧縮規則表現は，次にある，規則適用後に確信度を複数個所に付与可能な表現とする．

$$\langle \mathbf{rule}', \{(p_1, cl_1, c_1), \ldots, (p_q, cl_q, c_q)\}\rangle,$$

ここで，$\mathbf{rule}'$ は，素性あるいは素性の組合せである．また，p_p は**基準位置**の単語からみてどの位置の単語にクラスラベル cl_p（$1 \leq p \leq q$）の確信度 c_p を付与するかを示すものであり，**確信度付与位置**と呼ぶことにする．従来規則表現との大きな違いは，確信度付与位置が

⋆1 たとえば，first-order Markov モデルの学習であれば，cl_p の個所には連接するタグの組合せとなるなど，クラスラベルの個所に入る値に若干の違いはある．

```
## f: 学習アルゴリズムで生成された規則
## c : f の確信度
## cl : f のクラスラベル
## s(f): f の属性
## p(f): f の相対位置情報
## fn: 圧縮規則表現に変換後の f
## RC[fn]: fn の確信度情報
procedure ruleConv( f, c, cl)
 bp = selectBasePOS(f) ## f 中の素性の基準位置の選択.
 Foreach f ∈ f ## 変換開始
  lm = p(f) − bp ## f の新たな相対位置情報
  ## 変換後の f を追加
  fn = fn + "s(f):lm"
 endForeach
 RC[fn] = RC[fn] ∪ (−bp, cl, c)
```

図 1　従来規則表現から圧縮規則表現への変換
Fig. 1　Converting usual rule representation into compressed rule representation.

追加されている点である．これにより，規則適用後に，複数の対象に対し確信度が付与可能となり，同一の単語から得られる属性が処理対象の単語からの相対位置によって異なる素性として扱われるという問題を解決する．

圧縮規則表現の **rule**′ は，従来規則表現の規則と同様に，素性あるいは素性の組合せであるが，2 章で説明した相対位置情報の決め方に次のような違いがある．従来規則表現では判別対象の単語の位置を基準位置として素性生成に利用する各単語の相対位置情報を決め，各単語から得られる属性と相対位置情報を組み合わせて素性を生成する．これに対し，圧縮規則表現においては，**rule**′ の静的素性のうち，最も文頭側に出現する単語から得られる素性とのチェックに利用される静的素性の位置が基準位置となる．残りの素性の相対位置情報は，その基準の素性からの相対位置で表現される．静的素性が存在しない場合は，最も文頭側に出現する単語に対してのチェックに利用される動的素性を基準位置とする．

3.2　従来規則表現から圧縮規則表現への変換

ここでは，従来規則表現から圧縮規則表現に変換する方法について説明する．**図 1** は規則変換手法の概要である．この変換手法を，次の 3 種類の従来規則表現を使って説明する．

$\langle\{\text{I:}-2, \text{am:}-1\}, \text{JJ}, c_0\rangle$, $\langle\{\text{I:}-1, \text{am:}0\}, \text{VBP}, c_1\rangle$, $\langle\{\text{I:}0, \text{am:}1\}, \text{PRP}, c_2\rangle$

ここで，JJ，VBP，PRP はクラスラベルで，c_0，c_1，c_2 が確信度である．

ここでの説明では，各素性は，相対位置情報と属性という 2 つに分割できると仮定する．相対位置情報とは現在位置の単語からの相対位置を意味し，属性とは，単語表記，単語を構成する文字種別といった，相対位置情報をともなわない単語の属性情報である．":" が相対位置情報と属性の区切りである．I:−2 であれば，I が属性，−2 が相対位置情報となる．

変換のためには，規則の中でも最も文頭側に出現する単語から得られる静的素性の相対位置を示す相対位置情報を基準位置（bp）と定める．静的素性が存在しない場合は，規則の中でも最も文頭側に出現する単語から得られる動的素性の相対位置情報を bp として利用する．続いて，規則の中の各素性を属性と相対位置情報に分割し，bp からの相対位置情報で表現された相対位置情報に変換し，変換後の相対位置情報と属性で新たな素性を生成する．上記の例の，$\{\text{I:}-2, \text{am:}-1\}$ であれば -2 が bp となり，bp との差を用いて，$\{\text{I:}0, \text{am:}1\}$ と変換される．また，$\{\text{I:}-1, \text{am:}0\}$ であれば -1 が bp なので，$\{\text{I:}0, \text{am:}1\}$ と変換される．

最後に，$(bp \times -1)$ の値を，現在の規則の確信度の確信度付与位置とする．従来規則表現においては，現在位置単語が確信度付与対象となる．変換後は，規則中の最も文頭側の単語から生成された素性の位置 bp が基点となるので，確信度の付与位置は，$(bp \times -1)$ となる．変換前が $\{\text{I:}-2, \text{am:}-1\}$ であれば -2 が bp なので，この規則のクラスラベル JJ の確信度 c_0 の確信度付与位置は 2 となる．

これらの従来規則表現は，最終的に，次のような圧縮規則表現に基づく表現に変換される．

$\langle\{\text{I:}0, \text{am:}1\}, \{(2, \text{JJ}, c_0), (1, \text{VBP}, c_1), (0, \text{PRP}, c_2)\}\rangle$

続いて，変換前後で規則適用の変化を "I am happy."，という単語列を用いて説明する．この例であれば，変換前の従来規則表現を用いた場合は，3 回の規則チェックを実施する必要がある．これに対し，変換後の圧縮規則表現を用いた場合は，3 つの単語への確信度付与が，$\{\text{I:}0, \text{am:}1\}$ を，"I" を基点に 1 度チェックするだけで済む．規則チェック後に，クラスラベル "JJ" の確信度は "I" から 2 つ隣にある "happy" に，"VBP" の確信度は 'I" から 1 つ隣にある "am" に，"PRP" の確信度は "I" に付与される．

3.3　規則適用前提

圧縮規則表現の適用の前提および木構造による規則表現方法について説明する．$\mathcal{RC}$ を圧縮規則表現による規則集合とし，$\mathcal{RC}[\textbf{rule}']$ は規則 **rule**′ に対応する 〈 確信度付与位置, クラスラベル, 確信度 〉 の集合とする．

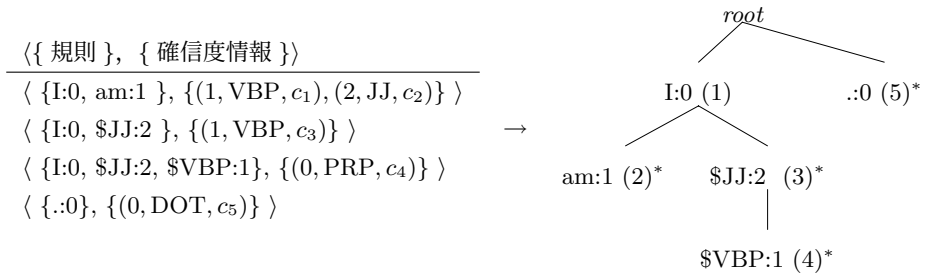


図 2　圧縮規則表現（左）とその規則の木構造表現（右）の例．丸括弧の数字は木構造表現内での位置を表す．また，∗ は確信度情報を有するノードという意味．*root* は規則木構造のルート

Fig. 2　An example of compressed rule representation and its tree representation. The number of each parenthesis indicates a position in the tree representation. Each node with asterisk (*) means the node has confidence values for class labels. The *root* indicates the root node of the tree.

また，$\mathbf{rule}' \in \mathcal{RC}$ のうち，素性集合の **rc** のすべての素性を含み，**rc** と $\mathbf{rule}'$ が同一でない場合が存在すれば，**rc** を $\mathcal{RC}$ の部分集合と呼ぶことにする．規則の部分集合の関係は，次のように記す．

$$\mathbf{rc} \subset \mathcal{RC}$$

$\mathbf{Rs}[i]$ には，i 番目の単語から得られる属性とその相対位置情報が 0 で表現される素性を含む規則の適用状況を記録する．**図 2** に規則の木構造表現の例を載せる．本実装では，Double Array [1)] というトライ構造を用いて，規則を木構造で表現する．この例では，説明のために，1 つの素性を 1 つのノードとした木構造としている．木構造で表現する理由は，チェックのために完全なキーが必要となるハッシュと違い，木構造では，部分的なキー情報であっても途中までのチェックが可能であるからである．$\mathbf{Rs}[i]$ には各規則の部分集合が到達したトライ構造内での位置を記録する．

2 章で説明したように，解析方向は文末から文頭方向として動的素性を決定するため，$JJ と $VBP というクラスラベルから得られる動的素性については，静的素性とは逆に出現位置の降順で記録する．

図 2 の規則集合において，“I am happy.” という入力に，単語表記だけを利用して判別を行う場合の規則適用状況の記録について説明する．まず，I が 1 番目の単語であるので，“I:0” を含む規則の適用状況は，$\mathbf{Rs}[1]$ に記録される．この例では，“I:0” $\subset \mathcal{RC}$ であり部分集合の条件を満たすので，$\mathbf{Rs}[1]$ には，木構造内での “I:0” に到達した位置 (1) が記録される．続いて，“I” を基準位置とした組合せチェックである “am:1” を含む規則のチェックでは，$\mathbf{Rs}[1]$ 中に記録された位置である (1) からチェックを開始する．“am” を基準位置としたチェックである “am:0” を含む規則の適用状況は，$\mathbf{Rs}[2]$ に記録されるが，この例では，条件に合わないため，記録はされない．

