修士論文

可変次数 Linear-Chain CRF の効率的な計算法

指導教員　　黒橋 禎夫 教授

京都大学大学院情報学研究科
修士課程知能情報学専攻

真鍋 宏史

平成 23 年 2 月 8 日

可変次数 Linear-Chain CRF の効率的な計算法

真鍋 宏史

内容梗概

系列ラベリング問題に対する手法の一つに，Conditional Random Fields(CRF) というものがある．これは系列全体に最大エントロピーモデルを適用するというものであり，属性とラベル間の柔軟な関係性を記述できる．CRF の中でよく使われるものは Linear-Chain CRF と呼ばれるもので，一般には一次のマルコフ性を仮定し，前向き・後ろ向きアルゴリズムという一種の動的計画法によって素性の期待値を効率的に計算する．しかし，これを高次にしようとすると指数的に計算量が増加してしまう．Ye ら (2009) の研究では，高次の素性を導入しつつ多項式時間に計算量を収める手法を紹介しているが，次数の高さのため，高次の素性が少ない場合にしか適用できなかった．本研究では，新たなアルゴリズムによって，より効率的に期待値を計算する手法を提案する．また，英語の品詞タグ付けタスクに本研究のアルゴリズムを応用し，従来の一次マルコフ Linear-Chain CRF よりも高い精度を実現できた．

An Efficient Algorithm for Variable-Order Linear-Chain CRFs

Hiroshi MANABE

Abstract

Conditional Random Fields (CRFs) is one of the most popular models for sequence labeling. Applying the maximum entropy model to the whole sequence, it can describe relationship between the observations and labels with consistency. Linear-Chain CRFs is the most prevalent and practical application of CRFs which focuses on linear-chained graphs and usually makes first-order Markov assumption. It allows us to efficiently compute the model expectation of the feature functions by using forward-backward algorithm. Extending the model to higher-order Markov model, however, has been problematic due to the exponential computational complexity. Ye et al.(2009) presented an algorithm which computes the model expectation in polynomial time. However, because of its still high computational complexity, the algorithm is only practical when the higher-order features are sparse. This paper presents a novel algorithm that computes the model expectation of the feature functions more efficiently. When applied to English POS-tagging, this model yields a higher precision than the traditional first-order Markov model.

可変次数 Linear-Chain CRF の効率的な計算法

目次

第1章　はじめに

電子計算機に学習機能を備えさせ，それによって人間の望むタスクを行わせることを機械学習と呼ぶ．その中に,「系列ラベリング」と呼ばれるタスクがある．これは，与えられた観測データに対してラベルを付与するタスクであり，遺伝子解析・音声認識・画像処理等の様々な分野で利用されている．自然言語処理においても，品詞タグ付け・固有表現抽出・チャンキング等の多くの応用がある．

機械学習は一般的に教師あり学習と教師なし学習に大別できる．前者では，人間が正解を用意し，それによって計算機は未知のデータに対する推測を行う．後者では，データのみを与え，その中に存在する規則性などを計算機が推測する．この論文では，教師ありの系列ラベリングを扱う．

系列ラベリングを扱うモデルには，Hidden Markov Model (HMM)，Support Vector Machine (SVM)，Maximum Entropy Markov Model (MEMM)[1] をはじめとする様々なものがあるが，中でも Conditional Random Fields(CRF)[2] のサブセットである Linear-Chain CRF は，その精度の高さから近年よく使われるものとなっている．Linear-Chain CRF には様々なよい性質があるが，その一方で計算量が大きいという欠点があり，特にラベルについて高次の素性を使用すると計算量が次数に対して指数的に増加するため，現実的な課題に対して適用するときには，主に一次マルコフ過程のモデルで使用される．しかし，例えば英語の品詞タグ付けのタスクにおいて，高次の素性を利用することによって精度の向上が達成できることは，MEMM の一種を用いる Stanford Tagger[3] によって示されている．Linear-Chain CRF において 高次の素性を扱う場合の期待値計算やデコードにおいて，計算量を多項式時間に抑える研究として Ye ら [4] のものがすでにあるが，本研究では，さらに計算量の少ない新しい手法を提案する．それによって，高次の素性を扱うことを容易にし，タスク適用時の精度向上を実現する．

第2章　系列ラベリングのための諸モデル

$\mathbf{x}, \mathbf{y}$ はそれぞれ観測列とラベル列を表す．位置 t での観測・ラベルをそれぞれ x_t, y_t と表す．系列長は T で表し，ラベル数は L で表す．また，位置 0 と位置 $T+1$ を仮想的に考え，それぞれの位置でのラベルを l_{BOS}, l_{EOS} とする．

系列ラベリングのためのモデルは，生成モデルと識別モデルに大別される．前者は，$P(\mathbf{x}, \mathbf{y})$ という観測列とラベル列の同時確率をモデル化し，後者は $P(\mathbf{y} \mid \mathbf{x})$ という，観測列によるラベル列の条件付き確率をモデル化する．本論文では後者に属する Linear-Chain Conditional Random Fields (Linear-Chain CRF) を扱うが，これに関連の深いものとして，前者に属する Hidden Markov Model (HMM) と，後者に属する Maximum Entropy Markov Model (MEMM) について先に触れる．なお，本章ならびに次章では，各モデルのマルコフ次数はすべて 1 とする．

2.1　Hidden Markov Model (HMM)

Hidden Markov Model (HMM) は生成モデルであり，観測列とラベル列の同時確率 $P(\mathbf{x}, \mathbf{y})$ をモデル化する．各観測は，その位置でのラベルに条件付けられ，確率的に出力されるとする．各ラベルの確率は，前の位置のラベルにのみ条件付けられるとする**マルコフ性**を仮定する．このモデルでは，系列全体の確率は次のようにモデル化される．

$$P(\mathbf{x}, \mathbf{y}) = \prod_t P(y_t \mid y_{t-1}) P(x_t \mid y_t) \tag{1}$$

このモデルには，多様な**素性**が利用できないという欠点がある．素性というのは，判別に役立てるために利用する特徴を表し，**素性関数**として定式化する．例えば，英語などの言語において，ある単語が大文字で始まるかどうかということは，その単語が固有名詞であるかどうかについての重要な指標である．しかし，HMM においては，ラベルが観測を出力する確率と，ラベル間の遷移のみをモデル化するため，判別に有用な素性を利用することができない．

Linear-Chain CRF において用いられる計算法は，HMM において使用されるものと共通するところが多い．Linear-Chain CRF の期待値計算に使われる **前向き・後ろ向きアルゴリズム** (forward-backward algorithm)，デコード (ラベル付与) に使われる**ビタビアルゴリズム** (Viterbi algorithm) は，共に HMM にお

いても使用されるものである．これらのアルゴリズムについては後に詳述する．

2.2 Maximum Entropy Markov Model (MEMM)

CRF は，最大エントロピー法を利用した手法の一種である．Maximum Entropy Markov Model (MEMM) は，同じく最大エントロピー法を使った系列ラベリング手法である．識別モデルであり，モデル化するものは観測列によるラベル列の条件付き確率 $P(\mathbf{y} \mid \mathbf{x})$ である．

このモデルにおいては，あるラベルへの遷移確率が，前のラベルと観測列によって条件付けられるとする．観測列の条件は自由に定めることができる．このとき，例えば「現在位置での単語が大文字で始まる」「次の位置の単語がある特定の単語である」といった，現在の位置を基準にした条件を使うことが多い．そのため，y_t の確率は，条件に観測列 $\mathbf{x}$ に現在位置 t を加え，次のように書くことができる．

$$P(y_t \mid y_{t-1}, \mathbf{x}, t) \tag{2}$$

これに最大エントロピー法を適用し，次のように表す．

$$P(y_t \mid y_{t-1}, \mathbf{x}, t) = \frac{1}{Z(\mathbf{x}, t, y_{t-1})} \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y_t)\right) \tag{3}$$

$Z(\mathbf{x}, t, y_{t-1})$ は正規化係数と呼ばれ，あるラベルからの遷移確率の合計が 1 となるように調整する．各 f_i は素性関数であり，観測列と位置と遷移を条件とする，一般的には二値の関数である．各 λ_i は素性関数の重みと呼ばれ，$-\infty < \lambda_i < +\infty$ である．

$P(\mathbf{y} \mid \mathbf{x})$ は，各位置での確率をかけ合わせ，次のように表す．

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_t P(y_t \mid \mathbf{x}, t) \tag{4}$$

このモデルにより，訓練データの尤度を最大化するように λ_i の学習を行う．学習にあたっては，L-BFGS 法 [5] 等を用いる．MEMM には，HMM と比べて明らかな利点があり，それは素性を自由に設定できるということである．

次に述べる Linear-Chain CRF と比べての利点は，各位置ごとに遷移確率の合計を 1 として正規化を行うため，Linear-Chain CRF で必要となる前向き・後ろ向きアルゴリズムを必要とせず，計算量が少ないという点である．その代わ

りに，**ラベルバイアス**と呼ばれる問題が発生する．これは，各位置ごとに遷移確率を正規化しているため，全体として最適な系列を選べないという問題である．また，日本語の形態素解析といった，長さの異なる要素によるラティス構造に適用した場合，**レングスバイアス**という，長いものが過度に優先されてしまうという問題もある．

2.3 Linear-Chain Conditional Random Fields(Linear-Chain CRF)

Conditional Random Fields (CRF) は，系列全体に対して最大エントロピー法を適用するというものであり，MEMM にあるラベルバイアスが存在しない．CRF は一般的なグラフに適用可能なものであるが，ここでは Linear-Chain Conditional Random Fields (Linear-Chain CRF) と呼ばれる，一本の線で表されるグラフにおいての CRF について述べる．

Linear-Chain CRF においては，$P(\mathbf{y} \mid \mathbf{x})$ を次のようにモデル化する．

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y_t)\right) \tag{5}$$

$Z(\mathbf{x})$ は正規化定数であり，すべての可能な系列の確率を合計すると 1 となるようにする．各 f_i は素性関数であり，観測列と位置と遷移を条件とする，一般的には二値の関数である．f_i は y_{t-1} と y_t を引数に含むが，y_{t-1} によらないものを特に **unigram 素性**と呼び，それ以外を **bigram 素性**と呼ぶ．各 λ_i は素性関数の重みであり，$-\infty < \lambda_i < +\infty$ である．

学習時には，訓練データセットの尤度を最大化するように，各 λ_i の値を決める．それには L-BFGS 法 [5] 等を用いる．このとき，適当な初期値で各パラメータを初期化した後，各素性の訓練データセット中での出現回数と，現在のモデルによるそれらの期待値，訓練データセットの対数尤度を用いてパラメータを更新していく．これらの値を求めるにあたって，ナイーブな実装では計算量が系列長に対して指数関数的に増加するため，**前向き・後ろ向きアルゴリズム**を使用する．また，デコード時には**ビタビアルゴリズム**を使用する．パラメータ更新は，対数尤度の更新量が十分に少なくなるまで行う．

第3章　Linear-Chain CRF のアルゴリズム

Linear-Chain CRF においては，パラメータ推定時には**前向き・後ろ向きアルゴリズム**を，デコード時には**ビタビアルゴリズム**を使用する．これらのアルゴリズムは Linear-Chain CRF 以外にも適用されるが，ここでは Linear-Chain CRF での場合を扱う．

3.1　前向き・後ろ向きアルゴリズム

Linear-Chain CRF のパラメータ推定では，訓練データ $\mathcal{T}$ が与えられたとき，正規化された対数尤度

$$\mathcal{L}_{\mathcal{T}} = \log \prod_{(\mathbf{x},\tilde{\mathbf{y}})\in\mathcal{T}} P(\tilde{\mathbf{y}} \mid \mathbf{x}) - \sum_i \frac{\lambda_i^2}{2\sigma_{reg}^2} \tag{6}$$

を最大化するようにモデルのパラメータ λ_i を学習する．σ_{reg} は正規化のためのパラメータである．

各 λ_i についての $\partial\mathcal{L}_{\mathcal{T}}$ の偏微分は，$\tilde{E}[f_i]$ を訓練データ中での f_i の経験的な期待値，$E[f_i]$ を f_i の現在のモデルによる期待値として，

$$\frac{\partial\mathcal{L}_{\mathcal{T}}}{\partial\lambda_i} = \tilde{E}[f_i] - E[f_i] - \frac{\lambda_i}{\sigma_{reg}^2} \tag{7}$$

