部分木に基づくマルコフ確率場と言語解析への適用

工藤 拓　　松本 裕治

奈良先端科学技術大学院大学 情報科学研究科

〒 630-0192 奈良県生駒市高山町 8916-5

{taku-ku,matsu}@is.aist-nara.ac.jp

近年, 統計的言語処理の分野において, 分類アルゴリズムを順次適用して結果を得る History-based Methods に代わり, 解析木, 解析タグ列 1 つを学習データとみなし, 大域的な最適解を導出できるモデル (マルコフ確率場 (MRF), 条件付き確率場 (CRF)) が注目を浴びている. しかし, 素性の設計は任意であり, 恣意的に設計されることが多かった. 本稿では, MRF に基づく言語解析のための, 一般的で, 効率よい素性選択手法を提案する. 具体的には, 全部分木を素性候補とし, 統計的な相関性に基づき, 有益な素性を選択する. 英語の品詞タグ付け, 英語の基本句同定を用いて有効性を検証したところ, 従来手法に比べ, 高精度を示した.
キーワード: マルコフ確率場, 対数線形モデル, 頻出部分木マイニング, テキストチャンキング, 品詞タグ付与

Subtree-based Markov Random Fields and Its application to Natural Language Analysis

Taku Kudo　　Yuji Matsumoto

Graduate School of Information Science, Nara Institute of Science and Technology

8916-5 Takayama, Ikoma Nara 630-0192 Japan

{taku-ku,matsu}@is.aist-nara.ac.jp

Global discriminative models, such as Markov Random Fields, (MRF) and Conditional Random Fields (CRF), have gained a great attention in the statistical natural language analysis. These models are shown to outperform the history-based approaches, where a parsed-tree can be generated by a sequential process of classifications. In the global optimization with MRF/CRF, the feature set can be selected heuristically. In this paper, we propose a subtree-based log-linear model, where statistically-useful subtrees are used as a feature set. Experiments with POS-Tagging and Base-Phrase Chunking show that our approaches outperform the previous heuristic selections.
Keywords: MRF, CRF, Log-Linear Model, Frequent Subtree Mining, Text Chunking, POS Tagging

1　はじめに

機械学習の各種の手法が, 品詞タグ付与, テキストチャンキング, 固有表現抽出, 構文解析といった自然言語処理に適用されるようになり, 数多くの成功をおさめている. これらの多くは, **History-based Methods** という一連の手法として一般化できる. History-based Methods では, 品詞列や構文木が, 「分類問題の順次適用」という形で表現され, 解析結果に対する確率は, それぞれの分類問題の確率の積で表現される. 個々の分類には, SVM といった既知の分類器が用いられる. この手法は, 比較的高精度であり, 機械学習の手法を直接適用可能であるといった利点を備えている. しかし, 「最適な」適用順序を一意に決定することは容易ではないという問題が内在する. 換言すると, 構文木や品詞タグ列を表現するための「構築順序」には多くの候補があり, 精度はその設計に強く依存している. 最適な設計はしばしば発見的に選択される.

この問題を解決するためのいくつかの手法が提案されている. Abney は, マルコフ確率場 (Markov Random Fields, MRF) を構文解析に最初に適用している [2]. Jung らは, MRF を, 品詞タグ付与問題に適用している [3]. Lafferty らは, MRF をタグ付与問題に特殊化した条件付き確率場 (Conditional Random Fields, CRF) を提案している [20, 27]. MRF は, 対数線形モデルの一種であり, 構文木の確率値の対数が, 構文木の局所的な, ポテンシャル (尤度) を返す素性関数の線形結合として表現される. MRF に基づく構文解析では, 素性は, 構文木の部分構造として特徴付けられる. これらの手法に共通する特徴としては, 1 つの構文木を 1 つの学習事例として捉えているところであろう. これらには, 構築順序という概念が無く, どのような方法, 順序で構文木を構築しても, 一意の確率値を得ることができる.

MRF は, 構文木全体をモデル化するのに優れているが, モデルの良さは, 部分構造を特徴付ける「素性関数」の設計に強く依存する. 広い文脈情報を考慮するために, 複雑な素性関数を用意すると, 素性関数の候補数が指数的に増えてしまう. 逆に, 素性関数を単純にすると, 構文解析に必要な重要な文脈情報を取

りこぼしてしまう可能性がある. 素性関数の設計は, モデル化するデータに強く依存しており, 事前に求まるものではない. これまでの研究では, いわゆる「素性テンプレート」を人手で発見的に用意し, それに基づき素性を選択していた [12, 27]. しかし, このような人手に基づく設計では, 重要な部分構造を取りこぼしてしまう可能性がある. かといって, テンプレートをいたずらに複雑にすることは, 解析効率という観点からも無駄が多く, 避けるべきである.

本稿では, MRF に基づく言語解析のための, 一般的で, 効率よい素性選択手法を提案する. 従来の MRF に基づく手法では, 人手によって素性を設計していたが, 本手法では, 膨大な素性候補から, 統計的な相関性に基づき, 有益な素性を選択する. 効率良い素性選択のために, 頻出部分木マイニングアルゴリズムである FREQT [1, 29] を用いる. さらに, FREQT を拡張した効率良い構文解析手法についても言及する. また, 英語の品詞タグ付与, 英語の基本句同定を用いて本手法の有効性を検証する.

1.1 統計的言語解析

構文解析や, 品詞タグ付与といった言語解析は, 木から木への写像関数 $f : \mathcal{X} \to \mathcal{Y}$, を与えられた学習データ $\{\langle X_k, Y_k\rangle\}_{k=1}^{L}$, (ただし $X_k \in \mathcal{X}$, $Y_k \in \mathcal{Y}$) から導出する一種の教師付き学習として定式化できる. 言語解析では, X は, 入力文であり, $\mathcal{Y}(X)$ は, 入力文 X に対する可能な解析候補である. 議論を一般化するために, X, Y は, 一般的な木として表現されるものとする. (文や品詞列, 構文木は, 木の特殊形である). このような定式化のもと, X, Y を, それぞれ **入力木** , **解析木** と呼ぶこととする. 図 1 に, 言語解析の例を示す.