3.4　圧縮規則表現の適用

図 3 に規則適用の概要を示す．規則適用では，2 章で説明した 2 種類の素性である静的素性と動的素性に対しての 2 段階の処理を行う．解析方向としては，2 章で説明したように，文末から文頭方向の解析で実装を行う．以降，W を利用する単語情報の範囲とする．また，Δ を動的素性を利用する範囲とする．

まず，静的素性の利用について説明する．W は，従来規則表現の適用では，現在位置の単語に加えて，現在位置から見て，文頭側および文末側に出現する $\frac{W-1}{2}$ 単語から得られる素性を利用するという意味になる．今回の圧縮規則表現の適用では，規則適用の基準位置が変更となるだけで，利用する単語範囲は同じである．よって，規則適用を i' 番目の単語から開始する場合，その単語を含んで，W 単語までとなり，利用される単語の範囲は次となる．

$$i' \quad \leq \quad i \quad \leq \quad i' + W - 1 \tag{1}$$

図 3 の前半部分が静的素性のチェックの個所である．i' が基点の単語の位置となり，式 (1) によって決まる，その位置 i' から $i' + W - 1$ 番目までの単語から得られる静的素性をチェックする．

続いて動的素性の利用について説明する．従来規則表現の適用では，動的素性は現在の判別対象から解析方向と逆向きの Δ 単語に付与されたクラスラベルを用いるという意味である．本稿における圧縮規則表現の適用では，i’ 番目の単語のクラスラベルが決定された後に，i’ 番目の単語を中心に決定される範囲内の単語から得られる素性と，i’ 番目の単語の動的素性のチェックを行う．本手法では，次の範囲にある単語から得られる素性と，i’ 番目から得られる動的素性をチェックを行う．

$$\left(i' - \frac{W-1}{2} - \Delta\right) \leq i \leq \left(i' + \max\left(\frac{W-1}{2}, \Delta\right) - 1\right) \tag{2}$$

動的素性の相対位置情報は，式 (2) の i との差によって決まり，i’$-i$ となる．

以下に，式 (2) にある圧縮規則表現の適用範囲の決定方法を説明する．ここでは，文末から文頭の解析による従来規則表現の適用時の動的素性の利用例とともに，i’ 番目の単語から得られる動的素性を利用した規則適用の範囲を，文頭側，文末側の順番で説明する．

まず，文頭側の範囲について説明する．従来規則表現の適用時は，i 番目の単語のクラス

```
## RC[rc] は rc の確信度情報,
## Rs[i]: i 番目の単語の規則適用状況．初期値は木構造による規則のルート（root）
procedure ruleApplication( {wd_1, ..., wd_N})
For  i' = 1, ..., N;  ## 静的素性のチェックを開始する単語の位置
 For  i = i', ..., (i' + W − 1);  ## 現在のチェック位置
  For  j = 1, ..., |wd_i|;  ## i 番目の単語の属性．|wd_i| は属性数.
   Foreach  rc ∈ Rs[i']  ## j 番目の属性 wd_{i,j} ∈ wd_i と rc の組合せをチェック
    lm = i − i';    rc' = rc + "wd_{i,j}:lm";   ## lm は i' からの相対位置情報
    assignScores(RC[rc'], i');   ## rc' が適用されたら確信度を対象単語に付与.
    If rc' ⊂ RC  Rs[i'] = Rs[i'] ∪ rc' ##部分集合の場合は適用状況を記録
   endForeach
  endFor;
  If Rs[i'] = {root} break;  ## i' 番目の規則適用状況が root だけの場合は i'+1 に
  移動
 endFor;
endFor;
For  i' = N, ..., 1;  ##動的素性を文末から文頭に向けてチェック
  For  i = (i' − (W−1)/2 − Δ), ..., (i' + max((W−1)/2, Δ) − 1);
  Foreach  rc ∈ Rs[i]
    lm = i' − i;    rc' = rc + "dft_{i':lm}"   ## dft'_i は i' 番目の単語のクラスラベル
    assignScores(RC[rc'], i); ##適用された場合は確信度付与
    If rc' ⊂ RC  Rs[i] = Rs[i] ∪ rc';  ## 適用状況の記録
  endForeach
 endFor
endFor
```

図 3　圧縮規則表現における系列ラベリング
Fig. 3　A sequential labeling method with compressed rule representation.

ラベル判別時は，i 番目の単語から文頭側および文末側に出現する $\frac{W-1}{2}$ 単語から得られる素性と，文末側の i+1 番目から i+Δ 番目の単語の動的素性を利用する．よって，クラスラベルが動的素性として利用される一番文末側の単語の位置 (i+Δ) と，素性として利用される範囲の中で，現在位置の単語から最も文頭側に出現する単語の位置 $(i - \frac{W-1}{2})$ の差は，$|\frac{W-1}{2} + \Delta|$ となる．よって，i' 番目の単語のクラスラベルを動的素性として利用する際は，i' から，文頭側に向けて $|\frac{W-1}{2} + \Delta|$ 単語が対象となる．

続いて文末側の範囲について説明する．従来規則表現の適用時に，i に位置する単語が現在位置である場合，最も近い動的素性を利用する単語の位置は (i+1) である．素性抽出対象となる一番文末側の単語の位置は (i+Δ) か $(i + \frac{W-1}{2})$ の大きい方の値となる[⋆1]．よって，現在位置から最も近い動的素性の位置 (i+1) と，現在位置から文末側の最も遠い素性抽出対象の単語の位置 i +max(Δ, $\frac{W-1}{2}$) の差は，max(Δ, $\frac{W-1}{2}$) − 1 となる．

これらから，最終的に，i' 番目の単語のクラスラベルから得られる動的素性を含む規則適用を行う場合は，式 (2) の範囲となる．

3.5　規則適用例

入力，“I am happy.” と図 2 にある規則を用いて説明する．また，$(W, \Delta) = (3, 1)$ という条件で適用するとする．

まず，静的素性の処理から説明する．1 番目の単語を現在位置としてチェックを開始する．$W = 3$ であるので，式 (1) に基づいて，1 番目の単語の “I” からチェックする範囲は，3 番目までの単語となる．1 番目の単語の “I” からは “I:0” が生成される．この素性は，{I:0} ⊂ $\mathcal{RC}$ なので，“I:0” は規則の部分集合であり図 2 の木構造表現内の位置である (1) が **Rs**[1] に記録される．

続いて，“I” から見て 2 番目の単語 “am” から生成される “am:1” を **Rs**[1] 中の適用状況の記録とともにチェックする．現在，(1) まで到達したという記録があるので，そこを開始位置として，“am:1” とチェックをすると，「*」が付与されたノードに到達したので，{I:0, am:1} と適合したとわかる．その後，{I:0, am:1} に対応する（クラスラベル，確信度，確信度付与位置）の情報を用いて，各単語に確信度を付与していく．“I” が基点なので，VBP の確信度である c_1 は 1 つ隣の “am” に対して，JJ の確信度である c_2 は 2 つ隣の “happy” に対して付与される．木構造内部の位置情報については子ノードが存在しないため記録しない．“I” を基点とした規則適用の最後として，“happy:2” を **Rs**[1] に記録されている (1) の位置からチェックするが，条件に合致しないため終了となる．続いて，次の単語 “am” に移動する．ここで，“am:0” が合致しないため，何も記録されずに，“am” から開始する規則

⋆1 文末側において，動的素性が利用される単語の範囲が静的素性が利用される単語の範囲より広い場合もあるためこのようになる．

適用は終了となる．残りの “happy” と “.” も同様にチェックする．“happy” から生成される “happy:0” は規則に含まれていないので，**Rs**[3] に何も記録されずに終了となる．“.” から生成される “.:0” については，(5) の位置で規則に適合するので確信度を付与するが，子ノードが存在しないので，**Rs**[4] に何も記録されずに終了となる．

すべての単語の静的素性のチェックを終えた後は，動的素性を含む規則が存在するかを，文末から文頭の方向へチェックする．すべての単語の静的素性のチェック後は，規則の部分集合に関する情報は，木構造内の到達位置として **Rs** に保持されているので，動的素性を含む規則のチェックでは，**Rs** に記録されている位置情報と各単語の動的素性を組み合わせてチェックすることで，確信度を付与できる．

今回，動的素性の適用は文末方向から文頭方向に行われるので，“.”，“happy”，“am”，“I” の順番でクラスラベルの決定がされる．これらのクラスラベルは次のようなチェックを経て決定される．

この例におけるすべての単語の静的素性のチェック後の適用状況は，$\mathbf{Rs}[1] = \{root, (1)\}$，$\mathbf{Rs}[2] = \{root\}$，$\mathbf{Rs}[3] = \{root\}$，$\mathbf{Rs}[4] = \{root\}$ である．この状況と順次生成される動的素性を用いての規則適用を説明する．