として表せる．$\mathcal{L}_{\mathcal{T}}$ は各 λ_i について凹関数であるため，それぞれの i について，式 (7) を 0 とすることで最大化できる．

このとき，現在のモデルパラメータによる各素性の期待値が必要となるが，これには**前向き・後ろ向きアルゴリズム** (forward-backward algorithm) を使用する．これは動的計画法の一種である．動的計画法とは，部分問題に対する解を求め，それらを記録することによって，全体の解を効率的に求める方法である．前向き・後ろ向きアルゴリズムにおいては，各位置 t の各ラベル y について，次のように α, β を定義する．

$$\alpha(y,t) \stackrel{\text{def}}{=} \sum_{\mathbf{y}_{0:t-1}} \exp\left(\sum_{t'=1}^{t-1}\sum_i \lambda_i f_i(\mathbf{x}, t', y_{t'-1}, y_{t'}) + \sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y)\right) \tag{8}$$

$$\beta(y,t) \stackrel{\text{def}}{=} \sum_{\mathbf{y}_{t+1:T+1}} \exp\left(\sum_{t'=t+2}^{T+1}\sum_i \lambda_i f_i(\mathbf{x}, t', y_{t'-1}, y_{t'}) + \sum_i \lambda_i f_i(\mathbf{x}, t, y, y_{t+1})\right) \tag{9}$$

ただし，$\alpha(l_{BOS}, 0) = 1, \beta(l_{EOS}, T+1) = 1$ とする．

このように定義すると，各 $\alpha(y,t), \beta(y,t)$ は次のように $\alpha(y', t-1), \beta(y', t+1)$ を使って次のように表せる．

$$\alpha(y,t) = \sum_{y'} \alpha(y', t-1) \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y', y)\right) \tag{10}$$

$$\beta(y,t) = \sum_{y'} \beta(y', t+1) \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y, y')\right) \tag{11}$$

動的計画法により，すべての α, β は $O(L^2T)$ で計算できる．手続きは，Algorithm 1 に示す前向き部分と，Algorithm 2 に示す後ろ向き部分に分けられる．

Algorithm 1 Forward-backward: forward part

$\alpha(l_{BOS}, 0) \leftarrow 1$
for $t = 1$ **to** $T + 1$ **do**
  for all y_t **do**
    $\alpha(y_t, t) \leftarrow \sum_{y'} \left(\alpha(y', t-1) \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y', y_t)\right)\right)$
  end for
end for

Algorithm 2 Forward-backward: backward part

$\beta(l_{EOS}, T+1) \leftarrow 1$
for $t = T$ **downto** 0 **do**
  for all y_t **do**
    $\beta(y_t, t) \leftarrow \sum_{y'} \left(\beta(y', t+1) \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y_t, y')\right)\right)$
  end for
end for

このようにして求めた α, β によって，位置 $t-1$ から t における，ラベル y' から y への遷移の期待値は次のように表せる．

$$P(y_{t-1} = y', y_t = y) = \frac{1}{Z(\mathbf{x})} \alpha(y', t-1) \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y', y)\right) \beta(y, t) \tag{12}$$

また，正規化係数 $Z(\mathbf{x})$ は次のように表せる．

$$Z(\mathbf{x}) = \alpha(l_{EOS}, T+1) = \beta(l_{BOS}, 0) \tag{13}$$

3.2 ビタビアルゴリズム

Linear-Chain CRF のデコードとは，与えられた $\mathbf{x}$ について，それを条件としたときにモデルによる確率が最大となる $\mathbf{y}^*$ を求めることである．**ビタビアルゴリズム** (Viterbi algorithm) はこのときに使用される．これも動的計画法の一つである．

まず，求める $\mathbf{y}^*$ は次のように表すことができる．

$$\begin{aligned} \mathbf{y}^* &= \arg\max_{\mathbf{y}} \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y_t)\right) \\ &= \arg\max_{\mathbf{y}} \exp\left(\sum_t \sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y_t)\right) \end{aligned} \tag{14}$$

ここで，$p(y_t, t), q(y_t, t)$ を次のように定義する．

$$p(y_t, t) \stackrel{\text{def}}{=} \max_{y_{t-1}} \left(p(y_{t-1}, t-1) \exp\left(\sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y_t)\right)\right) \tag{15}$$

$$q(y_t, t) \stackrel{\text{def}}{=} \arg\max_{y_{t-1}} p(y_t, t) \tag{16}$$

各位置の各ラベルにおいて，p はそこに至る最大得点を記録し，q はそこに至る最大得点を与える前位置のラベルを記録する．ビタビアルゴリズムにおいては，これらの値を Algorithm 3 のように，前向きに順次求めていく．末尾まで達したら，Algorithm 4 のように，最大得点を与える末尾ラベルから，最大得点を与える前位置のラベルを記録した q を順次逆向きにたどり，‘$\mathbf{y}^*$ を決定する．

これらの計算量は，L^2T である．

Algorithm 3 Viterbi algorithm(1)

$p(l_{BOS}, 0) \leftarrow 1$

for $t = 1$ **to** $T + 1$ **do**

  for all y_t **do**

    $p(y_t, t) \leftarrow \max_{y_{t-1}} \left(p(y_{t-1}, t-1) \exp \left(\sum_i \lambda_i f_i(\mathbf{x}, t, y_{t-1}, y_t) \right) \right)$

    $q(y_t, t) \leftarrow \arg\max_{y_{t-1}} \left(p(y_t, t) \right)$

  end for

end for

Algorithm 4 Viterbi algorithm(2)

for $t = T$ **to** 1 **do**

  $y_t^* \leftarrow q(y_{t+1}^*, t+1)$

end for

第4章　可変次数 Linear-Chain CRF

ここまでは，マルコフ次数が1である場合に限定して Linear-Chain CRF に使用されるアルゴリズムを紹介したが，それを高次マルコフ過程に適用する場合について考える．

n 次マルコフ過程とは，現在のラベルが過去 n 個のラベルに条件付けられるというモデルである．n 次マルコフ過程は，それら n 個のラベルの組をまとめたものによる新たなラベルを考えることにより，1次マルコフ過程として表現できる．そのため，高次 Linear-Chain CRF においても，同じアルゴリズムによってパラメータ推定・デコードが可能である．

n 個のラベルの組をまとめた新たなラベルの数は，すべての組み合わせを考えると L^n となる．遷移はその2乗とはならない．なぜならば，一つの組み合わせラベルから別の組み合わせラベルに対して，可能な遷移は限られるからである．例えば，元のラベル集合を $\{a, b, c\}$ とするとき，可能な組み合わせラベルは9通りあるが，例えば組み合わせラベル acb から遷移する組み合わせラベルは cba, cbb, cbc の3通りのみである．このように，一つの組み合わせラベル (L^n 通り) に対して元のラベル数 (L 通り) だけの遷移が存在するため，遷移数は L^{n+1} となる．そのため，前向き・後ろ向きアルゴリズム，ビタビアルゴリズム共に，計算量は $L^{n+1}T$ となる．これは次数 n に対して指数関数的であり，計算量もそれに比例するため，高次素性の利用は現実的には難しい．そこで，以下に述べる「疎素性集合」の考え方によって素性関数を選択的に設定する．

4.1　密素性集合・疎素性集合

Linear-Chain CRF の素性関数を考えるとき，一つの方法として「可能な遷移すべてに対して素性を設定する」というものがある (ここでは，そのようにして設定した素性の集合を「密素性集合」と呼ぶ)．例えば，1次であれば，$L \times L$ の遷移すべてに対して素性関数を与える．

もう一つの方法として,「一部の遷移のみに対して素性関数を設定する」というものがある (ここでは，そのようにして設定した素性の集合を「疎素性集合」と呼ぶ)．この場合，遷移を選ぶ基準としてはいろいろな可能性が考えられるが,最も自然なものは「訓練データセット中に出現する遷移のみを使う」というものであろう．

訓練データセット中に出現する遷移についての素性関数は，出現しないものよりも有用であると考えられる．しかし，訓練データセット中に出現しない遷移に対して素性関数を設定することにも利点がある．出現しない遷移についても，異なる重みを与えることが適当であるような場合である．例えば，次のような例が考えられる．

ラベル集合を $\{a, b, c\}$ とし，訓練データセット中では $a > b > c$ の順に出現頻度が多いとする．可能な遷移のうち，出現したものは $aa, ab, ac, ba, bb, bc, cb$ のみで，ca, cc はないとする．訓練過程では，unigram 素性として a, b, c のそれぞれの出現を支持するものが学習されるが，bigram 素性は，訓練データセット中に出現しない ca, cc という遷移には素性関数が設定されないため，その学習は行われない．出現頻度から考えると，cc が出現しないことは，ca が出現しないことほどには不自然ではない．もし，ca, cc に対する素性関数が存在すれば，ca に対して，より小さな重みが学習されるはずである．ca, cc に対する素性関数が存在しなければ，cb に対応する素性関数に対する重みを大きく学習することでその代わりとすることになるが，これは cc の出現に対しては過度に不利になり，ca の出現に対しては過度に有利になる．

このように，密素性集合を使うことにも利点がある．CRFSuite[6] での実験によると，英語のチャンキングというタスクにおいて，密素性集合を使用した場合に精度が 96.06% であったのに対し，訓練データセット中に出現する遷移に限った疎素性集合を使用した場合には 96.00% であった．

とはいえ，その差は大きくなく，学習の主要な部分が訓練データセット中に出現する遷移に対応する素性関数に対する重みによるものであることがわかる．

密素性集合の考え方により素性関数を設定すると，次数に対して指数関数的に素性関数の数が増加し，それに伴い計算複雑性が増加することは避けられない．そのため，ここでは疎素性集合を使用することを考える．

疎素性集合を使用する場合，位置・ラベルによってマルコフ次数が変わることになる．そのように素性を設定した Linear-Chain CRF を，ここでは**可変次数 Linear-Chain CRF**(Variable-Order Linear-Chain CRF) と呼ぶことにする．このモデル自体は，次に挙げる Ye ら [4] のものと共通であり，本研究の新規性は，このモデルにおいての計算量を小さくする計算法にある．

また，高次の素性を扱う CRF の研究としては，Qian ら [7] による Sparse Higher Order CRFs(SHO-CRFs) というものもあり，次で簡単に紹介する．

4.2 関連研究:Ye(2009)

Ye ら [4] は，本研究と同じく，多項式時間で高次 Linear-Chain CRF のパラメータ推定を行う方法を提案している．

彼らの手法においては，素性関数のユニークなラベル列数を M，素性関数の最大次数を K，訓練データの長さを X とするとき，悲観的な見積もりでは一回のイテレーションに $O(M^3K^4X)$ の計算量が必要となる．本研究での計算複雑性は，訓練データ中のそれぞれの位置でアクティブになる素性関数が持つユニークなラベル列の長さの合計であり，これは最大でも $O(MKX)$ である．正確な計算量は，各位置において 1 となる素性関数の数と，それらの持つユニークなラベル列の数に依存するため，次の章で具体的に述べる．本研究においては，計算量を下げることにより高次素性の導入を容易にし，その結果としてより柔軟な素性選択を可能にしている．

4.3 関連研究:Qian(2009)

Qian らによる sparse higher order CRFs (SHO-CRFs) [7] では，$\mathbf{Z}_{s:t} = Z_s \times Z_{s+1} \times \cdots \times Z_t$ の形で表されるラベル列に対して素性関数を割り当てるという方法をとっているため，本研究とは異なるクラスの問題を対象としている．本研究で扱うのは，各素性関数がそれぞれ一つの対応するラベル列をもつというモデルであり，Qian らの研究はこのような問題を多項式時間の計算量で扱えるようなものではない．