統計的な言語解析では, 木のペア $\langle X, Y\rangle$ に対し, 条件付き確率 $P(Y|X)$ を与えることを考える. 入力木 X に対する, 最も確からしい解析木 $\hat{Y}$ は, $\hat{Y} = \mathrm{argmax}_{Y \in \mathcal{Y}(X)} P(Y|X)$ として与えられる. 言語解析における研究の議論は, 確率 $P(Y|X)$ の精度良い導出方法と, 候補集合 $\mathcal{Y}(X)$ から $\hat{Y}$ を発見する高効率な探索手法に注がれる.

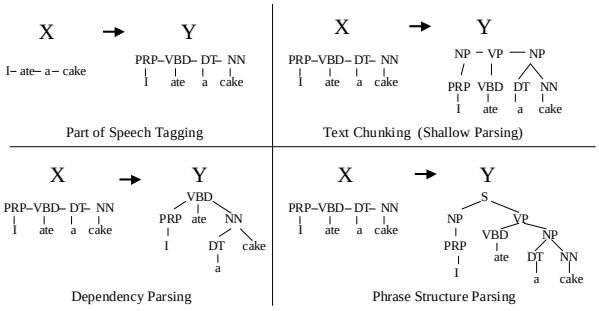


図 1: 言語解析の例

1.2 History-based Methods

History-based Methods[6](以下 HM) は, 言語解析の多くのタスクに適用されている [18, 30, 11, 25]. HM は, 言語解析を分類問題の順次適用として定式化し, 解析木 Y を, ある正準的な構築順序 $D = \langle d_1 \dots d_n\rangle$ に分解する. つまり, 解析とは, 最も確からしい構築順序 $\hat{D}$ を全候補 $Y \in \mathcal{Y}(X)$ から発見する一種の探索問題として定式化される. これらの表記を用いると, 条件付き確率 $P(Y|X)$ は, 以下のように与えられる.

$$P(Y|X) = \prod_{i=1\dots n} P(d_i|\ \Phi(\psi(X, d_1 \dots d_{i-1}))),$$

ただし, $(d_1 \dots d_{i-1})$ は, i 番目の決定に用いられる **履歴 (history)** であり, $\psi(\cdot,\cdot)$ は, 履歴と入力木から, 部分的に解析された解析木を構成する関数である. さらに, $\Phi(\cdot)$ は, 部分的に解析された解析木から, $\Re^H$ 次元のベクトルに写像する関数である. HM では, 現在の位置での分類は, 以前に与えられた分類結果に依存する. HM の学習は, 元の解析問題を, 既知の多値分類器が適用可能である部分問題の集合に分解することから始まる $(\langle X, Y\rangle \mapsto (\langle \mathbf{x}_1, y_1\rangle \dots \langle \mathbf{x}_n, y_n\rangle)$, ただし $y_i = d_i$, $\mathbf{x}_i = \Phi(\psi(X, d_1 \dots d_{i-1})))$. 次に, 個々の部分問題が iid (独立同一分布) から生成されたものと仮定し, 任意の機械学習アルゴリズムを実行する.

HM は, 既知の学習アルゴリズムをブラックボックス的に用いることができ, 実装が容易であり, 比較的精度が高いといった多くの利点を備える一方で, いくつかの欠点が存在する. まず, 木を正準的な構築順序に変換する方法には多くの候補 (トップダウン, ボトムアップ, 左角, 右角, 前向き, 後ろ向き など) が存在し, どの順序が最適が一般に決定できない. さらに, 個々の部分問題が, iid から生成されるといるという仮定にはいささか無理がある. 例えば, 品詞タグ付与やテキストチャンキングといったタグ付与問題を考えてみよう. これらのタスクでは, 文を前から後ろに解析する (前向き解析) か, 後ろから前に解析する (後ろ向き解析) かで, 解析精度が大きく変わることが知られている [18]. 最適な解析方法は, タスクのみならず個々の入力事例に依存する. 従来の手法では, 前向き解析と後ろ向き解析の発見的な組みあわせで, この問題を解決している [18]. HM におけるこのような問題は, **ラベルバイアス問題** として認知されている. 詳細は, 文献 [20] に譲る.

2 部分木に基づく対数線形モデル

この章では, Subtree-based Log-Linear Model (以下 SLL) を定式化する. 詳細に入る前に, 木に対するいくつかの定義, 表記について言及する.

2.1 準備

定義 1 ラベル付き順序木
ラベル付き順序木は, すべてのノードに一意のラベルが付与されており, 兄弟関係に順序が与えられる木である.

本稿では, ラベル付き順序木を単純に木と呼ぶ.

定義 2 部分木
T と U ラベル付き順序木とする. ある木 T が, U にマッチする, もしくは, T が U の部分木である *($T \subseteq U$)* とは, T と U ノード間に, *1* 対 *1* の写像関数 ψ が定義できることである. ただし, ψ は, 以下の *3* つの条件を満たす. *(1)* ψ は, 親子関係を保持する. *(2)* ψ は, 兄弟の順序関係を保持する. *(3)* ψ はラベルを保持する.

本稿では, 木 Y 中のすべての部分木の集合を $\mathcal{S}(Y)$ ($\mathcal{S}(Y) = \{Y' | Y' \subseteq Y\}$), さらに 木 Y のノードの数を $|Y|$ と表記する.