まず，最初に動的素性が生成される単語 “.” の位置である，$i' = 4$ から処理を開始する．“.” のクラスラベルは静的素性をすべて適用した時点で決定されている．まず，式 (2) によって，チェックの範囲は，2 番目から 4 番目の単語と決定される．ここで，**Rs**[2]，**Rs**[3]，**Rs**[4] 内の位置情報と，$i' = 4$ 番目の単語から得られる動的素性を含む規則が存在するかのチェックを行う．ただ，今回，範囲内の適用状況は，$\mathbf{Rs}[2] = \{root\}$，$\mathbf{Rs}[3] = \{root\}$，$\mathbf{Rs}[4] = \{root\}$ であるので，$i' = 4$ の位置から得られる動的素性に関しては，何も処理されることなく終了となる．ここまでのチェックを経て，“happy” の動的素性が “JJ” と決まったとする．

次に，3 番目の単語の “happy” のラベル “JJ” から生成される動的素性を含む規則が存在するかをチェックする．適用される範囲は，動的素性を利用する基点の位置が $i' = 3$ であるので，式 (2) によって，1 番目から 3 番目の単語となる．

まず，1 番目の単語 “I” を基点に，3 番目の単語から得られる動的素性である “$JJ:2” を用いて，**Rs**[1] 中にある木構造内部での位置情報 (1) からチェックを開始する．ここで，(1) から (3) に到達し，{I:0, $JJ:2} が合致したので，確信度を付与する．また，子ノードが存在するので到達位置 (3) を **Rs**[1] を追加する．

続いて，2 番目の単語 “am” を基点に，3 番目の単語から得られる動的素性である “$JJ:1” を用いて，2 番目の単語の適用状況が記録されている **Rs**[2] 中にある木構造内部での位置情報とチェックする．**Rs**[2] は {*root*} であり，合致せず終了となる．残りも同様に，3 番目の単語 “happy” を基点に，3 番目の単語から得られる動的素性である “$JJ:0” を用いた **Rs**[3] 中にある位置情報の {*root*} からチェックを行うが合致せず終了となる．

上記の処理が終了後に “am” のクラスラベルが決定される．ここで，“am” のクラスラベルが VBP と決定したとする．2 番目の “am” から生成される動的素性が適用される範囲は，1 番目から 2 番目の単語となる．まず，1 番目の単語 “I” を基点に，2 番目の単語 “am” から得られる動的素性である “$VBP:1” を用いて，**Rs**[1] に記録されている木構造内部での位置情報 (1) と (3) のそれぞれからチェックを開始する．ここで，(3) から (4) に到達し，{I:0, $JJ:2, $VBP:1} が合致したので，確信度を付与する．次に，残りの 2 番目の単語 “am” を基点に，2 番目の単語 “am” から得られる動的素性でを用いてチェックするが，規則は適用されない．この後に，“I” のクラスラベルが決定され，規則適用が終了となる．

3.6　圧縮規則表現の利点と欠点

圧縮規則表現の利点と欠点について述べる．圧縮規則表現の利点は，静的素性だけで構成される規則であれば，最大で規則数を $1/W$ にすることができる．欠点としては，2 つ以上の素性から構成されかつ動的素性を含む規則を圧縮規則表現に変換する場合は，動的素性の数が増加してしまう可能性があるという点にある．変換前の規則では，動的素性数は，動的素性を利用する範囲が Δ の場合は，最大でも “$\Delta \times CL$” である．ここで，CL とは，対象のタスクで判別するラベル数である．しかし，変換後の規則集合圧縮規則表現中においては，動的素性の総数は，最大で，“$(\frac{W-1}{2} + \Delta + \max(\frac{W-1}{2}, \Delta) - 1) \times CL$” まで増加する．この理由としては，各素性は，各規則の基準位置の相対位置とともに表現されるからである．

4.　素性組合せの生成調整可能な Boosting に基づく学習手法

本章では，本稿で利用する Boosting アルゴリズムを説明する．まず，4.1 節で前提条件，4.2 節で弱仮説，4.3 節で Boosting アルゴリズムについて説明する．続いて，4.4 節で，素性の組合せ生成調整可能な弱学習器を説明する．本手法では，組合せに考慮されない素性を Atomic として，組合せに考慮する素性を Non-atomic として指定する．単一の素性から構成される規則は Non-atomic と Atomic から，素性の組合せから構成される規則は Non-atomic 素性から学習する．これにより，素性の組合せの生成を調整可能にする．

4.1　前 提 条 件

$\mathcal{X}$ を事例集合とし，$\mathcal{Y}$ をラベル集合 $\{-1, +1\}$ とする．$\mathcal{F} = \{f_1, f_2, \ldots, f_M\}$ を M 種類

の素性とし，本稿では，素性は文字列で表現されるとする．

$S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$ を学習事例集合とし，各事例 $\mathbf{x}_i \in \mathcal{X}$ は，$\mathcal{F}$ 中の素性の集合で表現されるとする．また，学習事例のような素性の集合を素性集合と呼ぶことにする．また，$y_i \in \mathcal{Y}$ をクラスラベルとする．学習の目的は，次のマッピングを S から導出することである．

$$F : \mathcal{X} \to \mathcal{Y}.$$

$|\mathbf{x}_i|$（$0 < |\mathbf{x}_i| \leq M$）を素性集合 $\mathbf{x}_i$ に含まれる素性の数とし，$\mathbf{x_i}$ のサイズと呼ぶことにする．また，$x_{i,j} \in \mathcal{F}$（$1 \leq j \leq |\mathbf{x}_i|$）を $\mathbf{x}_i$ に含まれる j 番目の素性とする．本稿では，サイズ k の素性集合を k-素性集合と呼ぶ．また，2 つの素性集合，$\mathbf{x}_i$ と $\mathbf{x}_j$ において $\mathbf{x}_j$ が $\mathbf{x}_i$ のすべての素性を含む場合を，$\mathbf{x}_i$ は $\mathbf{x}_j$ の部分集合と呼び，次のように表記する．

$$\mathbf{x}_i \subseteq \mathbf{x}_j.$$

4.2 弱 仮 説

続いて，Boosting に基づく学習のための弱仮説を定義する．Boosting とは，学習事例の重みを変化させることで，複数の弱仮説を作成し，それらの組合せによる最終的な仮説を作成する手法である．まず，弱仮説の定義にあたり，$\mathbf{f}$ を素性集合で表現される規則，c を実数値である規則の確信度と呼ぶことにする．本稿では，the real-valued predictions and abstaining [20] に基づいて，規則が適用された場合にだけ確信度を付与する弱仮説を定義する．$\mathbf{x}$ を弱仮説への入力の素性集合とする．ここで，素性集合を分類するための弱仮説を次に定義する．

$$h_{\langle \mathbf{f}, c \rangle}(\mathbf{x}) = \begin{cases} c & \mathbf{f} \subseteq \mathbf{x} \\ 0 & \text{otherwise} \end{cases}.$$

4.3 Boosting アルゴリズム

本稿では，複数の規則を生成する弱学習器を扱うことが可能な Boosting アルゴリズムを用いる[9]．このアルゴリズムは，英語品詞タグ付け，Text Chunking のタスクで，高い精度を保持しつつ，高速な学習が行えることが報告されている[9]．本稿では，また，計算の高速化を目的に，文献 5)，9) にある，重みの正規化を行わない版の Boosting を用いる．**図 4** に概要を載せる．この Boosting アルゴリズムでは，次に定義される R 種類の弱仮説から構成される最終仮説 F を導出する．

$$F(\mathbf{x}) = sign\left(\sum_{r=1}^{R} h_{\langle \mathbf{f}_r, c_r \rangle}(\mathbf{x})\right)$$

$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$: $\mathbf{x}_i \subseteq \mathcal{X}$,$y_i \in \{\pm 1\}$
smoothing value ε =1
規則番号 r: 初期値は 1
$\{w_{r,1}, \ldots, w_{r,m}\}$：事例の重み．$r$=1 の時点では重みは 1 とする．
Initialize: **While** $(r \leq R)$
 ## weak-learner が $(S, \{w_{r,i}\}_{i=1}^{m})$ を入力とし，ν 種類の規則 $\{\mathbf{f}_j\}_{j=1}^{\nu}$ を学習
 $\{\mathbf{f}_j\}_{j=1}^{\nu} \leftarrow$ **weak-learner**$(S, \{w_{r,i}\}_{i=1}^{m})$;
 ## 各規則を使って各事例の重みを更新
 Foreach $\mathbf{f} \in \{\mathbf{f}_j\}_{j=1}^{\nu}$
  $\mathbf{f}_r = \mathbf{f}$; ## r 番目の規則として記録
  $c_r = \frac{1}{2}\log(\frac{W_{r,+1}(\mathbf{f}_r)+\varepsilon}{W_{r,-1}(\mathbf{f}_r)+\varepsilon})$ ## r 番目の規則の確信度を計算
  For $i = 1, \ldots, m$: $w_{r+1,i} = w_{r,i}\ \exp(-y_i h_{\langle \mathbf{f}_r, c_r \rangle}(\mathbf{x}_i))$ ##事例の重み更新
  r++; ## 規則番号を 1 増加
 endForeach
endWhile
Output: $F(\mathbf{x}) = sign(\sum_{r=1}^{R} h_{\langle \mathbf{f}_r, c_r \rangle}(\mathbf{x}))$

図 4　複数弱仮説を学習する弱学習器を用いる Boosting の概要
Fig. 4　A generalized version of a boosting algorithm using weak learner for learning several weak hypothesis.