第5章　可変次数 Linear-Chain CRF に対する本研究の計算法

この章では，本研究の主となる，可変次数 Linear-Chain CRF のための三つのアルゴリズムについて記す．一つは，パラメータ推定のために素性関数の期待値を計算する「合計・差分アルゴリズム」である．これは,「差分得点」と「合計得点」を考え,「前向き差分得点」を求めるときに「前向き合計得点」を,「後ろ向き合計得点」を求めるときに「後ろ向き差分得点」をそれぞれ補助的に使用することで，動的計画法の適用を可能にしている．

残り二つは，デコードのアルゴリズムである．共に，観測に対する最適なラベルを推測するものであるが，一つは前向きに，もう一つは後ろ向きに処理を行う．

これら三つのアルゴリズムは，すべて動的計画法の応用である．

5.1　表記

ここでは，この章で使う表記について述べる．

$\mathbf{x}, \mathbf{y}$ はそれぞれ長さ T の観測列・長さ T のラベル列を表す．$|\cdot|$ は列の長さを表す．ラベルの集合は $\mathcal{Y} = \{l_1, \cdots, l_L\}$ とする．また，位置 0 と位置 $T+1$ を仮想的に考え，それぞれの位置でのラベルを l_{BOS}, l_{EOS} とする．

位置 t でのラベル集合 $\mathcal{Y}_t$ を次のように定義する．これは，系列の開始位置と終端位置を扱うためである．

$$\mathcal{Y}_t = \begin{cases} \{l_{BOS}\} & \text{if } t = 0 \\ \{l_{EOS}\} & \text{if } t = T+1 \\ \mathcal{Y} & \text{otherwise} \end{cases} \tag{17}$$

ラベル集合の列 $\mathcal{Y}_n, \cdots, \mathcal{Y}_m$ に対して，あるラベル列 $\mathbf{z}$ が $|\mathbf{z}| = m-n+1, \forall(t : 1 \leq t \leq m-n+1)\mathbf{z}_n \in \mathcal{Y}_{n+t-1}$ を満たすとき，$\mathbf{z} \in \mathcal{Y}_{n:m}$ とする．

例えば，$L = 3, T = 2$ であるとき，$\mathcal{Y}_0 = \{l_{BOS}\}, \mathcal{Y}_1 = \{l_1, l_2, l_3\}, \mathcal{Y}_2 = \{l_1, l_2, l_3\}, \mathcal{Y}_3 = \{l_{EOS}\}$ となり，$\{\mathbf{z} \mid \mathbf{z} \in \mathcal{Y}_{0:3}\} = \{l_{BOS}l_1l_1l_{EOS}, l_{BOS}l_1l_2l_{EOS}, l_{BOS}l_1l_3l_{EOS}, l_{BOS}l_2l_1l_{EOS}, l_{BOS}l_2l_2l_{EOS}, l_{BOS}l_2l_3l_{EOS}, l_{BOS}l_3l_1l_{EOS}, l_{BOS}l_3l_2l_{EOS}, l_{BOS}l_3l_3l_{EOS}\}$ となる．

任意のラベル列 $\mathbf{z}$ に対し，$\mathbf{z}_{i:j} \stackrel{\text{def}}{=} \mathbf{z}_{i \cdots j}$ とする．$j < i$ であれば，$\mathbf{z}_{i:j}$ は空の列

(ϵ と表す) とする．また，表記の都合上，任意のラベル列について述べるとき，$\mathbf{z}^1, \mathbf{z}^2, \cdots$ のように右上に通し番号を付ける．

あるラベル列 $\mathbf{z}^1$ とラベル列 $\mathbf{z}^2$ をこの順に連結したものが $\mathbf{z}^3$ となるとき，$\mathbf{z}^3 = \mathbf{z}^1 + \mathbf{z}^2$ とする．

あるラベル列 $\mathbf{z}^1$ がラベル列 $\mathbf{z}^2$ の**接尾辞**である ($\mathbf{z}^2$ が $\mathbf{z}^1$ を含み，末尾を共有する) とき，$\mathbf{z}^1 \leq_s \mathbf{z}^2$ と表記する ($\mathbf{z}^1 = \mathbf{z}^2$ を含む)．任意のラベル列 $\mathbf{z}$ に対し，$\epsilon \leq_s \mathbf{z}$ とする．ラベル列 $\mathbf{z}^1$ が $\mathbf{z}^2$ の接尾辞でないということを，$\mathbf{z}^1 \not\leq_s \mathbf{z}^2$ と表す．また，$\mathbf{z}^1$ が $\mathbf{z}^2$ の接尾辞であり，$\mathbf{z}^1 \neq \mathbf{z}^2$ であるとき，$\mathbf{z}^1 <_s \mathbf{z}^2$ と表記し，$\mathbf{z}^1$ は $\mathbf{z}^2$ の**純接尾辞**であるとする．このとき，次の式が明らかに成り立つ．

$$\mathbf{z}^1 <_s \mathbf{z}^2, \mathbf{z}^1 \neq \epsilon \Rightarrow \mathbf{z}^1_{1:|\mathbf{z}^1|-1} <_s \mathbf{z}^2_{1:|\mathbf{z}^2|-1} \tag{18}$$

これは，$\mathbf{z}^1$ が $\mathbf{z}^2$ の純接尾辞であり，$\mathbf{z}^1 \neq \epsilon$ であれば，$\mathbf{z}^1$ から末尾の 1 ラベルを削除したものは $\mathbf{z}^2$ の純接尾辞になるということを表す (接尾辞に関しても同様のことがいえる)．

また，次も明らかに成り立つ．

$$\mathbf{z}^1 <_s \mathbf{z}^3, \mathbf{z}^2 <_s \mathbf{z}^3 \Rightarrow \mathbf{z}^1 <_s \mathbf{z}^2 \text{ or } \mathbf{z}^2 \leq_s \mathbf{z}^1 \tag{19}$$

これは，$\mathbf{z}^1, \mathbf{z}^2$ が共に $\mathbf{z}^3$ の純接尾辞であれば，$\mathbf{z}^1$ が $\mathbf{z}^2$ の純接尾辞であるか，$\mathbf{z}^2$ が $\mathbf{z}^1$ の接尾辞であるかのどちらかであることを示す．

また，$\mathbf{z}^2$ が，集合 $\mathcal{S}$ に含まれる要素の中で最も長い $\mathbf{z}^1$ の純接尾辞であるとき，それを $s(\mathbf{z}^1, \mathcal{S}) = \mathbf{z}^2$ と表わし，$\mathbf{z}^2$ を $\mathcal{S}$ に関する $\mathbf{z}^1$ の**最長純接尾辞**であるとする ($\mathbf{z}^1$ は $\mathcal{S}$ の要素でなくてもよい)．定義は次のようになる．

$$\begin{aligned} &s(\mathbf{z}^1, \mathcal{S}) = \mathbf{z}^2 \text{ if and only if } \quad \mathbf{z}^2 \in \mathcal{S} \text{ and } \mathbf{z}^2 <_s \mathbf{z}^1 \\ &\text{and } \forall(\mathbf{z} \in \mathcal{S})\mathbf{z} <_s \mathbf{z}^1 \Rightarrow \mathbf{z} \leq_s \mathbf{z}^2 \end{aligned} \tag{20}$$

また，$\mathbf{z}^2$ が，集合 $\mathcal{S}$ に含まれる要素の中で最も長い $\mathbf{z}^1$ の接尾辞であるとき，それを $S(\mathbf{z}^1, \mathcal{S}) = \mathbf{z}^2$ と表わし，$\mathbf{z}^2$ を $\mathcal{S}$ に関する $\mathbf{z}^1$ の**最長接尾辞**であるとする．定義は次のようになる．

$$\begin{aligned} &S(\mathbf{z}^1, \mathcal{S}) = \mathbf{z}^2 \text{ if and only if } \quad \mathbf{z}^2 \in \mathcal{S} \text{ and } \mathbf{z}^2 \leq_s \mathbf{z}^1 \\ &\text{and } \forall(\mathbf{z} \in \mathcal{S})\mathbf{z} \leq_s \mathbf{z}^1 \Rightarrow \mathbf{z} \leq_s \mathbf{z}^2 \end{aligned} \tag{21}$$

ある $\mathbf{z}$ が $\mathcal{S}$ の要素であるとき，またそのときに限り，$S(\mathbf{z}, \mathcal{S}) = \mathbf{z}$ である．

なお，任意の $\mathcal{S}$ に対し，$s(\epsilon, \mathcal{S}) = \perp$ とおく．$\perp$ は実体を持たない仮想的なラベル列で，他のラベル列と比較したときに一致することはない ($\forall(\mathbf{z})\mathbf{z} \neq \perp$ である)．

また，$\perp^n$ を，他のラベル列と比較したときに一致することのない，長さ n のラベル列とする ($\forall(\mathbf{z})\mathbf{z} \neq \perp^n, |\perp^n| = n$ である)．

素性関数を $f_1, \cdots, f_m$ とする．素性関数 f_i は，$\mathbf{x}, t$ の二値関数 $b_i(\mathbf{x}, t)$(「属性関数」と呼ぶ) と，二値関数 $L_i(\mathbf{y}, t)$(「ラベル関数」と呼ぶ) の積として表されるとする．

$$f_i(\mathbf{x}, \mathbf{y}, t) = b_i(\mathbf{x}, t) L_i(\mathbf{y}, t) \tag{22}$$

L_i は対応するラベル列 $\mathbf{z}^i$ を持っており，次のように定義する．

$$L_i(\mathbf{y}, t) = \begin{cases} 1 & \text{if } \mathbf{y}_{t-|\mathbf{z}^i|+1:t} = \mathbf{z}^i \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

例えば，ある $\mathbf{z}^i = l_1 l_2$ であれば，$y_{t-1} = l_1, y_t = l_2$ であるときに $L_i(\mathbf{y}, t) = 1$ となる．$\mathbf{z}^i$ と関連づけられた L_i を，次数 $|\mathbf{z}^i| - 1$ のラベル関数と呼ぶ．また，それを構成要素に持つ f_i を，次数 $|\mathbf{z}^i| - 1$ の素性関数と呼ぶ．

5.2 合計・差分アルゴリズム

現在のパラメータによる素性関数 f_i のモデル期待値は，次の式によって表される．

$$E[f_i] = \sum_{(\mathbf{x}, \tilde{\mathbf{y}}) \in \mathcal{T}} \sum_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \sum_{t=|\mathbf{z}^i|}^{|\mathbf{x}|+1} f_i(\mathbf{x}, \mathbf{y}, t) \tag{24}$$

これは，すべての観測に対してすべての可能な系列を考え，現在のパラメータによるそれぞれの確率にそれぞれの系列で 1 となる素性関数の出現回数を合計するということである．系列長は，ここでは訓練データ全体にわたって考えるため，T ではなく $|\mathbf{x}|$ で表している．また，$|\mathbf{x}| + 1$ という仮想的な位置に対しても素性関数を設定する．これは，素性関数は現位置までの遷移を引数にとるため，系列終端への遷移を考えるときに必要となるからである．系列開始からの遷移については，位置 $t = 1$ において 1 次の素性関数を設定することで十分であり，位置 $t = 0$ には素性関数を設定する必要はない．

以下に，式 (24) の効率的な計算法を示す．このアルゴリズムによって得られる素性関数の期待値と，訓練データから得られる素性関数の経験的な出現回数

と，現在のパラメータによるモデルの対数尤度によって，L-BFGS 法 [5] により素性関数の重みを推定することができる．

以下に，本アルゴリズムで使用する各変数を定義する．それにあたって，具体的な例として図 1 に示すような場合を考える．

下部の縦列は，それぞれ位置 $0 \leq t \leq T+1$(この例では $T=3$) を表している．$0, T+1$ は系列の開始位置・終端位置を表す仮想的な位置である．各列の最上部は，“/” の前が観測文字列 (アルゴリズム中では使用しない)，後ろがその位置に付与される属性を表す．この例では，a1 は「観測文字列が s で終わる」，a2 は「観測文字列が大文字で始まる」，a3 は「観測文字列が d で終わる」，a4 は「文末である」という条件によって生成されている (素性関数は文末位置にも設定可能である．文頭には設定できない)．a0 は観測文字列によらず常に付与される．これは，ラベル遷移のみによる素性関数を設定するために必要となる．各列に並んだ矩形は，それぞれが「パス」を表している.「パス」については後述する．

上部の枠内は，それぞれ素性関数を表している．“:” で区切られたそれぞれの値は，属性の条件 (b_i にあたる)，ラベル列 (L_i にあたる)，重みの $\exp(\exp(\lambda_i)$ にあたる) を表す (以降，重みの exp をとったものを「指数重み」と呼ぶ)．例えば，“a3:[YZ]:2.80” という素性関数は，ある位置に a3 という属性が付与されており，かつそこまでのラベル列が YZ で終わる場合に 1 という値を取り，その指数重みが 2.80 であるということを表す.