2.2 SLL の定式化

マルコフ確率場 (Markov Random Fields (MRF))[8] は, 画像のピクセル, ラベル列といった, 周辺の文脈に依存する値をモデル化するのに適した手法である. m 個のランダムな値 (実数, 離散値, ラベルなど) $f = \{f_1, \ldots, f_m\}$ を考える. f が, MRF であるとは, f が以下の条件を満たす時で, かつその時に限る. (1) $P(f) > 0$ $(\forall f \in F)$ (positivity), (2) $P(f_i | f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_m) = P(f_i | \mathcal{N}_i)$ (markovinaity) (ただし, $\mathcal{N}_i$ は, i の近隣 ($i \notin \mathcal{N}_i$) である). 近隣どうしを結ぶと一種の無向グラフが完成する. このグラフのクリーク集合 $\mathcal{C}$ を考えると, 確率 $P(f)$ は, 個々のクリーク $c \in \mathcal{C}$ のポテンシャル関数 $V_c(f)$ の対数線形モデルになることが知られている [16] (i.e., $P(f) = \exp(\sum_{c \in \mathcal{C}} V_c(f))/Z$).

我々の言語解析問題では, F は, $\mathcal{Y}(X)$ が張る森に, f は, 1 つの木 $Y \in \mathcal{Y}(X)$ に, 各値 f_j は, 木のノードに, 近隣は, 周辺のノードや構造にそれぞれ対応する. すべての近隣が全ノードであるような完全グラフを考え, そのクリークのうち, もとの木 Y の部分木 y のみに非 0 のポテンシャル λ_y を与えると, 以下のような対数線形モデルで条件付き確率 $P(Y|X)$ を定式化できる.

定義 3 *Subtree-based Log-Linear Model (SLL)*
木のペア $\langle X, Y \rangle$, に対し, *SLL* は以下のように条件付き確率 $P(Y|X)$ を与える.

$$P(Y|X) = \frac{\exp(\sum_{y \in \mathcal{S}(Y)} \lambda_y)}{Z(X)}, \quad (1)$$

$$\text{ただし } Z(X) = \sum_{Y \in \mathcal{Y}(X)} \exp(\sum_{y \in \mathcal{S}(Y)} \lambda_y), \ \lambda_y \in \Re$$

木の集合 $\mathcal{F} = \bigcup_k \mathcal{S}(Y_k)$ は, SLL の素性集合と呼ばれる. $\mathbf{\Lambda} = (\lambda_{y_1}, \ldots, \lambda_{y_{|\mathcal{F}|}})$ は, SLL のパラメータであり, 素性集合の個々のアイテム (木) に対応する. これらは, 単純な最尤推定法で求めることができる. つまり, 学習データ $\mathcal{T} = \{\langle Y_k, X_k \rangle\}_{k=1}^{L}$ に対する対数尤度

$$\mathcal{L}_{\mathbf{\Lambda}} = \sum_k \log(P(Y_k|X_k))$$
$$= \sum_k \Big[\sum_{y \in \mathcal{S}(Y_k)} \lambda_y - \log(Z(X_k)) \Big]$$

の最大化として与えられる. 実際には, 過学習を防ぐために, パラメータ Λ に対し正規分布の正則化を与える [9].

$$\mathcal{L}_{\mathbf{\Lambda}} = \sum_k \Big[\sum_{y \in \mathcal{S}(Y_k)} \lambda_y - \log(Z(X_k)) \Big] - \frac{||\mathbf{\Lambda}||^2}{\sigma^2}$$
$$\hat{\mathbf{\Lambda}} = \operatorname*{argmax}_{\Lambda} \mathcal{L}_{\Lambda}$$

これは, 一種の**事後確率最大化** (maximum a posteriori estimation, MAP) である. パラメータ σ は, SLL のハイパーパラメータであり, モデルの複雑さと, 学習データに対する対数尤度のバランスを決定する. σ は, デベロップメントデータを用いたり, 交差検定といった既知のモデル選択手法を用いて選択される. 最適解 $\hat{\mathbf{\Lambda}}$ は, IIS や GIS といった, Iterative Scaling Algorithm を用いて求められる [9, 24].

2.3 SLL の近似 (ASLL)

SLL は, すべての部分木を素性として用いることを除けば, 単純な 対数線形モデルである. ただし, SLL を実際に運用するには, 以下のような問題がある.

(1) **解析木の列挙**
$\mathcal{L}_{\mathbf{\Lambda}}$ の微分 $\mathcal{L}'_{\mathbf{\Lambda}}$ の計算は, 最適化問題に不可欠であるが, この導出には, 正規化項 $Z(X)$ を陽に計算する必要がある. 換言すると, 入力木 X に対する解析可能な**すべての**解析木 Y を列挙しなければならない. 解析木の候補数 $|\mathcal{Y}(X)|$ は, 入力木のサイズ $|X|$ に対し, 指数的に増えるため, すべての解析木を陽に列挙することは, 一般に困難である[1] [2]

(2) **素性集合の構築**
SLL における素性は, 解析木 Y 中のすべての部分木であり, 部分木のサイズは, 元の木のサイズに対し指数的に増えるため, 素性集合のサイズ $|\mathcal{F}|$ は, しばしば巨大になる. 素性 (部分木) それぞれに 1 つのパラメータが対応するため, 巨大なパラメータ空間となり, パラメータ Λ の導出が困難になる.

これらの問題を解決するために, 以下のような修正を SLL に与える.

(1) すべての解析候補を使う代わりに, history-based parser が出力する n ベスト解 (例えば $n = 20$) を, 全候補とみなす. これは, 文献 [12, 13, 14] と同じ設定である. 本稿では, このような候補集合を $\mathcal{Y}_h(X)$ と表記する. 全候補 $\mathcal{Y}(X)$ を $\mathcal{Y}_h(X)$ で代用した場合, 学習データ $\langle X_k, Y_k \rangle$ に対し,

[1] 文法や, 事前知識を用いて制限する事は可能であるが, これらの情報が常に利用できるとは限らない.