ここで，$sign$ は，引数の値が 0 以上の場合は 1 を，それ以外は，-1 を返す関数とする．本稿で用いる Boosting アルゴリズムでは，複数の規則を学習事例 $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ と，各学習事例の重み $\{w_{r,1}, \ldots, w_{r,m}\}$ を用いて学習する弱学習器を用いる．$w_{r,i}$ は，Boosting アルゴリズムによって決められる，i 番目の学習事例の $r-1$ 種類の規則を生成した後の重みである．各事例の初期の重みは，1 とする．

ここで，$0 < w_{r,i}$，$1 \leq i \leq m$，$1 \leq r \leq R$ である．このような入力から，弱学習器は ν 種類の規則を選択する．弱学習器は最大の $gain$ を持つ素性集合を r 番目の規則として選択し，最終的に上位 ν に入る $gain$ を持つ ν 種類の素性集合を規則として選択する．

続いて，弱学習器が ν 種類を選択後に，Boosting 学習器は，各規則の確信度を計算し，事例の重みを更新する．まず，選択された ν 種類の規則では最初の規則であり，全体では r 番目の規則である $\mathbf{f}_r$ の確信度 c_r は次のように計算する．

$$c_r = \frac{1}{2}\log\left(\frac{W_{r,+1}(\mathbf{f}_r)+\varepsilon}{W_{r,-1}(\mathbf{f}_r)+\varepsilon}\right)$$

ここで，$W_{r,y}(\mathbf{f}_r)$ $(y \in \{-1,+1\})$ は，

$$W_{r,y}(\mathbf{f}_r) = \sum_{i=1}^{m} w_{r,i}[[\mathbf{f}_r \subseteq \mathbf{x}_i \wedge y_i = y]],$$

である．ここで，$[[\pi]]$ は命題 π を満たすなら 1, それ以外は 0 とする．また，ε は，$W_{r,+1}(\mathbf{f})$ あるいは $W_{r,-1}(\mathbf{f})$ が 0 になるのを防ぐために導入する smoothing value である[19]．本稿の実験では，ε に 1 を用いた．$\mathbf{f}_r$ の c_r を計算後に，事例の重みを次の式を用いて更新する．

$$w_{r+1,i} = w_{r,i}\exp(-y_i h_{\langle \mathbf{f}_r, c_r\rangle}(\mathbf{x}_i))$$

重み更新後に，$h_{\langle \mathbf{f}_r, c_r\rangle}$ が F に対して，r 番目の弱仮説として追加される．続いて残りの規則の確信度の計算を行う．各規則の確信度の計算では，1 つ前の弱仮説で更新された事例の重みを用いる．たとえば，$\mathbf{f}_{r+1}$ の確信度 c_{r+1} を計算する場合は，$\{w_{r+1,1},\ldots,w_{r+1,m}\}$ を用いる．この Boosting 学習器は R 種類の弱仮説を獲得するまで学習を続ける．

このように，各イテレーションで複数の弱仮説を利用する場合でも，文献 9) で収束することが証明されており，学習時間の改善が行えることも報告されている．すべての規則の処理が終えたら，次のイテレーションに移る．もし，各イテレーションで学習する規則の数を 1 とすれば，従来の Boosting アルゴリズムと同様になる[5]．

4.4　素性の組合せ調整可能な規則学習器

本稿では，文献 9) で用いられている弱学習器を，素性の組合せ調整可能なように拡張する．文献 9) の弱学習器は，部分規則候補集合から複数規則を学習可能な弱学習器である．

図 5 に拡張後の弱学習器について説明する．まず，この弱学習器では，1 つの素性から構成される規則候補の集合である F_1 を受け取る．生成される素性組合せによる規則候補は，初期に与えられた素性集合 F_1 中の候補を少なくとも 1 つは含む素性集合とする．このように初期の素性候補を制限し，組合せもそれらの素性を含むものに限定するのは，各イテレーションで候補を絞り込むことで，高速化を実現するためである[9]．

この弱学習器は，規則候補の中から上位 ν 以内の以下に定義される $gain$ を持つ素性集合を規則の候補集合から選択する．

$$gain(\mathbf{f}) \stackrel{\text{def}}{=} |\sqrt{W_{r,+1}(\mathbf{f})} - \sqrt{W_{r,-1}(\mathbf{f})}|$$

$\mathbf{f}$ は素性集合である．この基準は，Boosting のトレーニングエラー上限値の削減に関係して

```
## F_k : k–素性集合
## S : 学習事例
## W_r : 事例の重み {w_{r,1}, ..., w_{r,m}}
## R_o : 上位 ν 規則 (素性集合)
## R_{k,ω} : 候補生成に利用する上位 ω k-素性集合.
## selectNBest(R, n, S, W_r): 上位 n を R から gain を基に選択
## FN : 組合せ生成に利用される Non-atomic 素性.
procedure weak-learner(F_k, S , W_r, ν, ω)
 ## 現時点での上位 ν の 素性集合 を規則として選択
 R_o = selectNBest( R_o ∪ F_k,  ν, S, W_r);
 If (ζ ≤ k) return R_o;  ## サイズによる枝刈り
 ## 候補生成のための F_k 中の上位 ω の素性集合を選択
 R_{k,ω} = selectNBest(F_k, ω, S, W_r);
 τ = min_{f∈R_o} gain(f);  ## ν 番目の規則の gain. 枝刈り用.
 Foreach ( f_k ∈ R_{k,ω})
  ## gain の上限値による枝刈り
  If ( u(f_k) < τ) continue;
  Foreach (f ∈ FN)   ## 候補生成
   F_{k+1} = (F_{k+1} ∪ gen(f_k, f));
  endForeach
 endForeach
 return weak-learner(F_{k+1}, S, W_r, ν, ω);
```

図 5　素性の組合せ生成が調整可能な弱学習器の概要図

Fig. 5　An overview of our weak learner. This weak learner controls generation of combinations of features.

定められている[5),9),20)]．この式を最大にする規則を選択することは，その時点での重みにおいて，Boosting のトレーニングエラー上限値を最も減少させる規則を候補から選択することと等価である．この弱学習器では，与えられた候補の中から，Boosting のトレーニングエラー上限値を減少させるかどうかという観点で，規則を選択していることになる．

まず，1–素性集合で構成される F_1 中の候補から規則選択を開始する．k–素性集合 $(1 < k)$

から構成される規則候補集合を生成する場合，弱学習器は F_{k-1} 中の上位 ω の内の $gain$ を持つ $(k-1)$–素性集合を選択し，その選択された素性集合と共起する素性を加えた k–素性集合を生成する．

ここまでは，文献 9) の弱学習器と同じであるが，素性の組合せを調整可能とするために次の変更を加える．まず，本稿では $\mathcal{FA}$ と $\mathcal{FN}$ という 2 種類の素性集合を用いることにする．ここで，$\mathcal{FA}$ と $\mathcal{FN}$ は，Atomic 素性の集合と Non-atomic 素性の集合である．2 つ以上の素性から構成される規則候補を生成する場合には，$\mathcal{FN}$ 中の Non-atomic 素性だけが利用され，1 つの素性から構成される規則候補を生成する場合は，$\mathcal{FA}$ 中の Atomic 素性と $\mathcal{FN}$ 中の Non-atomic 素性が対象となる．これらの関係は，$\mathcal{F} = \mathcal{FA} \cup \mathcal{FN}$ となる．また，$\mathcal{FA}$ と $\mathcal{FN}$ はあらかじめ定義されているものとする．この定義に基づいて，素性の組合せ生成を調整可能とする．

ここで，規則候補生成の例を示す．もし，$\mathcal{FA} = \{A, B, C\}$，$\mathcal{FN} = \{a, b, c\}$ という素性を用いる場合は，次の規則候補が対象となる．$\{A\}$，$\{B\}$，$\{C\}$，$\{a\}$，$\{b\}$, $\{c\}$, $\{a, b\}$，$\{b, c\}$，$\{a, b, c\}$.