実際の応用では，使用者が属性関数 $b_i(\mathbf{x}, t)$ のとる値を訓練データの各位置について記述し，そこまでのラベル列の履歴 $\mathbf{z}^i$ と組み合わせて次数 $|\mathbf{z}^i| - 1$ の素性関数 $f_i(\mathbf{x}, \mathbf{y}, t) = b_i(\mathbf{x}, t)L(\mathbf{y}, t)$ を生成するが，図 1 の例では，素性関数は恣意的に設定してある．

l_{BOS}, l_{EOS} は，系列開始と系列終端の位置にそれぞれ付与する仮想的なラベルである．図 1 においては，$\mathcal{Y} = \{X, Y, Z\}$ とし，$l_{BOS} = B, l_{EOS} = E$ としている．

5.2.1 パス

位置 t での「パス集合」として，$\mathcal{P}_t$ を次のように定義する．

$$\mathcal{P}_t = \mathcal{Y}_t \cup \{\epsilon\} \cup \bigcup_{t'=t}^{T} \{\mathbf{z}^k_{1:|\mathbf{z}^k|-(t'-t)} \mid b_k(\mathbf{x}, t) = 1, |\mathbf{z}^k| \geq t' - t\} \tag{25}$$

$\mathcal{P}_t$ を位置 t における「パス集合」とし，その要素を位置 t における「パス」とする．

パスとは，基本的には，ある位置で属性についての条件を満たす素性関数に関連付けられたラベル列の集合である．

本研究のモデルでは，式 (22) で示したように素性関数 f_i は属性関数 b_i とラベル関数に L_i に分けられる．ある位置におけるパス集合は，その位置で属性関数 b_i が 1 になるような素性関数 f_i のラベル関数 L_i に関連付けられたラベル列 $\mathbf{z}_i$ の集合を含む．

図 1 の例では，下段に縦に並ぶ矩形のそれぞれがパスを表している．それぞれのパスにおいて，“:” の前はそのパスのラベル列であり，後ろのそれぞれの数値は，そのパスに対応する $\exp(w), \exp(W), \sigma$(上段) と $\alpha, \gamma, \beta, \delta$(下段) を順に表している．これらの値については以下に述べる (順序がアルファベット順でないことに注意．また，図の中ではスペースの都合上 “exp” を省略している)．

この例では，位置 $t = 3$ では a0, a3 という属性が与えられており，このときに属性関数が 1 となるような素性関数は “a0:[X]:0.10”, “a0:[Y]:0.20”, “a0:[Z]:0.30”, “a0:[XY]0.70”, “a0:[XYZ]:0.80”, “a0:[YX]:0.90”, “a0:[YZ]:1.00”, “a0:[ZY]:1.10”, “a3:[YYY]:2.50”, “a3:[Y]:2.60”, “a3:[Z]:2.70”, “a3:[YZ]:2.80”, の 12 個であるが，ユニークなラベル列は $X, Y, Z, YX, XY, YYY, ZY, YZ, XYZ$ の 9 通りである．これに，ラベル列 ϵ を含めた 10 個が $t = 3$ でのパス集合となっている．あるパスに対して，そのラベル列を持つ素性関数を「関連付けられた素性関数」と呼ぶ．例えば，ラベル列 YZ に関連付けられた素性関数は “a0:[YZ]:1.00” と “a3:[YZ]:2.80” の二つである．

式 (25) によって定義される，ある位置のパス集合は，上記のもの以外に，後の位置のパス集合に含まれるラベル列の接頭辞も含む．例えば，図 1 の例では，位置 $t = 2$ に YY というパスがある．位置 $t = 2$ での属性 $a0, a1$ に関連付けられた素性関数には YY というラベル列を持つものはないが，位置 $t = 3$ において YYY というパスがあるため，位置 $t = 2$ ではそれより長さが 1 少ない YY という接頭辞を，位置 $t = 2$ においてパスとして作る．このようにして作られたパスは関連付けられた素性関数を持たない．

ある位置にパスがあれば，その前の位置に，そのラベル列の末尾を削除した

パスが必ず存在する．このことは，次のように表せる．

$$\mathbf{z} \in \mathcal{P}_t, \mathbf{z} \neq \epsilon, t > 0 \Rightarrow \mathbf{z}_{1:|\mathbf{z}|-1} \in \mathcal{P}_{t-1} \tag{26}$$

また，式 (25) のパス集合の定義より，ある位置 t においてあるパス $\mathbf{z}$ を接頭辞として持つような系列のその位置以降のスコアは，そのパスのその位置での最長接尾辞を接頭辞として持つような系列のその位置以降のスコアとして表せる．このことは，次のように表せる．

$$\sum_{\mathbf{z}' \in \mathcal{Y}_{t:T+1}} \sum_{t'=t}^{T+1} \sum_i f_i(\mathbf{x}, \mathbf{z} + \mathbf{z}', t) = \sum_{\mathbf{z}' \in \mathcal{Y}_{t:T+1}} \sum_{t'=t}^{T+1} \sum_i f_i(\mathbf{x}, S(\mathbf{z}, \mathcal{P}_t) + \mathbf{z}', t) \tag{27}$$

ここで，$f_i(\mathbf{x}, \mathbf{z} + \mathbf{z}', t)$ は $f_i(\mathbf{x}, \perp^{t'+1-|\mathbf{z}|} + \mathbf{z} + \mathbf{z}', t)$ の，$f_i(\mathbf{x}, S(\mathbf{z}, \mathcal{P}_t) + \mathbf{z}', t)$ は $f_i(\mathbf{x}, \perp^{t'+1-|S(\mathbf{z}, \mathcal{P}_t)|} + S(\mathbf{z}, \mathcal{P}_t) + \mathbf{z}', t)$ の，それぞれ略記である．以降，このような略記を使う．

式 (27) が成り立たないとすると，ある位置 t'' で，$|\mathbf{z}''| = t'' - t$ であるような $\mathbf{z}''$ というラベル列と，$S(\mathbf{z}) <_s \mathbf{z}' \leq_s \mathbf{z}$ であるような $\mathbf{z}'$ に対して，$b_i(\mathbf{x}, t'') = 1, L(\mathbf{z}' + \mathbf{z}'', t) = 1$($L$ に関しては略記である) となるような f_i が存在する．その場合，位置 t'' にはラベル列 $\mathbf{z}' + \mathbf{z}''$ を持つパスが存在することになり，式 (26) より，位置 t にはラベル列 $\mathbf{z}'$ を持つパスが存在することになる．しかし，$S(\mathbf{z}, \mathcal{P}_t) <_s \mathbf{z}' \leq_s \mathbf{z}$ であるため，$\mathbf{z}' \in \mathcal{P}_t$ であるとすると，式 (21) の最長接尾辞の定義に矛盾する．

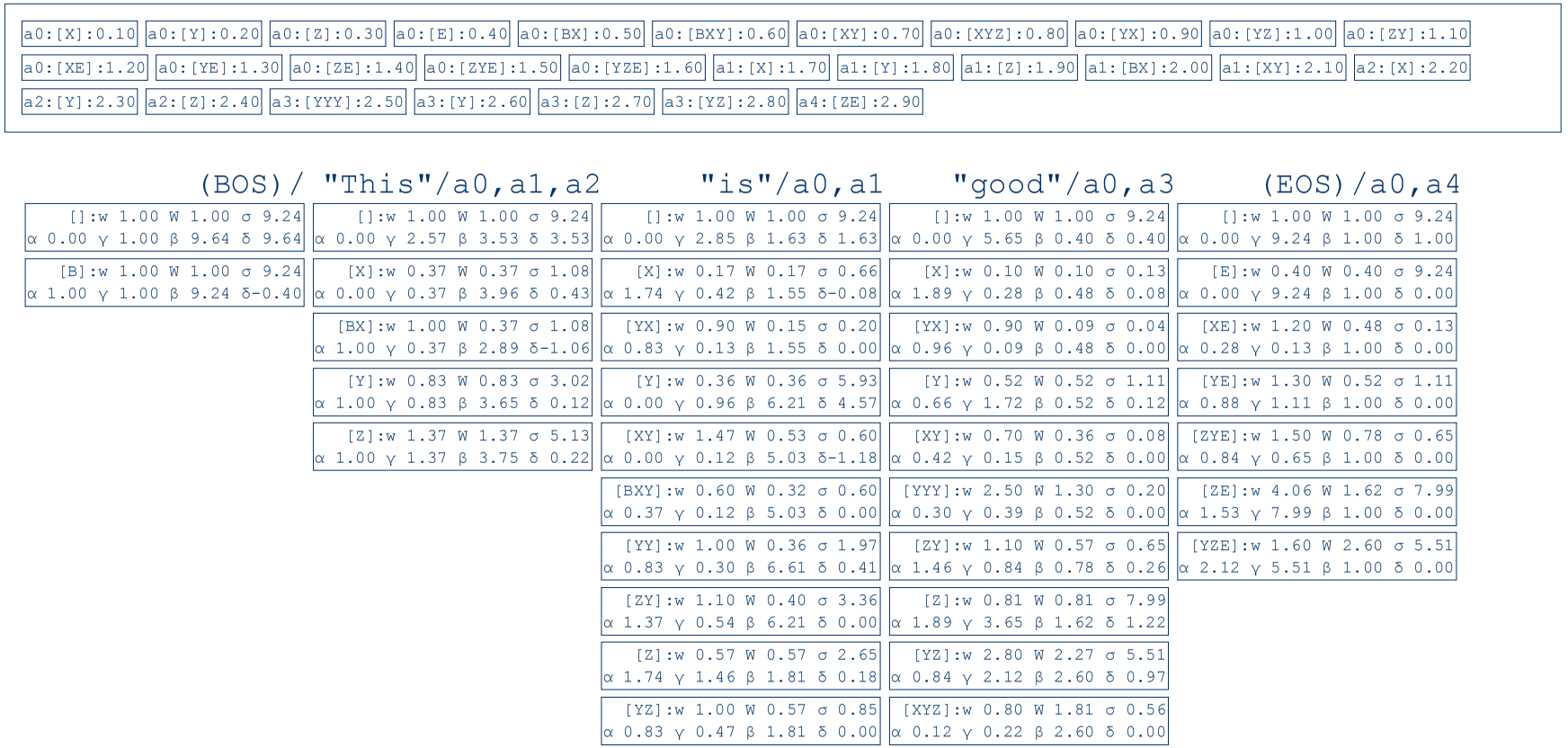

図 1: 合計・差分アルゴリズムにおける各変数

5.2.2 重み

あるパス $\mathbf{z}^p \in \mathcal{P}_t$ と位置 t について，$w(\mathbf{z}^p, t)$ を次のように定義する．

$$w(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{i:\mathbf{z}^i=\mathbf{z}^p, b_i(\mathbf{x},t)=1} \lambda_i \tag{28}$$

これは，あるパスに関連付けられた素性関数の重みを足し合わせたものである．図 1 の例では，重みはすべて exp をとってあるため，足し合わせるのではなく，かけ合わせたものとなっている．例えば，位置 $t = 2$ における XY というパスには，“a0:[XY]:0.70” と “a1:[XY]:2.10” という二つの素性関数が関連付けられているため，そのパスの指数重みは 1.47 となっている．

図 1 の例では，矩形で表されるそれぞれのパスにおいて，$\exp(w)$ は上段左端の数値にあたる．図の中では，exp を省略して “w” として表記されている．