[2] CRF[20] や, Feature Forests[21, 10] は, 動的計画法を用いて, この問題を解決している. 詳細は, 4.3 章で論じる

$\mathcal{Y}_h(X_k) \cap \{Y_k\} = \phi$ となる場合がある．このような場合は, 解析木 Y_k に対して SLL の学習が行えない．この問題は, **擬似結果** $\mathcal{Y}'_h(X_k) = \mathcal{Y}_h(X_k) \cup \{Y_k\}$ を用いたり, $\mathcal{Y}_h(X_k)$ から最も正解に近い (精度が良い) **擬似解答** Y'_k を正解 Y_k の代用とするなどして解決する．このような擬似データを用いることで, $\mathcal{Y}_h(X_k) \cap \{Y_k\} \neq \phi$ という条件を想定できる．

(2) 全素性集合 $\mathcal{F}$ を使う代わりに, その部分集合 $\mathcal{F}' \subset \mathcal{F}$ を素性として用いる (ただし, $|\mathcal{F}'| \ll |\mathcal{F}|$)．一般には, 任意の**素性選択関数** $\mathcal{Q}(\cdot,\cdot) \in \{true, false\}$ を与え, この関数が真を返す部分木のみを集合として用いる (i.e., $\mathcal{F}' = \{y | y \in \mathcal{F} \ \& \ \mathcal{Q}(y,\Sigma) = true\}$, ただし Σ は, **素性選択パラメータ**と呼ばれる)．

便宜上, 素性選択関数 $\mathcal{Q}$ を以下のような 3 つの部分関数 $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3$ に分解する (i.e., $\mathcal{Q}(y,\Sigma) = \mathcal{Q}_1(y,\Sigma) \wedge \mathcal{Q}_2(y,\Sigma) \wedge \mathcal{Q}_3(y,\Sigma)$)．

(1) **頻度:** Q_1
一般に, 頻出する集合が重要である．そこで, τ 個以上の学習事例に出現した 部分木のみを素性として用いる (i.e., $\mathcal{Q}_1(y,\Sigma) = \text{true if } |\{Y_k | y \subseteq Y_k\}| \geq \tau$)．このような頻度に基づくフィルタリングは, 統計的自然言語処理において頻繁に適用される．

(2) **部分木のサイズ:** Q_2
大きな部分木 (素性) は, 過学習の原因となりやすい．また, すべての部分木を用いることは, 汎化能力という意味で悪い影響を与えるとことが知られている [15]．さらに, 与えられたタスクに対し, 最適な部分木のサイズが存在すると考えるのが自然であろう．これらの理由から, 部分木のサイズに制限を設ける．具体的には, サイズが L から M の間にある部分木のみを素性として用いる (i.e., $\mathcal{Q}_2(y,\Sigma) = \text{true if } L \leq |y| \leq M$)．

(3) **統計的相関性:** Q_3
学習データ $\{\langle X_k, Y_k \rangle\}_{k=1}^L$, に対し, $\mathcal{P} = \{Y_k\}_{k=1}^L$ を正例, $\mathcal{N} = \bigcup_k \{\mathcal{Y}_h(X_k) - \{Y_k\}\}$ を負例とする．素性の「良さ」を, 正例, 負例の分布との一種の相関性で評価することは自然であろう．相関性を具体的に計測するために, 本稿では, カイ二乗統計量を用いる．O_{ij} (ただし $(i,j) \in \{P,N\} \times \{y, \bar{y}\}$) を部分木 j を含む (含まない), 正例 ($i = P$) もしくは, 負例 ($i = N$) の学習事例とする．例えば, $O_{N\bar{y}}$ は, 部分木 y を含まない負例の数となる．部分木 (素性) y に対する カイ二乗値は, 以下で与えられる．

$$chi(y) = (ad - bc)^2/(a+b)(a+c)(b+d)(c+d),$$
$$\text{ただし } a = O_{Py}, b = O_{Ny}, c = O_{P\bar{y}}\, d = O_{N\bar{y}}$$

3 つ目の制約として, カイ二乗値が, ある閾値 χ 以上となるすべての部分木を素性として用いる (i.e., $\mathcal{Q}_3(y,\Sigma) = \text{true if } chi(y) \geq \chi$)．

これらの近似を用い, Approximated SLL は以下のように定式化できる．

定義 4 *Approximated SLL (ASLL)*
学習データ $\{\langle X_k, Y_k \rangle\}_{k=1}^L$, 素性選択パラメータ $\Sigma = \langle \tau, \chi, L, M \rangle$, さらに任意の *history-based parser* $\mathcal{Y}_h(\cdot)$, が与えられた時に, *ASLL* は, 条件付き確率 $P(Y|X)$ を以下のように与える．

$$P(Y|X) = \frac{\exp(\sum_{y \in \mathcal{S}(Y) \cap \mathcal{F}'} \lambda_y)}{Z(X)}, \quad (2)$$

ただし

$$Z(X) = \sum_{Y \in \mathcal{Y}_h(X)} \exp\Big(\sum_{y \in \mathcal{S}(Y) \cap \mathcal{F}'} \lambda_y\Big),$$
$$\lambda_y \in \Re,$$
$$\mathcal{F}' = \{y |\ y \in \bigcup_k \mathcal{S}(Y_k) \ \& \ \mathcal{Q}(y,\Sigma) = true\}$$

素性選択ラメータ $\Sigma = \langle \tau, \chi, L, M \rangle$ は, 交差検定といった既知のモデル選択手法を用いて求めることができる．さらに, 最尤の解析木は以下で与えられる．