素性の組合せは次に定義される gen を用いて生成する．$\mathbf{f}' = \mathbf{f} + f$ を素性 f と k–素性集合である $\mathbf{f}$ から $(k+1)$–素性集合 $\mathbf{f}'$ を生成する意味とする．また，$ID(f)$ を f に対応する整数値とし id と呼ぶ．また，ϕ を 0-素性集合とする．ここで，gen は次のように定義する．

$$gen(\mathbf{f}, f) = \begin{cases} \phi & \text{if } (\ \mathbf{f} \subseteq \mathcal{FA}\ ) \\ \mathbf{f} + f & \text{if } ID(f) > \max\limits_{f' \in \mathbf{f}} ID(f')\ . \\ \phi & \text{otherwise} \end{cases}$$

この gen によって，2 つ以上の素性から構成される規則候補は，重複なく Non-atomic 素性だけから構成されるようになる．本稿では，文献 8) に従って，学習事例中での素性の出現頻度に基づいて id を付与し，低頻出な素性ほど小さい id を付与することにする．もし，同じ頻度の素性がある場合は，id は素性の辞書順で付与することにする．本稿でもさらなる高速化のため次の枝刈り手法を用いる．

- **サイズに基づく枝刈り**：サイズが ζ 以下の候補規則だけを対象とする．
- **gain の上限値に基づく枝刈り**：次に定義される gain の上限値を用いる．
  $u(\mathbf{f}) \stackrel{\text{def}}{=} \max(\sqrt{W_{r,+1}(\mathbf{f})}, \sqrt{W_{r,-1}(\mathbf{f})})$.
  $\mathbf{f} \subseteq \mathbf{f}'$ を満たす素性集合 $\mathbf{f}' \subseteq \mathcal{F}$ において，$0 \leq W_{r,y}(\mathbf{f}') \leq W_{r,y}(\mathbf{f})$（$y \in \{\pm 1\}$）であることから，$gain(\mathbf{f}')$ は $u(\mathbf{f})$ 以下になることが成り立つ．よって，もし，$u(\mathbf{f})$ が，今まで評価した規則候補のうち ν 番目の候補の $gain$ である τ 以下なら $\mathbf{f}$ を含む候補の計算は行わなくてもよいことが分かる．

4.5　組合せ調整可能な弱学習器を用いた Boosting

図 6 に本 Boosting 学習手法である AdaBoost for a weak-learner learning *S*everal rules from *D*istributed *F*eatures consisting of Atomic and Non-atomic（*AdaBoost.SDFAN*）の概要を載せる．この手法は，文献 9) で提案された手法を，Atomic 素性と Non-atomic 素性の指定を受けつけるように拡張したものである．

まず，クラス分布の差を反映させるために，デフォルト規則として，$\frac{1}{2}\log(\frac{W_{+1}}{W_{-1}})$ を用いて，初期の事例の重みを決定する．ここで，$W_y = \sum_{i=1}^{m}[[y_i = y]]$（$y \in \{\pm 1\}$）である．続いて，本手法では，$|B|$ 個のバケットを生成する．バケットとは，素性の部分集合である．このように素性を複数のバケットに分割するのは，各イテレーションで利用する素性を絞り込むことで，高速化を実現するためである[9]．

$|B|$ 個のバケットを生成するために *W-dist* という方法を用いる[9]．素性の分割方法としては，ランダム分割，出現頻度を用いた分割などいくつかの方法が提案されている[4),8)]．これに対し，*W-dist* では，各素性 $f \in \mathcal{F}$ の重みを Boosting アルゴリズムが事例に対して決定する重みを使う次の式で計算する．

$$W_r(f) = \sum_{i=1}^{m} w_{r,i}[[\{f\} \subseteq \mathbf{x}_i]]$$

続いて，各素性の重みを計算後に *W-dist* は素性の重みの降順にソートし，各素性を順番にバケットに分配する．分配されるバケットは，各素性のソート後の順番をバケット数で割った余りで決定する．ソート後の n 番目の素性は，n をバケット数 $|B|$ で割った余り番目のバケットに分配する．

続いて，規則学習を行う．規則学習の各イテレーションでは，$|B|$ 種類のバケットのうち 1 つを選択して，初期の 1–素性集合 の規則候補である F_1 として弱学習器に与える．弱学習器はそのバケット内の素性およびそのバケット内の素性を含む素性の組合せを候補として規則学習を行う．規則学習後は，各規則の確信度を計算し，事例の重みの更新を行う．

すべてのバケットの処理が終われば，再度，素性の重みを計算し，バケットに素性を再度分割する．この方法では，ランダムな分割では学習の再現性がないという問題を解決しつつ，バケットによる偏りをなくすために，このように再分割を行っている[9]．指定された個

$S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$: $\mathbf{x}_i \subseteq \mathcal{X}$, $y_i \in \{\pm 1\}$
$W_r = \{w_{r,i}\}_{i=1}^{m}$: r 種類の規則学習後の事例の重み
$|B|$: バケットサイズ．$B = \{B[0], \ldots, B[|B|-1]\}$
b, r : 現在のバケット番号と規則番号
ν, ω : 各ラウンドで学習する規則数と組合せ候補生成の数
procedure **AdaBoost.SDFAN**()
$B =$ **distFT**$(S, |B|)$; ## 素性を $|B|$ 個のバケットに分割．B が分割結果
事例の重みと変数を初期化:
$r = 1$; $b = 0$; $c_0 = \frac{1}{2}\log(\frac{W_{+1}}{W_{-1}})$;
For $i = 1, \ldots, m$: $w_{1,i} = \exp(-y_i c_0)$;
While $(r \leq R)$ ## R 個の規則を学習
$B[b]$ を基に ν 個の規則 $\mathcal{R}$ を選択．バケット番号 b を 1 増加
$\mathcal{R} =$ **weak-learner**$(B[b], S, W_r, \nu, \omega)$; b++;
Foreach $(\mathbf{f} \in \mathcal{R})$ ##各規則で事例の重みを更新
$\mathbf{f}_r = \mathbf{f}$; ## r 番目の規則として記録
$c_r = \frac{1}{2}\log(\frac{W_{r,+1}(\mathbf{f}_r)+\varepsilon}{W_{r,-1}(\mathbf{f}_r)+\varepsilon})$ ## r 番目の規則の確信度を計算
For $i = 1, \ldots, m$: $w_{r+1,i} = w_{r,i} \exp(-y_i h_{\langle \mathbf{f}_r, c_r \rangle}(\mathbf{x}_i))$ ##事例の重み更新
r++; ## 規則番号を 1 増加
endForeach
If $(b == |B|)$ ## 全てのバケットチェック後は再分割
$B =$ **distFT**$(S, |B|)$; b=0;
endIf
endWhile
return $F(\mathbf{x}) = sign(c_0 + \sum_{r=1}^{R} h_{\langle \mathbf{f}_r, c_r \rangle}(\mathbf{x}))$

図 6　AdaBoost.SDFAN の概要
Fig. 6　An overview of AdaBoost.SDFAN.

数の規則が選択された時点で学習は終了となる．

5. 実験設定

本章では，本提案手法を評価する英語品詞タグ付けおよび Text Chunking について説明する．

5.1 英語品詞タグ付け

英語品詞タグ付けの評価に，Penn Wall Street Journal treebank[15] を用いる．評価のためにこの treebank を，文献 2) に従って，学習（sections 0–18），開発データ（sections 19–21），評価（sections 22–24）の 3 つに分割する．

本評価では，次の素性を用いる．

- 素性抽出対象範囲となる W 単語に出現する，単語の表記，単語の表記中のアルファベットをすべて大文字に変換した結果
- 判別対象の単語から文末側 Δ 単語に付与された品詞ラベル
- 現在位置の単語が，ハイフンを含むか，数字を含むか，大文字アルファベットを含むか，すべて文字が大文字アルファベットであるか，すべて文字が小文字アルファベットであるか
- 単語の先頭および単語の終わりからの 1 から 4 文字
- 素性抽出対象範囲となる W 単語のタグ候補

ここでの，英語品詞タグ付けにおけるタグ候補とは，各単語にどれくらいの頻度でその品詞が付与されうるか，という情報である．本実験では，英語品詞タグ付け用のタグ候補は，ラベルなしコーパスに自動品詞タグ付けした結果から収集した情報を用いる．文献 9) の実験結果で，タグ候補を用いることで，精度改善が得られていることから，本実験でも同様の素性を用いることにする．英語品詞タグ付けにおける各単語のタグ候補を収集するために，CoNLL 2003 の shared task である English Named Entity recognition のために用意された，自動的に品詞タグ付けされたラベルなしコーパスを用いる．各単語のタグ候補としては，10 回以上付与されている品詞タグを用いる[⋆1]．各タグ候補は各単語に付与された品詞の頻度情報 fq が，$10 \leq fq < 100$，$100 \leq fq < 1{,}000$，$1{,}000 \leq fq$，いずれかの範囲に入るかどうかで表現する

たとえば，‘work’ という単語に対して NN が 2,000 回付与されていたら，“1,000 ≤ NN” のように表現する．‘work’ が現在位置の単語であれば，1,000 ≤ NN をタグ候補として用いる．‘work’ が現在位置の単語の文末側方向の隣に出現した場合は，1,000 ≤ NN が右隣の単語のタグ候補として追加される．

⋆1 http://www.cnts.ua.ac.be/conll2003/ner/（参照 2010-10-04）

5.2 Text Chunking

本稿では，CoNLL-2000 shared task のために用意されたデータセットを用いる[⋆1]．このタスクは，NP，VP，PP といった 10 種類のチャンクを判別するタスクである．このデータセットは，Penn Wall Street Journal treebank の一部分から構成され，学習（sections 15–18），評価（section 20）となっている．また，この treebank の section 21 を，文献 9) と同様に開発データとして用いる[⋆2]．

各チャンクは 1 つあるいは 2 つ以上の単語で構成される可能性がある．そこで，単語のチャンクを表現するために，**IOE2** 表現を用いることにした[⋆3]．各チャンクは次のラベルを用いて表現される．E-X は，クラス X のチャンクの最後の単語に用いられる．I-X は，クラス X のチャンクの最後以外の単語に用いられる．O はチャンク以外の単語に用いられる．たとえば，

“[He] (NP) [reckons] (VP) [the current account deficit] (NP)...”

は，IOE2 表現で次のように表現される．

“He/E-NP reckons/E-VP the/I-NP current/I-NP account/I-NP deficit/E-NP...”