また，あるパス $\mathbf{z}^p \in \mathcal{P}_t$ と位置 t について，$W(\mathbf{z}^p, t)$ を次のように定義する．

$$W(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{i:\mathbf{z}^i \leq_s \mathbf{z}^p, b_i(\mathbf{x},t)=1} \lambda_i \tag{29}$$

これは，あるパスについて，そのパス自身と，そのパスの接尾辞に関連付けられた素性関数の重みを足し合わせたもの (指数重みで考えると，かけ合わせたもの) である．このように重みを足し合わせるのは，あるパスを通るとき，そのパスに関連付けられた素性関数だけでなく，その接尾辞に関連付けられた素性関数も 1 となるからである．W は，式 (29) と式 (28) より，w を使って次のように表せる．

$$W(\mathbf{z}^p, t) = \sum_{i:\mathbf{z}^i \in \mathcal{P}_t, \mathbf{z}^i \leq_s \mathbf{z}^p} w(\mathbf{z}^i, t) \tag{30}$$

例えば，位置 3 で YX のラベル列を持つパスについて考えると，その指数重み $\exp(w)$ は 0.90 であり，その接尾辞である X というラベル列を持つパスの指数重みは 0.10 であるため，$\exp(W)$ は 0.09 となっている．

図 1 の例では，矩形で表されるそれぞれのパスにおいて，$\exp(W)$ は上段左から 2 番目の数値にあたる．図の中では，exp を省略して “W” として表記されている．

上のように W を定義すると，$\mathbf{z}^p \neq \epsilon$ について，$W(\mathbf{z}^p, t) = W(s(\mathbf{z}, \mathcal{P}_t), t) + w(\mathbf{z}^p, t)$ がいえる．$s(\mathbf{z}^p, \mathcal{P}_t)$ は，式 (20) で示した，$\mathcal{P}_t$ に関する $\mathbf{z}^p$ の「最長純接尾辞」であり，$\mathcal{P}_t$ に関する $\mathbf{z}^p$ の接尾辞で $\mathbf{z}^p$ 自身以外のものはすべて $s(\mathbf{z}, \mathcal{P}_t)$ に含まれるためである．

5.2.3 前向き変数

あるパス $\mathbf{z}^p \neq \epsilon \in \mathcal{P}_t$ と位置 $t \geq 1$ について,「前向き差分得点」である $\alpha(\mathbf{z}^p, t)$ を次のように定義する.

$$\alpha(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{\mathbf{z}:\mathbf{z}\in\mathcal{Y}_{0:t}, \mathbf{z}^p \leq_s \mathbf{z}} \exp\left(\sum_{t'=1}^{t-1} \sum_{i:f(\mathbf{x},\mathbf{z},t')=1} \lambda_i\right) - \sum_{\mathbf{z}:\mathbf{z}\in\mathcal{Y}_{0:t}, \exists(\mathbf{z}'\in\mathcal{P}_t, s(\mathbf{z}',\mathcal{P}_t)=\mathbf{z}^p)\mathbf{z}'\leq_s\mathbf{z}} \exp\left(\sum_{t'=1}^{t-1} \sum_{i:f(\mathbf{x},\mathbf{z},t')=1} \lambda_i\right) \tag{31}$$

これは，$\mathbf{z}^p$ を接尾辞として含む系列の位置 1 から $t-1$ までの前向き得点の合計から,「$\mathbf{z}^p$ を最長純接尾辞とする位置 t のパス」のいずれかを接尾辞として含む系列の位置 1 から $t-1$ までの前向き得点の合計を引いた差分である．$\alpha(\epsilon, t) = 0$ とする.

図 1 の例では，$t = 1$ の位置で X というラベル列を持つパスの α は 0 となっている．BX というパスが同位置にあるため，$\alpha(X, 1)$ は $t = 1$ で X を接尾辞として含む系列の前位置までの前向き得点の合計から，BX を接尾辞として含む系列の前位置までの前向き得点の合計を引いた差分になるが，$t = 1$ で X を接尾辞として含む系列はすべて BX を接尾辞として含む ($t = 0$ ではラベルは B のみであるため) ので，得点の差分は 0 になる.

別の例を挙げると，$t = 2$ の位置で X のラベル列を持つパスの α は 1.74 となっているが，YX というパスが同位置にあるため，これは $t = 2$ で X を接尾辞として含む系列の前位置までの前向き得点の合計から，YX を接尾辞として含む系列の前位置までの前向き得点の合計を引いた差分となり，つまり XX と ZX を接尾辞として含む系列の前位置までの前向き得点の合計ということになる.

また，あるパス $\mathbf{z}^p \in \mathcal{P}_t$ と位置 t について,「前向き合計得点」である $\gamma(\mathbf{z}^p, t)$ を次のように定義する.

$$\gamma(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{\mathbf{z}:\mathbf{z}\in\mathcal{Y}_{0:t}, \mathbf{z}^p \leq_s \mathbf{z}} \exp\left(\sum_{t'=1}^{t} \sum_{i:f(\mathbf{x},\mathbf{z},t')=1} \lambda_i\right) \tag{32}$$

これは，$\mathbf{z}^p$ を接尾辞として含む，位置 t までの系列の前向き得点を合計したものである．$\gamma(\epsilon, 0) = \gamma(l_{BOS}, 0) = 1$ とする.

例えば，$t = 2$ の位置で X というラベル列を持つパスの γ は 0.42 であるが，これは X を接尾辞として含む系列すべての前向き得点の合計を表す.

α と γ をこのように定義すると，次の等式が成り立つ．

$$\alpha(\mathbf{z}^p, t) = \gamma(\mathbf{z}^p_{1:|\mathbf{z}^p|-1}, t-1) - \sum_{\mathbf{z}:\mathbf{z}\in\mathcal{P}_t, s(\mathbf{z},\mathcal{P}_t)=\mathbf{z}^p} \gamma(\mathbf{z}_{1:|\mathbf{z}|-1}, t-1) \quad (33)$$

$$\gamma(\mathbf{z}^p, t) = \sum_{\mathbf{z}\in\mathcal{P}_t, \mathbf{z}^p\leq_s \mathbf{z}} \alpha(\mathbf{z}, t)\exp(W(\mathbf{z}, t)) \quad (34)$$

まず，式 (33) について説明する．$\gamma(\mathbf{z}^p_{1:|\mathbf{z}^p|-1}, t-1)$ は，$\mathbf{z}^p$ を接尾辞として含むパスの前位置までの合計得点である．これから，「$\mathcal{P}_t$ に関して $\mathbf{z}^p$ を最長純接尾辞とするようなパス」を接尾辞として含むようなパスの前位置までの合計得点を差し引くことによって，前向き差分得点である $\alpha(\mathbf{z}^p, t)$ を得ることができる．

次に，式 (34) について説明する．$\alpha(\mathbf{z}, t)$ は前位置までの前向き差分得点であるので，これに $\exp(W(\mathbf{z}, t))$ を乗じることによって現位置までの前向き差分得点となる．また，$\alpha(\mathbf{z}, t)$ は，それを $\mathcal{P}_t$ に関する最長純接尾辞として含む系列との差分得点であるため，$\mathcal{P}_t$ に関して $\mathbf{z}$ を接尾辞として含む系列の α に $\exp(W)$ を乗じたものを合計することによって，合計得点である $\gamma(\mathbf{z}, t)$ を得ることができる．

例えば，$t = 2$ の位置で YX というラベル列を持つパスの γ は 0.13 となっている．これは YX を接尾辞として含む系列すべての前向き得点の合計を表す．これは，そのパスの α である 0.83 に W である 0.15 をかけて求められている．また，同位置で X というラベル列を持つパスの γ は 0.42 である．これは，パス YX の γ である 0.13 に，パス X の α である 1.74 と，W である 0.17 をかけて求められる 0.29 を加えたものである．位置 2 においてパス X の α は，X で終わり YX で終わらないパスの位置 1 までのスコア (差分スコア) を保持しているため，それにパスの W をかけることで位置 2 までの差分スコアとなる．それに YX のパスのスコアを加えることで，X のパスの合計スコアとなる．

5.2.4 後ろ向き変数

あるパス $\mathbf{z}^p \in \mathcal{P}_t$ と位置 t について，「後ろ向き合計得点」である $\beta(\mathbf{z}^p, t)$ を次のように定義する．

$$\beta(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{\mathbf{z}\in\mathcal{Y}_{t+1:T+1}} \exp\left(\sum_{t'=t+1}^{T+1} \sum_{i:f_i(\mathbf{x},\mathbf{z}^p+\mathbf{z},t')} \lambda_i\right) \quad (35)$$

これは，位置 $t+1$ から $T+1$ までのすべての系列について，それが $\mathbf{z}^p$ を接頭辞として持つときの後ろ向き得点を合計したものである．任意の $\mathbf{z}$ について，$\beta(\mathbf{z}, T+1) = 1$ とする．なお，$f_i(\mathbf{x}, \mathbf{z}^p + \mathbf{z}, t')$ は，正確には $f_i(\mathbf{x}, \perp^{t+1-|\mathbf{z}^p|} + \mathbf{z}^p +$

$\mathbf{z}, t')$ であるが，以降このように略記する．

さらに，あるパス $\mathbf{z}^p \in \mathcal{P}_t$ と位置 $t \leq T$ について，$\delta(\mathbf{z}^p, t)$ を次のように定義する．

$$\delta(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{\mathbf{z} \in \mathcal{Y}_{t+1:T+1}} \left(\exp\left(\sum_{t'=t+1}^{T+1} \sum_{i: f_i(\mathbf{x}, \mathbf{z}^p + \mathbf{z}, t')=1} \lambda_i \right) - \exp\left(\sum_{t'=t+1}^{T+1} \sum_{i: f_i(\mathbf{x}, s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}, t')=1} \lambda_i \right) \right) \tag{36}$$

これは，位置 $t+1$ から $T+1$ までの系列について，それが $\mathbf{z}^p$ を接頭辞を持つとしたときの後ろ向き得点から，「$\mathbf{z}^p$ の最長純接尾辞」を接頭辞と持つとしたときの後ろ向き得点を引いた差分の合計である．$\delta(\epsilon, T+1) = 1$ とする．

このように定義すると，次の等式が成り立つ．

$$\begin{aligned} \delta(\mathbf{z}^p, t) &= \sum_{\mathbf{z} \in \mathcal{P}_{t+1}, \mathbf{z}_{1:|\mathbf{z}|-1} = \mathbf{z}^p} (\beta(\mathbf{z}, t+1) \exp(W(\mathbf{z}, t+1)) - \\ &\qquad \beta(s(\mathbf{z}, \mathcal{P}_{t+1}), t+1) \exp(W(s(\mathbf{z}, \mathcal{P}_{t+1}), t+1))) \qquad (37) \\ \beta(\mathbf{z}^p, t) &= \sum_{\mathbf{z} \in \mathcal{P}_t, \mathbf{z} \leq_s \mathbf{z}^p} \delta(\mathbf{z}, t) \qquad (38) \end{aligned}$$

式 (37) の導出はそれほど自明ではないため，詳しく述べる．まず，式 (36) を次のように変形する．

$$\sum_{\mathbf{z} \in \mathcal{Y}_{t+2:T+1}} \sum_{\mathbf{z}' \in \mathcal{Y}_{t+1:t+1}} \left(\exp\left(\sum_{t'=t+2}^{T+1} \sum_{i: f_i(\mathbf{x}, \mathbf{z}^p + \mathbf{z}' + \mathbf{z}, t')=1} \lambda_i \right) \exp\left(\sum_{i: f_i(\mathbf{x}, \mathbf{z}^p + \mathbf{z}', t+1)} \lambda_i \right) - \exp\left(\sum_{t'=t+2}^{T+1} \sum_{i: f_i(\mathbf{x}, s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}' + \mathbf{z}, t')=1} \lambda_i \right) \exp\left(\sum_{i: f_i(\mathbf{x}, s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}' + \mathbf{z}, t)} \lambda_i \right) \right) \tag{39}$$