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}_h(X)} P(Y|X) = \operatorname*{argmax}_{Y \in \mathcal{Y}_h(X)} \Big(\sum_{y \in \mathcal{S}(Y) \cap \mathcal{F}'} \lambda_y\Big) \quad (3)$$

実際には, History-based Method (HM) と, ASLL は異なる観点に基づき解析が行われる．例えば, 品詞タグ付与問題では, HM の多くは, 未知語処理のために, 部分文字列の情報を用いているが, これらの情報は, ASLL では用いられない．それぞれの情報をうまく組みあわせるために, 実際の解析では, 以下のような線形補完を考える．

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}_h(X)} \Big[\alpha \sum_{y \in \mathcal{S}(Y) \cap \mathcal{F}'} \lambda_y + (1-\alpha) L_h(Y|X)\Big] \quad (4)$$
$$\text{ただし } 0 \leq \alpha \leq 1$$

$L_h(Y|X)$ は, 解析木 Y の HM $\mathcal{Y}_h(\cdot)$ に基づく対数尤度である．このような線形補完は, 文献 [12, 13, 14] でも用いられている．α ($0 \leq \alpha \leq 1$) は, それぞれの影響を決定するパラメータであり, Σ と同様, 交差検定を用いて選択される．

3 実装

3.1 部分木マイニングアルゴリズムの適用

本章では, 素性集合 $\mathcal{F}'$ を効率良く構築するためのアルゴリズムについて言及する．最も単純な方法は, すべての部分木を事前に完全に列挙して, その後に, 低頻度の素性を取り除くという手法であるが, 素性集合のサイズは一般に巨大であり, この手法は実際には適用困難である．さらに, 全部分木を列挙する問題は, NP 困難であることが知られている．この問題を解決するために, 本稿では, 頻出部分木マイニングアルゴリズムを適用する．

問題 1 頻出部分木マイニング
木の集合 $\mathcal{T} = \{Y_1, \ldots, Y_L\}$ が与えらえた時に, すべ

ての部分木 $y \in \bigcup_k \mathcal{S}(Y_k)$ の候補から, ξ 個以上の木に含まれる部分木のみを列挙せよ *(i.e.,* $\sum_k I(y \subseteq Y_k) \geq \xi$*)*.

ξ はユーザが与えるパラメータで, 一般に, 最小サポート値と呼ばれる. Abe と Zaki は, この問題のために, 効率良い方法 FREQT, TreeMinerV をそれぞれ独立に提案している[3][1, 29]. FREQT, TreeMinerV の鍵となるアイデアは, 木の集合から効率良く部分木を列挙する手法, **最右拡張**に集約される.

定義 5 最右拡張 *[1]*
3 つの木 $T(\neq \phi)$, $T'(\neq \phi)$ $U(\neq \phi)$, が与えらた時, T' が, T の U に基づく最右拡張であるとは, 以下の *3* つの条件が満たされる時であり, かつその時に限る. *(1)* T と T' は U の部分木である *(i.e.,* $T \subseteq U$ かつ $T' \subseteq U$*)*. *(2)* T' は, T に *1* つの ノードを追加することで構築される *(i.e.,* $T \subset T'$ かつ $|T|+1 = |T'|$*)*. *(3)* ノードは, T の最右パス上のノード *(*ルートから常に右端のノードを葉まで選ぶ*)* に追加される. *(4)* 追加されるノードは, 最右の兄弟である.

便宜上, ここでは次のような表記を用いる: $\{\langle T'_1, i'_1\rangle, \ldots, \langle T'_n, i'_n\rangle\} = \mathrm{RME}(T, i, U)$. ただし $T, T'_j\,(j = 1, \ldots, n)$ 及び U は, 定義 5 で導入した木である. つまり, $\{T'_j | j = 1, \ldots, n\}$ は T の U に基づく最右拡張となる木の集合である. また, i (および i'_j) は, T (および T'_j) の U に基づく最右葉の**位置** である (U 中のノードは, 前順序 (pre-order) で順序が付与されているものとする). 図 2 に 最右拡張の具体例を示す. 最右拡張を, 再帰的に適用することで, 一種の

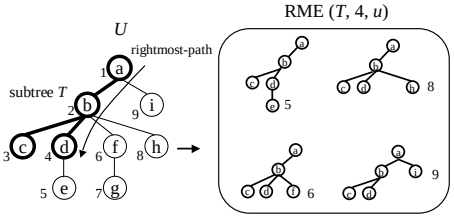


図 2: 最右拡張

「探索木」を構築するとができる. FREQT, TreeMinierV は, この探索木を深さ優先探索し, 頻出部分木を発見していく. 位置 i は, 部分木の出現位置を示す. また, T の任意の上位集合 (上位木) $T'(T' \supset T)$ の頻度は, T の頻度以下になることを利用して, 効率良く探索空間を枝刈りすることができる (T が頻出でなければ, どんな上位木 T' も頻出ではない). 素性集合 $\mathcal{F}'$ は, 基本的に FREQT(TreeMinerV) を用いて構築できる. ただし, カイ二乗値については, 上記の頻度のような性質[4]を持たないので, 単純には枝刈りできない. そのため, 頻度とサイズを条件にいったん素性を列挙した後に, カイ二乗に基づき素性を再選択する[5].

3.2 解析の高速化

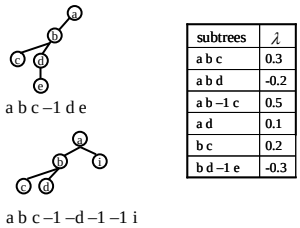
式 (3) (もしくは (4)) を用いて, 実際の解析が行われる. この計算量は, 以下の TreeMatcher 問題のそれと等価であるため, ここではこの問題に着目する.