本実験では，次の素性を用いる．

- 素性抽出対象範囲となる W に出現する単語の表記と品詞ラベル
- 判別対象の単語から文末側 Δ 単語に付与されたチャンクラベル
- 素性抽出対象範囲となる W 単語のタグ候補に基づく次の素性
  - タグ候補
  - 品詞タグ付けと同様に，頻度情報とともに表現されたタグ候補
  - 各単語のタグ候補の頻度に基づくランキング

ここでの，Text Chunking におけるタグ候補とは，各単語はどのような Text Chunk タグが付与されうるか，各単語にどれくらいの頻度でその Text Chunk タグが付与されているか，対象単語に付与されるタグ候補の中での順位，という情報を，ラベルなしコーパスに自動品詞タグ付けした結果から収集した情報である．各単語のタグ候補を集めるために，英語品詞タグ付けと同様に，CoNLL 2003 の shared task の English Named Entity recognition のために用意された同様のコーパスを用いた．こちらのラベルなしコーパスには，Text Chunk タグが自動付与されている[⋆4]．

表 1　実験データ．POS と ETC は英語品詞タグ付け，Text Chunking の意味．♯ of S，♯ of cl，M はそれぞれ，学習事例数，クラスラベル数，(W, Δ) 別の素性の異なり数である

Table 1　Training data for experiments. POS and ETC indicate POS tagging and text chunking. ♯ of S, ♯ of cl and M indicate the number of samples, the number of classes in each data set and the distinct number of feature types for each pair of (W, Δ).

			M (W, Δ)		
data	♯ of S	♯ of cl	$(3, 1)$	$(5, 2)$	$(7, 3)$
POS	912,344	45	283,979	440,725	593,065
ETC	211,727	22	56,917	93,333	128,651

次にタグ候補の例を示す．たとえば，“work” が I-NP として 2,000 回，E-VP として 100 回ラベル付けされていたとする．この場合，“work” のタグ候補素性として，1,000 ≤ I-NP，100 ≤ E-VP < 1,000，rank:I-NP = 1，rank:E-NP = 2，candidate = I-NP，candidate = E-VP が生成される．

5.3 実験条件

本実験では，4 章で説明した Boosting アルゴリズムのパラメータとして，規則学習数 $R = 200{,}000$，バケットサイズ $|B| = 1{,}000$，各イテレーションで学習する規則数 $\nu = 10$，組合せ規則候補に利用する幅 $\omega = 10$，素性の組合せ最大長制約 $\zeta = \{1, 2, 3\}$ を用いる．また，素性抽出に関してのパラメータとして，$(W, \Delta) = \{(3, 1), (5, 2), (7, 3)\}$ を用いる．**表 1** に学習に用いるデータの学習事例数，クラスラベル数，素性数を示す．

また，本実験では，素性の組合せ生成調整の効果を測定するために，−Atomic と +Atomic の次の 2 パターンの実験を行う．

- −Atomic: すべての素性を Non-atomic とし，素性の組合せはすべての素性から生成する．
- +Atomic: Atomic と Non-atomic の 2 種類を用い，素性の組合せは Non-atomic から生成する．

Atomic に指定する素性の選択方法としては，出現頻度が多い素性ほど多くの素性の組合せに含まれることから，各タスクにおいて出現頻度が多い種類の素性を指定することにした．英語品詞タグ付け実験では，Atomic として，単語の先頭の 1 から 4 文字，単語の終

⋆1 http://lcg-www.uia.ac.be/conll2000/chunking/（参照 2010-10-04）

⋆2 開発データの生成のために http://ilk.uvt.nl/~sabine/chunklink/chunklink_2-2-2000_for_conll.pl（参照 2010-10-04）を用いた．

⋆3 **IOE2** 表現に基づく Text Chunker は，先行研究[10] において良い性能を示していることから採用した．

⋆4 タグ候補を集めるためにこのコーパス内の IOB2 表現を IOE2 表現に変換し，各単語に 10 回以上付与されたタグラベルをタグ候補生成のために利用した．

わりの 1 から 4 文字，タグ候補を指定した．また，Text Chunking の実験では，タグ候補を Atomic として指定した．二値分類を対象とする AdaBoost.SDFAN を，マルチクラス問題に対応させるために，one-vs-the-rest 法を用いた．また，Text Chunking においては，IOE2 の定義にあった適切なラベル列候補の中から，Viterbi 探索を用いて，入力へ付与するラベル列を決定する．Viterbi 探索を用いて解を求めるために，Boosting アルゴリズムにて学習した各分類器が出力する確信度を 0 から 1 の範囲に次の sigmoid 関数を用いてマッピングする．

$$s(X) = 1/(1 + \exp(-\beta X))$$

ここで，X は AdaBoost.SDFAN で学習した規則によって決定される $\mathbf{x}$ に対する各クラスラベルの確信度である[⋆1]．最終的なラベル列は，この変換後の値の log 値の和を最大にするタグ列とする．

6. 実験・評価

6.1 精度評価

表 2 に英語品詞タグ付け，Text Chunking の評価結果を載せる．F-measure ($F_{\beta=1}$) は次のように計算する[⋆2]．

$$F_{\beta=1} = 2 \times Recall \times Precision/(Recall + Precision)$$

表 2 の数値は，開発データ上を用いて最も高い $F_{\beta=1}$ を示す規則数を決定し，評価データに適用して得られた $F_{\beta=1}$ である．

表 2 から，英語品詞タグ付け，Text Chunking ともに，素性の組合せを考慮することで精度が改善されていることが分かる．また，今回の評価では，すべての素性の組合せを考慮せずに，部分的な素性から組合せを生成する “ +Atomic ” により学習した場合でも，すべての素性からの組合せを考慮する “−Atomic ” と同等の精度が得られていることが分かる．

表 2　テストデータでの評価結果．数値は $F_{\beta=1}$ である．−Atomic は Atomic の利用なしで，+Atomic は Atomic の利用ありの意味．太字は Atomic の利用ありとなしで，それぞれでの最も高い値
Table 2　$F_{\beta=1}$ on test data. −Atomic indicates results without Atomic. +Atomic indicates results with Atomic. Each bold number indicates the best $F_{\beta=1}$ of −Atomic or +Atomic.

英語品詞タグ付け						Text Chunking				
		−Atomic		+Atomic			−Atomic		+Atomic	
$(W, \Delta)/\ \zeta$	1	2	3	2	3	1	2	3	2	3
$(3, 1)$	96.81	97.09	97.05	97.04	96.98	92.40	93.87	93.69	93.91	93.82
$(5, 2)$	96.96	97.30	97.30	97.25	97.26	92.87	94.31	94.14	**94.34**	94.31
$(7, 3)$	96.99	**97.36**	97.30	**97.33**	97.29	93.09	**94.32**	94.11	94.12	94.11

⋆1 本実験では，$\beta = 5$ を用いた．
⋆2 英語品詞タグ付けでは，単語単位で，Text Chunking ではチャンク単位で，$F_{\beta=1}$ を計算する．

6.2 処理速度評価

表 3 に英語品詞タグ付け，Text Chunking の処理速度を載せる．処理速度の比較では，1 秒間に処理される単語数を用いる[⋆3]．まず，$\zeta = \{1, 2\}$ および “−Atomic” という，Atomic 素性を用いない場合においての比較では，圧縮規則表現形式に変換することで，処理速度改善が得られていることが分かる．これらの結果から，$\zeta = 2$ という組合せが含まれる規則集合を用いた場合より，$\zeta = 1$ という組合せを考慮しない場合において，速度が改善していることが分かる．$\zeta = 1$ という組合せが含まれない規則集合を用いる場合であれば，英語品詞タグ付けの $(W, \Delta) = (7, 3)$ で比較すると，約 2.4 倍高速である．$\zeta = 2$ という組合せが含まれる規則集合を用いた場合では，英語品詞タグ付けの $(W, \Delta) = (7, 3)$ で比較すると，約 1.2 倍高速である．

しかし，$(\zeta = 3, -Atomic)$ という条件で学習した規則集合を圧縮規則表現に変換した場

表 3　英語品詞タグ付け，Text Chunking の処理速度．表中の各数値は 1 秒間に処理される単語数であり，3 回実行後の平均．太字は表 2 中における利用ありとなしでの最も高い $F_{\beta=1}$ に対応する処理単語数．
Table 3　Tagging and Chunking Speed. Each number is average number of processed words per second on three times measurements. Each bold number corresponds to best $F_{\beta=1}$ of −Atomic or +Atomic in Table 2.