ここで，式 (21) より，式 (39) は次のように書き直せる．

$$\sum_{\mathbf{z} \in \mathcal{Y}_{t+2:T+1}} \sum_{\mathbf{z}' \in \mathcal{Y}_{t+1:t+1}} \left(\exp\left(\sum_{t'=t+2}^{T+1} \sum_{i: f_i(\mathbf{x}, S(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) + \mathbf{z}, t')=1} \lambda_i \right) \exp\left(\sum_{i: f_i(\mathbf{x}, S(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}), t+1)} \lambda_i \right) - \exp\left(\sum_{t'=t+2}^{T+1} \sum_{i: f_i(\mathbf{x}, S(s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}', \mathcal{P}_{t+1}) + \mathbf{z}, t')=1} \lambda_i \right) \exp\left(\sum_{i: f_i(\mathbf{x}, S(s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}', \mathcal{P}_{t+1}), t)} \lambda_i \right) \right) \tag{40}$$

ここで，次が成り立つ．

$$s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) \leq_s s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}' \tag{41}$$

そうでないと仮定する．式 (20) の最長純接尾辞の定義より，$s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) <_s \mathbf{z}^p + \mathbf{z}', s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}' <_s \mathbf{z}^p + \mathbf{z}'$ であるため，式 (19) より，$s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}' <_s s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1})$ となる．$\mathbf{z}'' = s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1})$ とおくと，$\mathbf{z}'' \in \mathcal{P}_{t+1}$ であり，$\mathbf{z}'' \neq \epsilon$ であるため，式 (26) より $\mathbf{z}''_{1:|\mathbf{z}''|-1} \in \mathcal{P}_t$ となり，また式 (18) より，$s(\mathbf{z}^p, \mathcal{P}_t) <_s \mathbf{z}''_{1:|\mathbf{z}''|-1} <_s \mathbf{z}^p$ となるが，これは式 (20) の最長純接尾辞の定義に矛盾する．

また，式 (20) の最長純接尾辞の定義より，位置 $t+1$ に $s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) <_s \mathbf{z}'' <_s \mathbf{z}^p + \mathbf{z}'$ であるような $\mathbf{z}''$ は存在しない．

これらにより，次が成り立つ．

$$S(s(\mathbf{z}^p, \mathcal{P}_t) + \mathbf{z}', \mathcal{P}_{t+1}) = s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) \tag{42}$$

さらに，$\mathbf{z}^p + \mathbf{z}' \notin \mathcal{P}_{t+1}$ であれば，式 (21) の最長接尾辞の定義より $S(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) \neq \mathbf{z}^p + \mathbf{z}'$ となるため，次が成り立つ．

$$S(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) = s(\mathbf{z}^p + \mathbf{z}', \mathcal{P}_{t+1}) \tag{43}$$

よって，式 (40) で，$\mathbf{z}^p + \mathbf{z}' \notin \mathcal{P}_{t+1}$ のものについては，前項と後項が等しくなるため，省略できる．

これらより，式 (40) は次のように書き直せる．

$$\sum_{\mathbf{z} \in \mathcal{Y}_{t+2:T+1}} \sum_{\mathbf{z}' \in \mathcal{P}_{t+1}, \mathbf{z}'_{|\mathbf{z}'|-1} = \mathbf{z}^p} \left(\exp\left(\sum_{t'=t+2}^{T+1} \sum_{i: f_i(\mathbf{x}, \mathbf{z}'+\mathbf{z}, t')=1} \lambda_i \right) \exp\left(\sum_{i: f_i(\mathbf{x}, \mathbf{z}', t+1)} \lambda_i \right) - \exp\left(\sum_{t'=t+2}^{T+1} \sum_{i: f_i(\mathbf{x}, s(\mathbf{z}', \mathcal{P}_{t+1})+\mathbf{z}, t')=1} \lambda_i \right) \exp\left(\sum_{i: f_i(\mathbf{x}, s(\mathbf{z}', \mathcal{P}_{t+1})+\mathbf{z}, \mathcal{P}_{t+1}), t)} \lambda_i \right) \right) \tag{44}$$

ここで，式 (35) の β の定義と，式 (29) の W の定義より，式 (37) が導ける．

図 1 の例では，位置 $t=3$ でラベル列 X を持つパスの δ は 0.08 となっている．これは，後列で XE というラベル列を持つパスの β にパスの指数重みをかけた 0.48 の，E というラベル列を持つパスの $\beta(1.0)$ に指数重みをかけた 0.40 からの差分となっている．

5.2.5 前向き-後ろ向き変数

あるパス $\mathbf{z}^p \in \mathcal{P}_t$ と位置 t について，$\theta(\mathbf{z}^p, t), \sigma(\mathbf{z}^p, t)$ を次のように定義する．

$$\theta(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \alpha(\mathbf{z}^p, t) \exp(W(\mathbf{z}^p, t)) \beta(\mathbf{z}^p, t) \tag{45}$$

$$\sigma(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \sum_{\mathbf{z}: \mathbf{z} \in \mathcal{P}_t, \mathbf{z} \leq_s \mathbf{z}^p} \theta(\mathbf{z}, t) \tag{46}$$

$\theta(\mathbf{z}^p, t)$ は，位置 t までの部分系列が $\mathbf{z}^p$ を含み，かつ「$\mathbf{z}^p$ を接尾辞として含む位置 t のパス」のいずれも接尾辞として含まないような系列の $t = 0$ から $T + 1$ までの得点を合計したものである．図 1 の例では，$t = 2$ で X というラベル列を持つパスの θ は 0.46 である (θ は図の中では示していない)．これは，位置 $t = 2$ で X を接尾辞として含み，YX を接尾辞として含まない (つまり，$t = 1, 2$ でのラベルが XX または ZX である) 系列の正規化前の差分得点である．

$\sigma(\mathbf{z}^p, t)$ は，位置 t までの部分系列が $\mathbf{z}^p$ を含むような系列の $t = 0$ から $T + 1$ までの得点を合計したものである．図 1 で，$t = 2$ で X というラベル列を持つパスの σ は 0.66 であるが，これは位置 $t = 2$ で X を接尾辞として含む系列の正規化前の合計得点である．$\sigma(\mathbf{z}^p, t) = \sigma(s(\mathbf{z}^p, \mathcal{P}_t), t) + \theta(\mathbf{z}^p, t)$ となる．

正規化係数 $Z(\mathbf{x})$ は，γ を使って次のように表せる．

$$Z(\mathbf{x}) = \gamma(\epsilon, T + 1) \tag{47}$$

現在のモデルパラメータによる，ラベル列 $\mathbf{y}$ が位置 t で終わる場所でパス $\mathbf{z}^p \in \mathcal{P}_t$ を含む確率は次のように表せる．

$$P(\mathbf{z}^p \leq_s \mathbf{y}_{1:t} \mid \mathbf{x}) = \sigma(\mathbf{z}^p, t)/Z(\mathbf{x}) \tag{48}$$

これが，現在のモデルパラメータによる，それぞれのパスに関連付けられた素性関数の確率となるため，それを足し合わせることにより，素性関数の期待値が求められる．

5.2.6 各変数を求める手続き

ここでは，前節までに定義した各変数を効率的に求める手続きを示す．

まず，$t = 1, \cdots, T + 1$ について，$\mathcal{P}_t$ を求める．また，位置 t でラベル列 $\mathbf{z}$ を持つパスに対応する素性関数の集合を $\mathcal{F}_{\mathbf{z},t}$ とし，それらを列挙する．この手続きは，Algorithm 5 で表せる．

この手続きの中では明示的に記述していないが，この過程で，位置 t でのあるパス $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ に対して，それに対応する位置 $t-1$ のパス $\mathbf{z}_{1:|\mathbf{z}|-1} \in \mathcal{P}_{t-1}$(以下，「接頭辞パス」と呼ぶ) を容易に関連付けることができる．また，パス集合を，パスのラベル列を逆に並べたものをキーとするトライによって管理することによって，位置 t でのある $\mathbf{z}$ に対して，それに対応する最長純接尾辞 $s(\mathbf{z}, P_t)$ を関連付けることができる．また，トライを深さ優先で探索することにより，$\mathbf{z}^1 <_s \mathbf{z}^2$ のときに $\mathbf{z}^1$ が $\mathbf{z}^2$ よりも前になるような順序で並べることができる．この順序

Algorithm 5 Make paths

for $t = 1$ **to** $T + 1$ **do**
  $\mathcal{F} \leftarrow \{f_k \mid b_k(\mathbf{x}, t) = 1\}$
  for all $f_i \in \mathcal{F}$ **do**
    $\mathcal{F}_{\mathbf{z}^i,t} \leftarrow \mathcal{F}_{\mathbf{z}^i,t} \cup \{f_i\}$
    for $j = 0$ **to** $|\mathbf{z}^i|$ **do**
      $\mathcal{P}_{t-j} \leftarrow \mathcal{P}_{t-j} \cup \{\mathbf{z}^i_{1:|\mathbf{z}|-j}\}$
    end for
  end for
end for

を「昇順」(ascending order) と定義し，その逆を「降順」(descending order) と定義する．これらの情報 (パスに対応する素性関数の集合，接頭辞パス，最長純接尾辞パス，パスの並び順) はイテレーションにかかわらず不変であるため，記録しておくことにより重複計算を避けられる．

Algorithm 6 は，イテレーションごとに実行するものである．任意の $w(\mathbf{z}, t)$，また他のすべての数値変数は 0 で初期化されているものとする．

この手順によって，式 (33) が満たされることを示す．他のものの接尾辞とならない要素 $\mathbf{z}^i$ に対しては，$\alpha(\mathbf{z}^i, t) = \gamma(\mathbf{z}_{1:|\mathbf{z}|-\mathbf{1}}, t)$ となり，式 (33) を満たす．他のものの接尾辞となる要素 $\mathbf{z}^i$ に対しては，それを接尾辞とするものがすべて正しく計算されていれば，2 行目の減算によって，式 (33) が満たされる．降順に処理するため，あるラベル列を接尾辞とする他のラベル列は，接尾辞よりも先に計算されている．よって，帰納法によりすべての $\mathbf{z} \in \mathcal{P}_t$ に対して $\alpha(\mathbf{z}, t)$ が正しい値となることが示せる．

その他，類似の証明を省略する．

この手続きの計算量は，1 イテレーション・1 系列ごとに

$$O\left(\sum_{t=1}^{T+1} \sum_{\mathbf{z}^p : \mathbf{z}^p \in \mathcal{P}_t} |\mathcal{F}_{\mathbf{z}^p,t}| + \sum_{t=1}^{T+1} |\mathcal{P}_t|\right) \tag{49}$$

となる．

この手続きの中で，γ, δ は一時的にしか使用しないため，それぞれ $t-1$ よりも前のもの・$t+1$ よりも後のものは記憶する必要がない．また，すべての素性関数について exp をとったものを記憶することにより重複計算を避けることが