問題 2 *TreeMatcher*
木の集合 $\mathcal{F}'$ と, 木 Y が与えられた時, Y 中のすべての部分木のうち, $\mathcal{F}'$ に含まれるものを列挙せよ. *(i.e.,* 集合 $\{y | y \in \mathcal{S}(Y) \cap \mathcal{F}'\}$ を構築せよ*)*

これまでの MRF に基づく言語解析では, 事前に用意された **素性テンプレート** を用いていた. つまり, テンプレートに基づき素性を完全に生成し, $\mathcal{F}'$ 中の有無をテストする単純な **「生成-テスト」** で実現できる. しかし, 本モデルでは, そのような明示的なテンプレートがない. 最も単純な解析手法は, Y 中の全素性 (部分木) を完全に列挙することであるが, 素性集合 $\mathcal{F}'$ に存在しない無駄な部分木を大量に生成する可能性があり効率が悪い. 実際に, この問題も, 最右拡張を用いて高速に実現できる. 詳細の前に, 木の文字列表現について言及する.

定義 6 木の文字列表現 *[29]*
木 T を, 以下の方法で一意の文字列表現 $str(T)$ に変換する. *(1)* 初期化 $str(T) = \phi$ *(2)* 前順序探索 *(pre-order)* で, 木のノードを列挙し, 各ノードのラベルを $str(T)$ の末尾に追加する. *(3)* 探索中に, 子から親に後戻りする場合, ラベル集合に含まれない特殊なラベル *-1* を $str(T)$ の末尾に追加する.

このような文字列表現の例を図 3 に示す. さらに, $\mathcal{F}'$ 中のすべての木を, 文字列表現に変換し, それらすべてを単一の TRIE で表現する (図 4). 文字列表現中のラベルの順番と, 最右拡張が広げていくラベルの順番は同一なので, 構築された TRIE は, 最右拡張によって定義される「探索木」と同一になる. つまり, 木 Y が与えられると, Y の各ノードに対し, 最右拡張を行いながら, TRIE を探索していくことで, TreeMatcher が実現できる. TreeMatcher の計算量は, TRIE の深さに依存するが, 深さは定数で抑えられるため, $O(|Y|)$ となる.



subtrees	λ
a b c	0.3
a b d	-0.2
a b –1 c	0.5
a d	0.1
b c	0.2
b d –1 e	-0.3

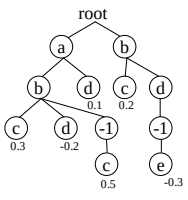


図 3: 文字列表現　図 4: 素性集合の TRIE

[3]FREQT と TreeMinerV は, 定式化や, 解こうとしている問題が微妙に異なるが, 本質的なアイデアは同一である
[4]反単調性 (anti-monotonicity) と呼ばれる
[5]これまでに, カイ二乗といった統計的検定に基づくマイニングアルゴリズムが提案されている [22]. しかし, 本タスクでは, 後処理で十分実行可能であったため, 採用を見送った.

4 実験と考察

4.1 実験環境, 設定

実験は, 以下の 2 種類のタスクで行った.

- 英語品詞タグ付与 (**POS-Tagging**)
  一般的な英語品詞タグ付与タスクである. Penn Tree-bank/WSJ の 15-18 を学習データ, 21 をテストデータとし, 各種パラメータは, 20 を用いて設定した. History-based Tagger は, 最大エントロピー法 (ME) に基づく解析器 [25] を用いた[6]. ASLL の適用には, Histroy-based Tagger の結果を木構造に変換する必要があるが, 図 5(上) のように単純な左下りの木とした. 部分木 (素性) は, 任意の品詞 n-gram や語彙が付与された品詞 n-gram である. 評価は, 精度で行った.
- 英語基本句同定 (**Base-Phrase Chunking**)
  CoNLL-2000 の Shared Task[26] で用いられた, 英語の基本句 (入れ子が無い最小の句構造) の同定タスクである[7]. Penn Tree-bank/WSJ の 15-18 を学習データ, 21 をテストデータとし, 各種パラメータは, 交差検定にて設定した. History-based Chunker には, 正則化付き最大エントロピー法 (ME)[9] に基づく解析器を独自に設計した. 基本的に文献 [25] の拡張であるが, 詳細は割愛する. History-based Chunker の結果は, 図 5(下) のような木構造に変換した. 基本的に, 句レベルでは, 直後の句に係けておき, 単語はその品詞に, 品詞は, 句内の主辞に係ける. 主辞は, 文献 [11] のルールに従って選択した. また, 図中には示していないが, 左右どちらから係るのか, 句の先頭か末尾か, といった情報を示すノードを単語の兄弟として追加した. 評価は, 一般的な F 値 (精度と再現率の調和平均) で行った.

本手法は, 木構造であれば, どのような表現でも学習対象になりえ, その点からも汎用的な手法であると言える. もちろん, POS-Tagging, Chunking のための, より洗練された「木表現」について議論の余地はあるが, これらの表現に固定して実験を行った.

4.2 実験結果

表 1 に, POS-Tagging の結果を示す. ME に基づく History-based Tagger (ME-HT) に加え, SVM に基づく修正学習法 (RLT)[23] の結果を示す. RLT に用いられるパラメータも, WSJ20 を用いて選択したため, 公平な比較となっている.

表 2 に, Base-Phrase Chunking の結果を示す. ME に基づく History-based Chunker (ME-HC) の結果に加え, SVM に基づく History-based Chunker(SVM-HC)[19], 8 つの SVM-HC の多数決 (8-SVM-HC)[18], Reguralized-Winnow に基づく History-based Chunker (RW-HC)[30] の結果を示す. RW-ESG-HC[30] は, ESG と呼ばれる高度な言語情報 (syntactic role 等) を素性として用いているため, 直接の比較は行わない. 表 3 は, 本手法の詳細な結果である. また, CRF は, 8-SVM-HC と同等の精度であることが報告されている [27].