英語品詞タグ付け										
	圧縮規則表現利用なし					圧縮規則表現利用あり				
	−Atomic			+ Atomic		−Atomic			+ Atomic	
$(W, \Delta)/\ \zeta$	1	2	3	2	3	1	2	3	2	3
$(3, 1)$	9,477	4,023	2,505	5,669	4,303	19,467	4,258	2,013	6,343	3,508
$(5, 2)$	8,118	2,564	1,445	4,301	2,705	1,8261	2,807	1,102	4,549	1,767
$(7, 3)$	6,615	**1,842**	1,007	**2,940**	2,242	15,658	**2,195**	754	**3,084**	1,300
Text Chunking										
	圧縮規則表現利用なし					圧縮規則表現利用あり				
	−Atomic			+ Atomic		−Atomic			+ Atomic	
$(W, \Delta)/\ \zeta$	1	2	3	2	3	1	2	3	2	3
$(3, 1)$	14,510	3,995	1,036	13,975	12,221	27,705	4,282	863	19,496	16,169
$(5, 2)$	11,266	1,681	401	**9,571**	7,018	25,471	2,477	352	**13,692**	8,475
$(7, 3)$	9,434	**961**	230	6,849	4,595	23,338	**1,758**	206	10,058	5,701

⋆3 評価には，3.6 GHz の Intel Xeon CPU と 10 GB memory のメモリを搭載した計算機を用いた．

合は，変換前と比較し，速度が低下している．この速度低下の理由としては，次の 2 点が考えられる．この圧縮規則表現に変換することで，規則評価回数は，最大で $1/W$ にすることができる．したがって，圧縮規則表現処理速度を線形で改善することが可能であるといえる．しかし，素性の組合せは，指数的に増加するため，複数の素性から構成される規則を変換する場合は，効果が軽減すると予想される．もう 1 つの理由としては，動的素性を利用している場合は，3.4 節で述べたように，圧縮規則表現に変換することで，動的素性が利用される回数が増加してしまう可能性があり遅くなっていると考えられる．

一部の素性を Atomic として指定して学習した規則を圧縮規則表現に変換することで，さらなる速度改善が得られていることが分かる．圧縮規則表現に変換した規則を用いない Text Chunking での評価結果であれば，同一の ζ，W，Δ のパラメータで，Atomic の利用ありなしで比較すると，たとえば，($\zeta = 3$，$W = 7$，$\Delta = 3$，$-$Atomic) による Text Chunker の処理数は 1 秒間に 230 単語，($\zeta = 3$，$W = 7$，$\Delta = 3$，$+$Atomic) による Text Chunker の処理速度は 1 秒間に 4,595 単語であり，約 20 倍高速である．このように，本手法による素性の組合せ生成の調整により処理速度改善が行えたことが分かる．

また，($\zeta = 3$，$W = 7$，$\Delta = 3$，$+$Atomic) で学習した規則を圧縮規則表現に変換した規則を用いた場合の Text Chunker の処理数 1 秒間に 5,701 単語，($\zeta = 3$，$W = 7$，$\Delta = 3$，$-$Atomic) で学習した規則を圧縮規則表現変換せずに用いる Text Chunker の処理数 1 秒間に 230 単語であり，約 25 倍高速である．このように，本手法による素性の組合せ生成の調整と圧縮規則表現への変換を実施することでも処理速度改善のために有効であることが分かる．また，表 2 にあるように，一部の素性を Atomic に指定した場合でも精度に大きな差がないことから，圧縮規則表現への変換および，本手法による素性組合せの調整を行うことで，精度を保持したまま大幅な速度改善の可能性があることが分かる．

6.3 SVMs に基づく品詞タグ付けと Text Chunking との比較

本手法を，SVMs に基づく品詞タグ付け，Text Chunking と比較を行った．SVMs に基づく分類器の高速化のためには，Polynomial Kernel Expanded（PKE）という手法が提案されている[11]．

PKE は，暗黙的に素性の組合せを考慮することができる Polynomial Kernel に基づく分類器を近似の線形分類器に変換する方法であり，English Base NP Chunking, 日本語単語分割，日本語係り受け解析というタスクで精度を保持したまま，30 倍から 300 倍の速度改善が得られると報告されている[11]．

本比較での SVMs においては，先行研究[9),10)] で高い精度を示している二次の Polynomial Kernel を用いた．素性は，本 Boosting に基づく実験と同じ素性を用いている．この比較では，静的素性と動的素性のパラメータとして，高い精度を残した $\{W = 7, \Delta = 3\}$ を英語品詞タグ付けに，$\{W = 5, \Delta = 2\}$ を Text Chunking に用いた．SVMs の学習パラメータの 1 つである soft margin parameter には 1 を用いた．本評価では，SVMs に基づく Chunker である YamCha に実装されている PKE 機能を用いる[⋆1]．両タスクにおいて，PKE のパラメータ σ は，素性の頻度による枝刈りを用いずに，$\{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$ から選択した．最終の値は，評価データ上で最も高い精度 $F_{\beta=1}$ を示した σ を選択した．

実験結果を**表 4** に載せる．まず，PKE による変換前の SVMs に基づく品詞タグ付け，Text Chunking の精度（$F_{\beta=1}$）を評価した結果，英語品詞タグ付けが 97.32，Text Chunking が 94.31 であった．ただし，選択された σ による PKE 適用後は，英語品詞タグ付けが 96.96，Text Chunking が 94.30 と若干低い精度となった．同様の静的素性と動的素性のパラメー

表 4　SVMs との比較．# of words は 1 秒間に処理した単語数を意味する．SVMs の精度は丸括弧の外が PKE 適用後，丸括弧の中が PKE 適用前，英語品詞タグ付けは $\{W = 7, \Delta = 3\}$，Text Chunking は $\{W = 5, \Delta = 2\}$ による結果．素性の組合せ，Polynomial Kernel の次元数は 2．Boosting とは，圧縮規則表現の利用も Atomic の指定も行わない場合．圧縮規則表現とは圧縮規則表現を利用し，Atomic の指定を行わない場合．圧縮規則表現+Atomic とは両方利用した場合

Table 4　Comparison with SVMs. # of words indicates number of words processed by each parse per second. We show the accuracy of SVMs after applying PKE is the outside of parentheses, and the accuracy of SVMs before applying PKE is the inside of parentheses, We use $\{W = 7, \Delta = 3\}$ for English POS tagging, and $\{W = 5, \Delta = 2\}$ for Text Chunking. The parameter for maximum number of combination of features and polynomial kernel degree is 2. Boosting indicates results obtained without compressed rule representation and Atomic. The compressed rule representation indicates results obtained with only compressed rule representation compressed rule representation+Atomic indicates results obtained with compressed rule representation and Atomic.

英語品詞タグ付け				
	SVMs	Boosting	圧縮規則表現	圧縮規則表現+Atomic
# of words	1,347	1,842	2,195	3,084
$F_{\beta=1}$	96.96 (97.32)	97.36		97.33
Text Chunking				
	SVMs	Boosting	圧縮規則表現	圧縮規則表現+Atomic
# of words	1,535	1,681	2,477	13,692
$F_{\beta=1}$	94.30 (94.31)	94.31		94.34

⋆1 http://www.chasen.org/~taku/software/yamcha/（参照 2010-10-04）

タで，一部を Atomic に指定した場合の本手法の結果は，英語品詞タグ付けが 97.33，Text Chunking が 94.34 であり，ほぼ同等の精度である．

処理速度としては，本評価では，SVMs に基づく英語品詞タグ付けは 1,347 単語を 1 秒間で処理し，SVMs に基づく Text Chunking では 1,535 単語を 1 秒間で処理という結果になった．これに対し，表 4 の Boosting にあたる，−Atomic でありかつ圧縮規則表現を適用しない場合であっても，英語品詞タグ付けでは 1 秒間に 1,842 単語を処理し，Text Chunking では 1 秒間に 1,681 単語を処理している．

このように，圧縮規則表現を用いない場合でも高速であった．この理由は，先行研究[12]にあるように，Boosting に基づく規則学習器は，小さいサイズの規則集合を生成する傾向にあることが関係していると予想される．

また，表 4 の圧縮規則表現にあたる，すべての素性の組合せを考慮した規則に圧縮規則表現を適用した場合は，英語品詞タグ付けでは 1 秒間に 2,195 単語を処理し，Text Chunking では 1 秒間に 2,477 単語を処理している．

さらに，表 4 の “圧縮規則表現 +Atomic” にあたる，すべての素性ではなく，部分的に素性の組合せを考慮する場合は，英語品詞タグ付けでは 1 秒間に 3,084 単語を，Text Chunking では 13,692 単語を処理している．これらの結果から，本提案手法により，SVMs に基づく英語品詞タグ付けおよび Text Chunking と同等の精度を保持したまま，高速な処理が行えていることが分かる．

7. 関 連 研 究

7.1 英語品詞タグ付けおよび Text Chunking における先行研究

本手法による精度を評価するために，**表 5** に英語品詞タグ付け，Text Chunking における先行研究の結果を載せる．本提案手法に基づく結果は，圧縮規則表現および一部を Atomic 素性として指定した結果のうち，開発データで最も高い精度を示した結果である．これらの比較からも本提案手法に基づく英語品詞タガーおよび Text Chunker は高い精度を残していることが分かる．

表 5　先行研究との比較．本稿の英語品詞タグ付けの結果は，$\{W = 7, \Delta = 3\}$，Atomic の結果．本稿の Text Chunking の結果は，$\{W = 5, \Delta = 2\}$，Atomic の結果

Table 5　Comparison with previous work. The our result of English POS tagging is the accuracy obtained with $\{W = 7, \Delta = 3\}$ and Atomic features. The our result of Text Chunking is the accuracy obtained with $\{W = 5, \Delta = 2\}$ and Atomic features.