Algorithm 6 Sum-difference

for $t = 1$ **to** $T + 1$ **do**
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in ascending order **do**
    for all $f_i \in \mathcal{F}_{\mathbf{z},t}$ **do**
      $w(\mathbf{z}, t) \leftarrow w(\mathbf{z}, t) + \lambda_i$
    end for
    $W(\mathbf{z}, t) \leftarrow W(s(\mathbf{z}, \mathcal{P}_t), t) + w(\mathbf{z}, t)$
  end for
end for
$\gamma(\epsilon, 0) \leftarrow 1, \gamma(l_{BOS}, 0) \leftarrow 1$
for $t = 1$ **to** $T + 1$ **do**
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in descending order **do**
    $\alpha(s(\mathbf{z}, \mathcal{P}_t), t) \leftarrow \alpha(s(\mathbf{z}, \mathcal{P}_t), t) - \gamma(\mathbf{z}_{1:|\mathbf{z}|-1}, t - 1)$
    $\alpha(\mathbf{z}, t) \leftarrow \alpha(\mathbf{z}, t) + \gamma(\mathbf{z}_{1:|\mathbf{z}|-\mathbf{1}}, t - 1)$
  end for
  $\alpha(\epsilon, t) \leftarrow 0$
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in descending order **do**
    $\gamma(\mathbf{z}, t) \leftarrow \gamma(\mathbf{z}, t) + \alpha(\mathbf{z}, t) \exp(W(\mathbf{z}, t))$
    $\gamma(s(\mathbf{z}, \mathcal{P}_t), t) \leftarrow \gamma(s(\mathbf{z}, \mathcal{P}_t), t) + \gamma(\mathbf{z}, t)$
  end for
end for
$Z(\mathbf{x}) \leftarrow \gamma(\epsilon, T + 1)$
$\delta(\epsilon, T + 1) \leftarrow 1$
for $t = T + 1$ **downto** 1 **do**
  $\beta(\epsilon, t) \leftarrow \delta(\epsilon, t)$
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in ascending order **do**
    $\beta(\mathbf{z}, t) \leftarrow \beta(s(\mathbf{z}, \mathcal{P}_t), t) + \delta(\mathbf{z}, t)$
  end for
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ **do**
    $\delta(\mathbf{z}_{1:|\mathbf{z}|-1}, t-1) \leftarrow \delta(\mathbf{z}_{1:|\mathbf{z}|-1}, t-1) + \beta(\mathbf{z}, t) \exp(W(\mathbf{z}, t)) - \beta(s(\mathbf{z}, \mathcal{P}_t), t) \exp(W(s(\mathbf{z}, \mathcal{P}_t)))$
  end for
end for
for $t = 1$ **to** $T + 1$ **do**
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in descending order **do**
    $\theta(\mathbf{z}^p, t) \leftarrow \alpha(\mathbf{z}^p, t) \exp(W(\mathbf{z}^p, t)) \beta(\mathbf{z}^p, t)$
    $\sigma(\mathbf{z}, t) \leftarrow \sigma(\mathbf{z}, t) + \theta(\mathbf{z}, t)$
    $\sigma(s(\mathbf{z}, \mathcal{P}_t), t) \leftarrow \sigma(s(\mathbf{z}, \mathcal{P}_t), t) + \sigma(\mathbf{z}, t)$
    for all $f_i \in \mathcal{F}_{\mathbf{z},t}$ **do**
      $E[f_i] \leftarrow E[f_i] + \sigma(\mathbf{z}, t)/Z(\mathbf{x})$
    end for
  end for
end for

できる．さらに，複数のループをまとめることも可能である (Algorithm 6 では変数をそれぞれ別のループで求めているが，例えば α, β の計算は一つのループで十分である)．

なお，この手続きにおいては，各訓練データに対して各位置でのパスを記録するため，計算量に比例した記憶領域が必要となる．しかし，その領域に対してランダムアクセスする必要はなく，ハードディスクドライブといった比較的安価な媒体に記録しても速度に対する影響は大きくない．

5.3 デコード

ここでは，高次 Linear-Chain CRF において，学習されたパラメータを用いて，観測列から最適なラベル列を求める手順について記述する．

まず，求める $\mathbf{y}^*$ は次のように表すことができる．

$$\begin{aligned}\mathbf{y}^* &= \arg\max_{\mathbf{y}} \frac{1}{Z(\mathbf{x})} \exp\left(\sum_t \sum_i \lambda_i f_i(\mathbf{x}, \mathbf{y}, t)\right) \\ &= \arg\max_{\mathbf{y}} \exp\left(\sum_t \sum_i \lambda_i f_i(\mathbf{x}, \mathbf{y}, t)\right) \end{aligned} \tag{50}$$

これを求めるために，前向きに解く方法と，後ろ向きに解く方法の二通りを紹介する．計算量は，前向きの方法が $O\left(\sum_{t=1}^{T+1} \sum_{i:b(\mathbf{x},t)=1} 1 + L\left(\sum_{t=1}^{T+1} \sum_{\mathbf{z}^p:\mathbf{z}^p \in \mathcal{P}_t} 1\right)\right)$，後ろ向きの方法が $O\left(\sum_{t=1}^{T+1} \sum_{i:b(\mathbf{x},t)=1} 1 + \log L\left(\sum_{t=1}^{T+1} \sum_{\mathbf{z}^p:\mathbf{z}^p \in \mathcal{P}_t} 1\right)\right)$ である．後ろ向きの方法については，これが計算量の下界であるかは確認していない．

前向きの方法は計算量が大きく，後ろ向きの方法は手順が煩雑であるという欠点がある．

なお，前向き・後ろ向き共に，それぞれのパスについて W を求めておく必要がある．この手順は前節の合計・差分アルゴリズムと共通であるが，その部分を Algorithm 7 として再掲する．

5.3.1 デコード・前向きの方法

パス $\mathbf{z}^p \in \mathcal{P}_t$ について，$p(\mathbf{z}^p, t), q(\mathbf{z}^p, t)$ を次のように定義する．

$$\begin{aligned} p(\mathbf{z}^p, t) &\stackrel{\text{def}}{=} \max_{\mathbf{z}:\mathbf{z}\in\mathcal{P}_{t-1}, \mathbf{z}^p_{1:|\mathbf{z}^p|-1} \leq_s \mathbf{z}, \forall(\mathbf{z}':\mathbf{z}'\in\mathcal{P}_{t-1}, \mathbf{z}^p \leq_s \mathbf{z}')\mathbf{z}' \not\leq_s \mathbf{z}} \\ &\quad (p(\mathbf{z}, t-1)) \cdot \exp(W(\mathbf{z}^p, t)) \end{aligned} \tag{51}$$

$$\begin{aligned} q(\mathbf{z}^p, t) &\stackrel{\text{def}}{=} \arg \max_{\mathbf{z}:\mathbf{z}\in\mathcal{P}_{t-1}, \mathbf{z}^p_{1:|\mathbf{z}^p|-1} \leq_s \mathbf{z}, \forall(\mathbf{z}':\mathbf{z}'\in\mathcal{P}_{t-1}, \mathbf{z}^p \leq_s \mathbf{z})\mathbf{z}' \not\leq_s \mathbf{z}} \\ &\quad (p(\mathbf{z}, t-1)) \end{aligned} \tag{52}$$

Algorithm 7 Calculate weights

for $t = 1$ **to** $T + 1$ **do**
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in ascending order **do**
    for all $f_i \in \mathcal{F}_{\mathbf{z},t}$ **do**
      $w(\mathbf{z}, t) \leftarrow w(\mathbf{z}, t) + \lambda_i$
    end for
    $W(\mathbf{z}, t) \leftarrow W(s(\mathbf{z}, \mathcal{P}_t), t) + w(\mathbf{z}, t)$
  end for
end for

$p(\mathbf{z}^p, t)$ は，$\mathbf{z}^p$ を接尾辞として含み，かつ $\mathbf{z}^p$ を接尾辞として含む位置 t のパスのいずれも接尾辞として含まないような位置 $t-1$ のパスの中での最大スコアに $\mathbf{z}^p$ の指数重みをかけたものを，$q(\mathbf{z}^p, t)$ はそのような $p(\mathbf{z}^p, t)$ を与える位置 $t-1$ のパスを記録する．

このように定義すると，Algorithm 8 のようにして p, q を求めることができる．アルゴリズム内では，補助的に r, u という変数を用いている．なお，このアルゴリズムの前に Algorithm 5 によりそれぞれの位置におけるパスを求め，Algorithm 7 によりパスの重みを求めておく必要がある．

その後，Algorithm 9 により，後ろから q をたどり，最適なラベル列 $\mathbf{y}^*$ を求める．

この方法によるデコードの利点としては，次に述べる後ろ向きによる方法に比べて単純であり，また合計・差分アルゴリズムと同じパスの並べ方を使うため，その部分に共通した手順を使用できるということがある．欠点としては，計算量が多いということがある．Algorithm 8 の計算量は，合計・差分アルゴリズムに対して，素性関数の列挙に関する部分を除けば，$O(L)$ 倍となる．

Algorithm 8 Decode-forward(1)

$p(\epsilon, 0) \leftarrow 1, p(l_{BOS}, 0) \leftarrow 1$
for $t = 1$ **to** $T + 1$ **do**
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in descending order **do**
    if this is the first iteration or the last label in $\mathbf{z}$ has changed **then**
      $\mathbf{z}' \leftarrow$ the first $\mathbf{z}' \in \mathcal{P}_{t-1}$ in descending order
      for all $\mathbf{z}'' \in \mathcal{P}_{t-1}$ **do**
        $r(\mathbf{z}'', t-1) \leftarrow p(\mathbf{z}'', t-1)$
        $u(\mathbf{z}'', t-1) \leftarrow \mathbf{z}''$
      end for
    end if
    while $\mathbf{z}' \neq \mathbf{z}_{1:|\mathbf{z}|-1}$ **do**
      if $r(\mathbf{z}', t-1) > r(s(\mathbf{z}', t-1), t-1)$ **then**
        $r(s(\mathbf{z}', t-1), t-1) \leftarrow r(\mathbf{z}', t-1)$
        $u(s(\mathbf{z}', t-1), t-1) \leftarrow u(\mathbf{z}', t-1)$
      end if
      $\mathbf{z}' \leftarrow$ next $\mathbf{z}' \in \mathcal{P}_{t-1}$ in descending order
    end while
    $p(\mathbf{z}, t) \leftarrow r(\mathbf{z}', t) \exp(W(\mathbf{z}, t))$
    $q(\mathbf{z}, t) \leftarrow u(\mathbf{z}', t)$
  end for
end for

Algorithm 9 Decode-forward(2)

$\mathbf{z} \leftarrow \arg\max_{\mathbf{z} \in \mathcal{P}_{T+1}} p(\mathbf{z}, T+1)$
for $t = T + 1$ **to** 1 **do**
  $\mathbf{y}_t^* \leftarrow \mathbf{z}_{|\mathbf{z}|}$
  $\mathbf{z} \leftarrow q(\mathbf{z}, t)$
end for
$\mathbf{y}_0^* \leftarrow l_{BOS}$

5.3.2 デコード・後ろ向きの方法

後ろ向きの方法では，パス $\mathbf{z}^p \in \mathcal{P}_t$ について，$p(\mathbf{z}^p, t), q(\mathbf{z}^p, t)$ を次のように定義する．

$$p(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \max_{\mathbf{z}:\mathbf{z}\in\mathcal{P}_{t+1}, \mathbf{z}_{1:|\mathbf{z}|-1} \leq_s \mathbf{z}^p, \forall(\mathbf{z}':\mathbf{z}'\in\mathcal{P}_{t+1}, \mathbf{z}\leq_s\mathbf{z}')\mathbf{z}'_{1:|\mathbf{z}'|-1} \not\leq_s \mathbf{z}^p} (p(\mathbf{z}, t+1)) \cdot \exp(W(\mathbf{z}^p, t)) \tag{53}$$

$$q(\mathbf{z}^p, t) \stackrel{\text{def}}{=} \arg \max_{\mathbf{z}:\mathbf{z}\in\mathcal{P}_{t+1}, \mathbf{z}_{1:|\mathbf{z}|-1} \leq_s \mathbf{z}^p, \forall(\mathbf{z}':\mathbf{z}'\in\mathcal{P}_{t+1}, \mathbf{z}\leq_s\mathbf{z}')\mathbf{z}'_{1:|\mathbf{z}'|-1} \not\leq_s \mathbf{z}^p} (p(\mathbf{z}, t+1)) \tag{54}$$

ここでの $p(\mathbf{z}^p, t)$ は，$\mathbf{z}^p$ を接頭辞として持ち，かつ $\mathbf{z}^p$ を接尾辞として含む位置 t のパスのいずれも接頭辞として持たないような位置 $t+1$ のパスの中での最大スコアに $\mathbf{z}^p$ の指数重みをかけたものを，$q(\mathbf{z}^p, t)$ はそのような $p(\mathbf{z}^p, t)$ を与える位置 $t+1$ のパスを記録する．

このように定義すると，Algorithm 10 のようにして p, q を求めることができる．アルゴリズム内では，補助的に r, u, v, l, H という変数を用いている．

H は最大ヒープであり，Heap-insert, Heap-delete, Heap-max という手続きにより，挿入・削除・最大値の調査という操作ができるものとする．Heap-insert はヒープ以外にラベルと数値を引数をとり，それらを関連付けて記録する．Heap-delete はヒープ以外にラベルを引数にとり，そのラベルと関連付けられたノードを削除する．Heap-max はヒープのノード中で数値が最大であるもののラベルを返す．前二者の計算量は $O(\log L)$ で，Heap-max は $O(1)$ である．