結果, 両タスクにおいて, 提案手法が, 既存の手法に比べ高い精度を示すことが分かった.

表 1: 実験結果 (POS-Tagging)

モデル	精度 (%)
ME-HT(n ベスト出力)[25]	95.98
RLT[23]	95.87
ASLL(提案手法)	96.10

表 2: 実験結果 (Base-Phrase Chunking)

モデル	F 値
ME-HC(n ベスト出力)[25]	93.40
SVM-HC[19]	93.46
8-SVM-HC[18]	93.91
RW-HC[30]	93.57
ASLL(提案手法)	94.13
RW-ESG-HC [30]	94.17

ASLL には, 選択すべきパラメータがいくつかある. その中で L, M は, 各タスクにおいて必要な部分木のサイズであり, 素性選択の上で最も重要なパラメータだと考えられる. $L = 2$ に固定し, M を変化させた場合, 精度の最も良い M は, $M = 4$ (POS-Tagging), $M = 5$ (Base-Phrase Chunking) であった. この結果は, 最適なサイズが存在するという我々の直観, 及び, 文献 [15] の分析と合致する.

4.3 関連研究との比較

Collins は, 対数線形モデルで, History-based Parser の結果をリランキングする手法を提案している [12]. CRF[20] は, 同様に対数線形モデルであるが, History-based Parser の結果を用いず, 直接式 (1) の結果を解析木の全候補から学習している. さらに近年, 対数線形モデルではなく, SVM の枠組みで, ラベル付与問題を定式化する手法も提案されている (HMM-SVM)[4, 5]. これらの手法は, 人手で設計された「素性テンプレート」を用いてる点が本手法と異なる. 本手法は, 自動的に素性を選択し, その手法が個々のタスクに依存しないという点で, 優れていると考える.

Data Oriented Parsing (DOP), は, SLL のように木の中のすべての部分木を用いる構文解析アルゴリズムである [7]. DOP は, 木のサイズや語彙化に関係なく, すべての部分木を, PCFG ベースの文法に変換する. 個々の PCFG の派生ルールの確率は, 親ラベルを共有する全部分木のルールの確率の総和で計算される. DOP は NP 困難であることが知られ, 実現には, モンテカルロ等のサンプリング法が用いられる. しかし, このようなテクニックを使っても, 十分

[6]通常の ME ではなく, 正則化項付きの ME[9] を用いた.
[7]http://lcg-www.uia.ac.be/conll2000/chunking/

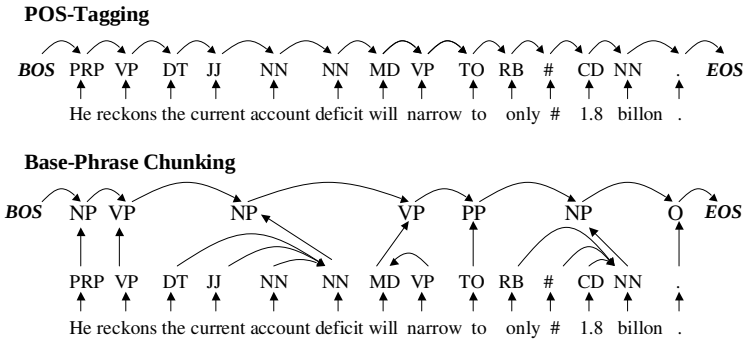


図 5: 木構造への変換, POS-Tagging(上) Base-Phrase Chunking(下)

表 3: ASLL による Chunking の結果 (詳細)

	precision	recall	$F_{\beta=1}$
ADJP	83.59%	74.43%	78.74
ADVP	81.65%	82.22%	81.93
CONJP	38.46%	55.56%	45.45
INTJ	100.00%	50.00%	66.67
LST	0.00%	0.00%	0.00
NP	94.64%	94.55%	94.60
PP	96.82%	98.19%	97.50
PRT	79.80%	74.53%	77.07
SBAR	91.33%	84.67%	87.88
VP	94.13%	94.38%	94.25
all	94.17%	94.08%	94.13

な統計量を得るためには, 文のサイズに対し指数的に増えるルールが必要であり, 実際の応用に適用できるかどうか疑問が残る.

Tree Kernel [13, 14, 17] に基づく言語解析でも, SLL, DOP と同様にすべての部分木の情報を用いている. Kernel 法では, 結果が事例間の内積 (類似度) のみに依存するため, DOP のように, 陽に部分木を列挙する必要はなく, 効率良く内積が計算できさえすれば良い. 実際に Tree Kernel は, 動的計画法を用いて, 陽に列挙することなく内積を計算している. しかし, Tree Kernel を用いた言語解析の計算量は, $O(L \cdot N \cdot |Y|)$ であり, 他の手法に比ばれば依然大きい (ただし, N はサポートベクターとなるの木の平均サイズ, L はサポートベクターの数, Y は解析木の候補). 本手法の計算量は, $O(|Y|)$ であり, Tree Kernel に基づく手法に比べれば, 高速に解析が行える.

CRF[20] や, その木構造への拡張である Feature Forests [21, 10] は, 2.3 節の (1) で述べた解析木の列挙の問題を, 巧妙に解決している. 具体的には, Forward-Backward, Inside-Outside といった, 動的計画法を拡張し, 正規化項を, 陽に全解析木を列挙することなく, 陰に計算している. しかし, これらの手法は, 部分木のサイズが小さい場合は有効だが, 部分木のサイズが大きくなるにつれ, 空間計算量が指数的に増加してしまう. 実際に, 彼等の実験では, 人手で用意された単純な素性テンプレートを用いたり, 既存の文法上に実装していたりと, 計算量は問題にならない規模である. 我々の目的は, 自動的な素性選択と, 解析に必要な十分大きい素性 (部分木) を用いて解析することであるため, この目的が実現できるようなアルゴリズムを適用するほうが望ましいと考える.