英語品詞タグ付け	$F_{\beta=1}$
Guided learning [21]	97.33
Boosting [9]	97.32
半教師あり CRF [22]	97.40
本稿	97.33

Text Chunking	$F_{\beta=1}$
LaSo [3]	94.4
Boosting [9]	94.30
半教師あり CRF [22]	95.15
本稿	94.34

7.2 分類高速化手法における先行研究

機械学習に基づく分類の高速化手法としては，学習後のモデルを変換あるいは枝刈りする方法が提案されている．

SVMs に基づく分類器の高速化のためには，Polynomial Kernel を近似の線形分類器に変換する方法が提案されている[6),11)]．これらの手法では，学習された Support Vectort との類似度を Polynomial Kernel を用いて順次計算し素性の組合せを暗黙的に考慮していたのに対し，あらかじめ明示的に素性の組合せを展開しその重みを計算しておく．その結果，Support Vector との内積計算を行わずに，素性および素性の組合せチェックで判別を行えるようになる．

その他，素性の組合せを考慮する分類器の高速化手法としては，学習で選択された素性集合から，素性部分集合を選択し，選択された部分集合においてはその集合中の素性およびそれらの素性の組合せで得られるスコアをあらかじめ計算しておく方法が提案されている[25]．分類時は，入力の素性集合が，スコア計算済みの部分集合を含む場合は，その部分素性集合については，事前に計算した結果を利用することで，その部分集合中の素性の組合せを生成して計算することを行わずに済む．一致しなかった残りに関しては，組合せをそのつど生成して計算する．

また，Boosting アルゴリズム向けには，学習された弱仮説の枝刈り方法が提案されている[16]．この方法では，学習された N 種類の弱仮説から，M 種類の弱仮説を選択する際に，KL-divergence，Kappa 値などを基準にして選択を行う．KL-divergence の利用時は，M 種類の弱仮説から構成される弱仮説集合のうち，その集合内の弱仮説の組から計算される KL-divergence の和が最大となる集合を選択している．

先行研究では学習したモデルや規則の別表現への変換や学習結果の枝刈りを行っているのに対し，本手法では，系列ラベリングに特化した規則表現方法に変換することで高速化を実現している．本手法は，系列ラベリングという限定された問題に対する手法であるが，変換後も学習した規則と等価の意味であり，変換前後での精度は保証される．

また，本手法の学習では，素性の組合せを人手で与えるのではなく，素性の組合せに利用しない素性を指定するという方法で，素性の組合せ生成を調節する．この方法では，素性

の組合せに利用されない素性が少ない場合は，人手で素性の組合せを指定する場合と比較し，低コストでの調整が可能であるといえる．また，先行研究 9) と同様に，素性の組合せは Boosting アルゴリズムのトレーニングエラーの上限値の減少という観点から自動で生成されるので，あからじめ組合せを列挙する場合と比較し，有益な組合せが選択されると期待される．

さらに，素性の組合せを人手で与える場合は，学習の前に素性の組合せを展開するため，入力素性ベクトルのサイズが増大するという問題がある．これに対し，本手法では，組合せは学習時に展開されるため，初期の入力素性ベクトルのサイズは，素性の組合せを考慮しない場合と同様であるという利点もある．

8. 今後の課題

本系列ラベリングのための規則表現方法である圧縮規則表現に基づく手法は，本 Boosting アルゴリズム以外の機械学習アルゴリズムによる学習結果に対しても適用可能と考える．たとえば，PKE 法であれば，変換後の線形分類器は，本 Boosting アルゴリズムと類似する形式のモデルとなる．よって，本手法と同様に，前後の単語から得られる素性を用いて，英語品詞タグ付けや Text Chunking を実現している場合は，そのモデル中の素性に対し，圧縮規則表現を適用することが可能である．ほかにも，CRF [14]，Structured Perceptron [2] などの構造予測に基づく系列ラベリングにも同様に適用可能な手法である．今後は，他の学習アルゴリズムに基づく系列ラベリングにも，本圧縮規則表現を適用し，有効性を評価することが課題としてあげられる．また，さらなる高速化のために，文献 25) で提案されている方法を応用することも考えられる．

今回，圧縮規則表現は，系列ラベルリング問題において，素性が出現するかしないかというバイナリ素性で表現された事例集合からの学習結果が対象となっている．系列ラベリング問題においては，実数値の素性を扱うことも考えられるので，今後の課題としては，素性がバイナリで表現されない場合への拡張があげられる．

9. ま と め

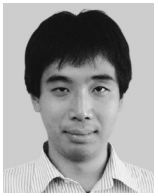
本稿では，品詞タグ付け，Text Chunking といった系列ラベリングにおける高速化手法を提案した．本稿では，通常の系列ラベリング問題では，判別対象の単語からの相対位置により，各単語が，複数回処理されるという問題に対処可能な圧縮規則表現を提案した．また，素性の組合せ生成を調整可能な Boosting アルゴリズムを提案した．この学習手法では，あらかじめ組合せに利用しない素性を指定することで，組合せ生成を調整できる．本手法を，英語品詞タグ付けと Text Chunking にて評価した結果，精度を保持したまま，最大で約 25 倍の速度改善が得られた．

参 考 文 献

1) Aoe, J.: An efficient digital search algorithm by using a double-array structure, *IEEE Trans. Software Engineering*, Vol.15, No.9 (1989).
2) Collins, M.: Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms, *Proc. EMNLP 2002*, pp.1–8 (2002).
3) Daumé III, H. and Marcu, D.: Learning as search optimization: Approximate large margin methods for structured prediction, *Proc. ICML 2005*, pp.169–176 (2005).
4) Escudero, G., Màrquez, L. and Rigau, G.: Boosting applied to word sense disambiguation, *Proc. 11th ECML*, pp.129–141 (2000).
5) Freund, Y. and Mason, L.: The alternating decision tree learning algorithm, *Proc. 16th ICML*, pp.124–133 (1999).
6) Isozaki, H. and Kazawa, H.: Efficient Support Vector classifiers for named entity recognition, *Proc. COLING 2002*, pp.390–396 (2002).
7) Iwakura, T. and Okamoto, S.: An assistant tool for concealing personal information in text, *HCI* (*9*), pp.38–46 (2007).
8) Iwakura, T. and Okamoto, S.: Fast training methods of boosting algorithms for text analysis, *Proc. RANLP 2007*, pp.59–65 (2007).
9) Iwakura, T. and Okamoto, S.: A fast boosting-based learner for feature-rich tagging and chunking, *Proc. CoNLL 2008*, pp.17–24 (2008).
10) Kudo, T. and Matsumoto, Y.: Chunking with Support Vector Machines, *Proc. NAACL 2001*, pp.192–199 (2001).
11) Kudo, T. and Matsumoto, Y.: Fast methods for kernel-based text analysis, *Proc. ACL-03*, pp.24–31 (2003).
12) Kudo, T. and Matsumoto, Y.: A boosting algorithm for classification of semi-structured text, *Proc. EMNLP 2004*, pp.301–308 (2004).
13) Kudo, T., Suzuki, J. and Isozaki, H.: Boosting-based parse reranking with subtree features, *Proc. ACL 2005*, pp.189–196 (2005).
14) Lafferty, J.D., McCallum, A. and Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Proc. ICML 2001*, pp.282–289 (2001).
15) Marcus, M.P., Santorini, B. and Marcinkiewicz, M.A.: *Building a large annotated corpus of english: The Penn Treebank*, pp.313–330 (1994).
16) Margineantu, D.D. and Dietterich, T.G.: Pruning adaptive boosting, *Proc. ICML*

1997, pp.211–218 (1997).
17) Minkov, E., Wang, R.C. and Cohen, W.W.: Extracting personal names from email: Applying named entity recognition to informal text, *HLT/EMNLP* (2005).
18) Sassano, M.: An experimental comparison of the voted perceptron and support vector machines in japanese analysis tasks, *Proc. IJCNLP'08*, pp.829–834 (2008).
19) Schapire, R.E. and Singer, Y.: Improved boosting algorithms using confidence-rated predictions, *Machine Learning*, Vol.37, No.3, pp.297–336 (1999).
20) Schapire, R.E. and Singer, Y.: Boostexter: A boosting-based system for text categorization, *Machine Learning*, Vol.39, No.2/3, pp.135–168 (2000).
21) Shen, L., Satta, G. and Joshi, A.: Guided learning for bidirectional sequence classification, *Proc. ACL 2007*, pp.760–767 (2007).
22) Suzuki, J. and Isozaki, H.: Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data, *Proc. ACL-08: HLT*, pp.665–673 (2008).
23) Toutanova, K., Klein, D., Manning, C.D. and Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network, *Proc. NAACL 2003*, pp.173–180 (2003).
24) Uchimoto, K., Ma, Q., Murata, M., Ozaku, H., Utiyama, M. and Isahara, H.: Named entity extraction based on a maximum entropy model and transformation rules, *Proc. ACL 2000*, pp.326–335 (2000).
25) Yoshinaga, N. and Kitsuregawa, M.: Polynomial to linear: Efficient classification with conjunctive features, *EMNLP*, pp.1542–1551 (2009).

(平成 22 年 11 月 5 日受付)
(平成 23 年　4　月 8 日採録)



岩倉　友哉（正会員）

2003 年 3 月九州工業大学情報工学研究科博士前期課程修了．同年株式会社富士通研究所入社．自然言語処理・機械学習の研究開発に従事．言語処理学会会員．