また，位置 t での「パス接頭辞集合」として，$\mathcal{Q}_t$ を次のように定義する．

$$\mathcal{Q}_t = \{\mathbf{z}_{1:|\mathbf{z}|-1} \mid \mathbf{z} \in \mathcal{P}_t\} \tag{55}$$

さらに，任意のラベル列 $\mathbf{z}$ に対して，位置 t のラベルを z_t とする．また，ラベル列 $\mathbf{z}^1$ の末尾にラベル z を追加したラベル列が $\mathbf{z}^2$ であるとき，それを $\mathbf{z}^2 = \mathbf{z}^1 + z$ と表す．

なお，このアルゴリズムの前に Algorithm 5 によりそれぞれの位置におけるパスを求め，Algorithm 7 によりパスの重みを求めておく必要がある．

その後，Algorithm 11 により，前から q をたどり，最適なラベル列 $\mathbf{y}^*$ を求める．

Algorithm 10 Decode-backward(1)

$v(\epsilon, T+1) \leftarrow 1$
for $t = T+1$ **downto** 1 **do**
  for all $\mathbf{z} \neq \epsilon \in \mathcal{P}_t$ in ascending order **do**
    if $v(\mathbf{z}, t) = 0$ **then**
      $v(\mathbf{z}, t) = v(s(\mathbf{z}, \mathcal{P}_t), t)$
      $q(\mathbf{z}, t) = q(s(\mathbf{z}, \mathcal{P}_t), t)$
    end if
    $p(\mathbf{z}, t) = v(\mathbf{z}, t) \exp(W(\mathbf{z}, t))$
  end for
  $H \leftarrow$ empty heap
  for all $\mathbf{z} : \mathbf{z} \in \mathcal{P}_t, |\mathbf{z}| = 1$ **do**
    HEAP-INSERT$(H, z_1, t))$
    $u(z_1) \leftarrow p(\mathbf{z}, t)$
  end for
  $\mathbf{z}' \leftarrow \epsilon$
  for all $\mathbf{z} \neq \epsilon \in \mathcal{Q}_t$ in ascending order **do**
    while $\mathbf{z}' \neq s(\mathbf{z}, \mathcal{Q}_t)$ **do**
      for all $\mathbf{z}'' : \mathbf{z}'' \in \mathcal{P}_t, \mathbf{z}''_{1:|\mathbf{z}''|-1} = \mathbf{z}'$ **do**
        HEAP-DELETE$(H, z''_{|\mathbf{z}''|-1})$
        HEAP-INSERT$(H, z''_{|\mathbf{z}''|-1}, r(\mathbf{z}''))$
      end for
      $\mathbf{z}' \leftarrow s(\mathbf{z}', \mathcal{Q}_t)$
    end while
    for all $\mathbf{z}'' : \mathbf{z}'' \in \mathcal{P}_t, \mathbf{z}''_{1:|\mathbf{z}''|-1} = \mathbf{z}'$ **do**
      $r(\mathbf{z}'') \leftarrow u(z''_{|\mathbf{z}''|-1})$
      $u(z''_{|\mathbf{z}''|-1}) \leftarrow p(\mathbf{z}'', t)$
      HEAP-DELETE$(H, z''_{|\mathbf{z}''|-1})$
      HEAP-INSERT$(H, z''_{|\mathbf{z}''|-1}, p(\mathbf{z}'', t))$
    end for
    $l \leftarrow$ HEAP-MAX(H)
    $q(\mathbf{z}^p, t) \leftarrow s(\mathbf{z}^p + l, \mathcal{P}_{t+1})$
    $v(\mathbf{z}^p, t) \leftarrow u(l)$
    $\mathbf{z}' \leftarrow \mathbf{z}$
  end for
end for

Algorithm 11 Decode-backward(2)

if $v(l_{BOS}, 0) = 0$ **then**
  $q(l_{BOS}, 0) \leftarrow q(\epsilon, 0)$
end if
$\mathbf{z} \leftarrow l_{BOS}$
for $t = 0$ **to** T **do**
  $\mathbf{y}^*_t \leftarrow \mathbf{z}_{|\mathbf{z}|}$
  $\mathbf{z} \leftarrow q(\mathbf{z}, t)$
end for
$\mathbf{y}^*_{T+1} \leftarrow l_{EOS}$

このアルゴリズムの利点は，理論的な計算量が低いことである．前向きの方法では計算量が合計・差分アルゴリズムに対して，素性関数の列挙に関する部分を除いて $O(L)$ 倍になるのに対し，後ろ向きの方法では $O(\log L)$ 倍に抑えられる．ただし，新たに $\mathcal{Q}_t$ という「パス接頭辞集合」を構築する必要がある他，最大ヒープの利用等，手順が煩雑であり，実装する上での負担が大きい．本研究での実験においては，前向きの方法を用いている．

第6章　実験

CRFSuite[6] に本研究の計算法を組み込んだプログラムを使用し，英語品詞タグ付けの実験を行った．Penn Treebank 3.0 の Wall Street Journal 部分の 0-18 を訓練データ，22-24 をテストデータとして使用した．訓練データは 38,219 文 912,344 トークン，テストデータは 5,462 文 129,654 トークンである．

使用した素性は，英語の品詞タグ付けにおいて標準的な素性セット (Stanford Tagger[3] で使われているものからコーパス固有の素性を除いたもの) から，計算量が大きくなることを避けるためにラベルのみが 3 つ以上連続するものを除き，ラベルと観測を組み合わせた素性を追加した．素性関数は，訓練データセットに出現するラベル遷移のみについて設定している．ベースラインとして 1 次マルコフ Linear-Chain CRF を使用し，本研究のモデルでは最大次数を $2, 3, 4$ として素性関数を設定した実験を行った．それぞれの実験に使用した素性を表 1 に示す．

実験は，2.9GHz CPU，128GiB メモリの環境で行った．結果は表 2 の通りである．「最大 2 次」の素性セットを使用することによる精度の向上は認められたが，最大次数を 3 次以上にすることによる精度の向上はほぼ認められなかった．また，最大次数を高くすることによる更新時間の変化が大きくないことも示している．

また，副次的な効果として，本手法では exp や log の演算を最小限度に抑えられ，それによりより精度の高い計算が行える．そのため，CRFSuite[6] に比べてより多くのパラメータ更新を行うことができ，それによる精度の向上が達成できた．

Stanford Tagger[3] では，比較的少ない素性数で高い精度を実現している．その背景には，固有名詞認識モジュールを含むこともあるが，主な要因として MEMM[1] を基礎としたモデルであることによる計算量の低さと，それによる前後のラベル情報の利用しやすさが挙げられる．Stanford Tagger においては，3 つ以上の連続したタグといった素性を用いているが，本研究のモデルにおいては，そのような素性を使用すると一つの位置で 1 となる素性関数が急激に増加するため，利用することが難しい．

モデル	素性関数
ベースライン (1 次マルコフ)	y_i
	y_i, x_i
	y_i, x_{i-1}
	y_i, x_{i+1}
	y_i, x_{i-1}, x_i
	y_i, x_i, x_{i+1}
	y_i, x_{i-2}, x_{i-1}
	$y_i, x_{i-2}, x_{i-1}, x_i$
	$y_i, x_{i-3}, x_{i-2}, x_{i-1}$
	y_i, x_{i+1} の接尾辞 (10 文字まで)
	y_i, x_{i+1} の接頭辞 (10 文字まで)
	y_i, x_{i+1} がハイフンを含むか
	y_i, x_{i+1} が数字を含むか
	y_i, x_{i+1} が大文字を含むか
	y_i, y_{i-1}
最大 2 次	y_i, y_{i-1}, x_i
	y_i, y_{i-1}, x_{i-1}
	$y_i, y_{i-1}, y_{i-2}, x_{i-1}$
最大 3 次	$y_i, y_{i-1}, y_{i-2}, x_i, x_{i-1}$
	$y_i, y_{i-1}, y_{i-2}, x_{i-1}, x_{i-2}$
	$y_i, y_{i-1}, y_{i-2}, y_{i-3}, x_{i-1}, x_{i-2}$
最大 4 次	$y_i, y_{i-1}, y_{i-2}, y_{i-3}, x_i, x_{i-1}, x_{i-2}$
	$y_i, y_{i-1}, y_{i-2}, y_{i-3}, x_{i-1}, x_{i-2}, x_{i-3}$
	$y_i, y_{i-1}, y_{i-2}, y_{i-3}, y_{i-4}, x_{i-1}, x_{i-2}, x_{i-3}$

表 1: 実験に使用した素性

素性関数セット	素性数	平均更新時間 (秒)	更新回数	精度
baseline (1 次 CRF)	3113069	26	202	96.99%
1 次 CRF(提案手法)	3113069	75	1408	97.12%
最大 2 次	3618876	87	1277	97.18%
最大 3 次	5163811	87	1313	97.19%
最大 4 次	7329406	90	1154	97.19%
Stanford Tagger[3]	460552	-	-	97.24%

表 2: WSJ コーパスに対する POS-tagging の精度

第7章　考察

本研究の計算法により，Linear-Chain CRF において高次素性を使用するときの計算量を引き下げることができた．高次素性を使うことの有用性は Ye ら [4] の研究でも示されており，計算量を下げたことにより，Linear-Chain CRF の適用範囲が広がる可能性を示唆している．

実験では，Stanford Tagger[3] には及ばないものの，一次 Linear-Chain CRF に比較すると，精度の向上が実現できている．また，計算量が最高次数によらないことも示している．

本研究では，Linear-Chain CRF において高次素性を使用することが英語の POS-tagging 精度向上につながることを示した．今後は，日本語形態素解析等への応用を考えたい．

本研究で使用したプログラムのソースコード等は `http://vocrf.net/` で公開している．

謝辞

本研究を進めるにあたり，自由に研究を進めることを許してくださった黒橋禎夫教授に心より御礼申し上げます．

日々の研究生活を送るにあたり，様々な面で精神的な支えとなってくれた同期の黒川勇輝君，大友謙一君，渋田和宏君，望月道章君に深く感謝します．中でも，黒川勇輝君の存在なしでは，研究室でやっていくことができたかどうかわかりません．本当に有り難く思っています．

また，研究を進めるにあたり，エアコンとホワイトボードのあるミーティングルームを自由に使わせていただいたことに感謝します．この環境のおかげで，最大エントロピーモデルや Linear-Chain CRF を理解し，ひいては本研究のアイディアを練ることができました．

本研究について話を聞いていただき，アドバイスをいただいた河原大輔准教授，森信介准教授，柴田知秀助教，笹野遼平特定研究員 (当時)，博士課程 3 年の村脇有吾氏に深く感謝します．

最後に，本研究を進めるにあたって様々な協力をしてくださいました黒橋研究室の皆様に感謝致します．

参考文献

[1] McCallum, A., Freitag, D. and Pereira, F. C. N.: Maximum Entropy Markov Models for Information Extraction and Segmentation, *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp. 591–598 (2000).

[2] LAFFERTY, J.: Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data, *Proceedings of ICML, 2001* (2001).

[3] Toutanova, K., Klein, D., Manning, C. D. and Singer, Y.: Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network, *In Proceedings of HLT-NAACL 2003*, pp. 252–259 (2003).

[4] Ye, N., Lee, W. S., Chieu, H. L. and Wu, D.: Conditional Random Fields with High-Order Features for Sequence Labeling, *Advances in Neural Information Processing Systems 22* (Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I. and Culotta, A.(eds.)), pp. 2196–2204 (2009).

[5] Liu, D. and Nocedal, J.: On the Limited Memory Method for Large Scale Optimization, *Mathematical Programming B*, Vol. 45(3), pp. 503–528 (1989).

[6] Okazaki, N.: CRFsuite: a fast implementation of Conditional Random Fields (CRFs) (2007).

[7] Qian, X., Jiang, X., Zhang, Q., Huang, X. and Wu, L.: Sparse higher order conditional random fields for improved sequence labeling, *ICML*, p. 107 (2009).