5 おわりに

本稿では, MRF に基づく言語解析のための, 一般的で, 効率よい素性選択手法を提案した. さらに, 英語の品詞タグ付与, 英語の基本句同定を用いて本手法の有効性を検証した. 今後の計画として, 以下が挙げられる.

- 他のタスク
  本手法は, 言語解析全般に適用できる汎用的な手法である. 今後は, 係り受け解析, 構文解析といったタスクで有効性を検証したいと考える.
- グラフ構造
  これまでは, 木構造のみに着目していたが, より複雑な情報を表現するには, グラフを用いるほうがよい. 効率良いグラフマイニングアルゴリズムが提案されているので [28], 今後は, それらを適用したいと考える.
- 候補木の列挙
  本稿では, History-based Parser の n ベスト解を用いて, 全候補を近似した. しかし, この近似は妥当とは言いがたい. 今後は, マルコフ連鎖モンテカルロといった精度の良いサンプリング手法を用いて, 解析候補を選択したいと考える[8].
- 学習アルゴリズム
  本稿では, 単純な 対数線形モデルを学習に用いた. しかし, 近年, HMM-SVM のように, SVM の枠組みで定式化する手法が提案されている [4, 5]. SVM を用いると, 限られたサポート事例のみで, パラメータを表現できるため, 陽に正規化項を列挙しなければならない問題を解決できるばかりでなく, 汎化能力を高めることができる[9].

[8] n ベスト解を用いることは, Hitory-based Parser を近似分布とする重点サンプリングを行ってるものと解釈できるが, その近似がうまくいくのは, n ベスト解の確率の和がほぼ 1 になる場合のみで, 実際にはこのような仮定は成立しない.

[9] HMM-SVM は, 式 (1) を, SVM の枠組みで学習する. 通常の

参考文献

[1] Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. Optimized substructure discovery for semi-structured data. In *Proc. 6th European Conference on PKDD*, 2002.

[2] Steven P. Abney. Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618, 1997.

[3] Sung-Young Jung adn Young C. Park, Key-Sun Choi, and Youngwhan Kim. Markov random field based english part-of-speech tagging system. In *The 16th International Conference on Computational Linguistics, Vol.1*, pages 236–242, 1996.

[4] Yasemin Altun, Mark Johnson, and Thomas Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proc. of EMNLP*, pages 145–152, 2003.

[5] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines. In *Proc. of ICML*, pages 3–10, 2003.

[6] Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceedings, DARPA Speech and Natural Language Workshop*, 1992.

[7] Rens Bod. *Beyond Grammar: An Experience Based Theory of Language*. CSLI Publications/Cambridge University Press, 1998.

[8] Rama Chellappa and Anil Jain (eds.). *Markov Random Fields: Theory and Applications*. Academic Press, 1993.

[9] Stanley F. Chen and Ronald. Rosenfeld. A gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, 1999.

[10] Stephen Clark and James Curran. Log-linear models for wide-coverage CCG parsing. In *Proc. of EMNLP*, pages 97–104, 2003.

[11] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[12] Michael Collins. Discriminative reranking for natural language parsing. In *Proc. of ICML*, pages 175–182, 2000.

[13] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Proc. of Neural Information Processing Systems (NIPS)*, 2001.

[14] Michael Collins and Nigel Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc of ACL*, 2002.

[15] Chad Cumby and Dan Roth. On kernel methods for relational learning. In *Proc. of ICML*, pages 107–114, 2003.

[16] Hammersley and Clifford. Markov fields on finite graphs and lattices. 1971.

[17] Hisashi Kashima and Teruo Koyanagi. Svm kernels for semi-structured data. In *Proc. of the ICML*, pages 291–298, 2002.

[18] Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proc. of NAACL*, pages 192–199, 2001.

[19] Taku Kudoj and Yuji Matsumoto. Use of Support Vector Learning for Chunk Identification. In *Proc. of of CoNLL-LLL-2000*, pages 142–144, 2000.

[20] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289, 2001.

[21] Yusuke Miyao and Junichi Tsujii. Maximum entropy estimation for feature forests. In *Proc. of HLT-NAACL*, 2002.

[22] Shinichi Morishita and Jun Sese. Traversing itemset lattices with statistical metric pruning. In *Proc. of ACM SIGACT-SIGMOD-SIGART Symp. on Database Systems (PODS)*, pages 226–236, 2000.

[23] Tetsuji Nakagawa, Taku Kudo, and Yuji Matsumoto. Revision learning and its application to part-of-speech tagging. In *Proc. of ACL*, pages 497–504, 2002.

[24] Della Pietra, Stephen, Vincent J. Della Pietra, and John D. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[25] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proc. of EMNLP*, pages 133–142, 1996.

[26] Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc of CoNLL-LLL*, pages 127–132, 2000.

[27] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*, pages 213–220, 2003.

[28] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proc. of ICDM*, pages 721–724, 2002.

[29] Mohammed Zaki. Efficiently mining frequent trees in a forest. In *Proc. of KDD*, pages 71–80, 2002.

[30] Tong Zhang, Fred Damerau, and David Johnson. Text chunking based on a generalization of winnow. *Journal of Machine Larning Resarch*, 2:615–637, 2002.

SVM と異なるのは, 負例 (全候補解析木から正解の解析木を取りのぞいた集合) が明示的に与えれれないことである. HMM-SVM は, 学習しながら負のサポートベクターを発見していく