Sequential Tagging メモ

浅原 正幸
奈良先端科学技術大学院大学 情報科学研究科
masayu-a@is.naist.jp

1 はじめに

sequential tagging (系列タギング) とは、ある観測された sequence (系列) $X = X_1, X_2, \ldots, X_n$ に対し、隠れ変数 (hidden variable) 列 $Y = Y_1, Y_2, \ldots, Y_n$ を付与する技術の総称である。自然言語処理の分野では X_i が単語であり、X が単語列であることが多い。図 1 に sequential tagging で解くことができるタスクの例を示す。例えば、X を単語列、Y を品詞とすると品詞タグ付けを行うことができる。また、chunk tag (範囲指定するようなタグ／ B を範囲の開始位置、I を範囲の内側、O を範囲の外側、S を単独で範囲となる位置) を導入することにより、系列中に出現する chunk (かたまり) を抽出することができる。内部に名詞句を含まない名詞句の範囲を chunk tag で同定することにより、基本名詞句同定を行うことができる。固有表現の範囲をチャンクタグで同定することにより、固有表現抽出を行うことができる。

確率モデルで解く場合、訓練時には $P(Y|X)$ をモデル化し、テスト時には新たに入力される X に対して $Y = arg\max_Y P(Y|X)$ を解くことによってタグ列を推定する。$P(Y|X)$ のモデル化には様々な手法がある。次の章では古くから用いられている Hidden Markov Model (隠れマルコフモデル) について解説する。

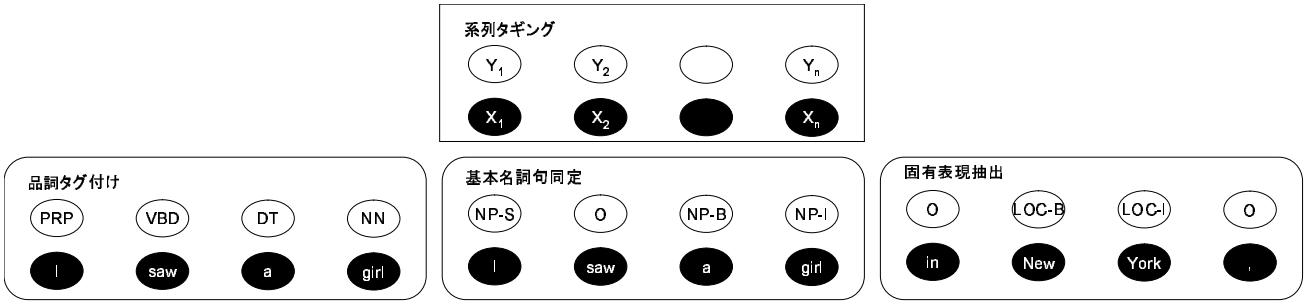

図 1: Sequential tagging で解くことができるタスク

2 Hidden Markov Model

2.1 モデルの概要

Hidden Markov Model は、マルコフ過程を持つ離散状態と、状態毎に定義される出力分布を持つモデルである。同一の出力をする状態が複数あり、実際に観測される出力列からは内部の状態列を特定することができないため “Hidden” と呼ばれる。

隠れ変数 (Y_i) に割り当てられる値の集合を $T = \{t^1, t^2, \ldots, t^m\}(|T| = m)$、観測変数 (=observed variable) (X_i) に割り当てられる値の集合を $W = \{w^1, w^2, \ldots, w^l\}(|W| = l)$ とする。

最も良いタグ集合を見つけるためには、与えられた X に対して $P(Y|X)$ を最大化する Y を発見すれば良い。式に書くと

$$\begin{aligned} arg\max_{Y} P(Y|X) &= arg\max_{Y} \frac{P(X|Y)P(Y)}{P(X)} \\ &= arg\max_{Y} P(X|Y)P(Y) \end{aligned}$$

となる。1 行目は Bayes' rule の適用による。

ここでいくつかの独立仮定を置いて近似する：

- Limited horizon(直前のタグ情報にのみ依存): [1]

$$P(Y_{i+1} = t^j|Y_1, \ldots, Y_i) \approx P(Y_{i+1} = t^j|Y_i)$$

- Time invariant(時間不変):

$$P(Y_{i+1} = t^j|Y_i) \approx P(Y_2 = t^j|Y_1)$$

- 個々の観測変数 X_i は互いに独立
- 観測変数 X_i は対応する隠れ変数 Y_i にのみ依存

これら近似により $P(X|Y)P(Y)$ は以下のように展開される：

$$\begin{aligned} P(X|Y)P(Y) &\approx \prod_{i=1}^{n} P(X_i|Y) \\ &\quad \times P(Y_n|Y_{1,n-1}) \times P(Y_{n-1}|Y_{1,n-2}) \times \cdots \times P(Y_2|Y_1) \times P(Y_1|Y_0 = BOS) \\ &\approx \prod_{i=1}^{n} P(X_i|Y_i) \\ &\quad \times P(Y_n|Y_{n-1}) \times P(Y_{n-1}|Y_{n-2}) \times \cdots \times P(Y_2|Y_1) \times P(Y_1|Y_0 = BOS) \\ &= P(Y_1|Y_0 = BOS) \times \prod_{i=1}^{n} [P(X_i|Y_i) \times P(Y_i|Y_{i-1})] \end{aligned}$$

2.2 タグつきコーパスからのモデル推定

訓練時にタグつきコーパスが利用可能なときに、隠れ変数の正しいタグ割り当ての情報が利用可能である。そのような場合コーパス中の頻度を用いて、確率値を最尤推定することができる。$C(E)$ を事象 E が生起する頻度とすると：

初期状態確率 initial state probabilities (Π でこれらの確率値の集合を表わす):

$$\pi_{Y_1=t^k} = P(Y_1 = t^k|Y_0 = BOS) = \frac{C(Y_1 = t^k)}{C(Y_0 = BOS)}$$

状態遷移確率 state transition probabilities (A でこれらの確率値の集合を表わす):

$$a_{Y_{i-1}=t^j, Y_i=t^k} = P(Y_i = t^k|Y_{i-1} = t^j) = \frac{C(Y_i = t^j, Y_{i-1} = t^k)}{C(Y_{i-1} = t^j)}$$

シンボル出力確率 symbol emission probabilities (B でこれらの確率値の集合を表わす):

$$b_{Y_i=t^j, X_i=w^l} = P(X_i = w^l|Y_i = t^j) = \frac{C(X_i = w^j, Y_i = t^k)}{C(Y_i = t^j)}$$

[1] 上付きの t^j はタグ集合中のタグタイプを意味する

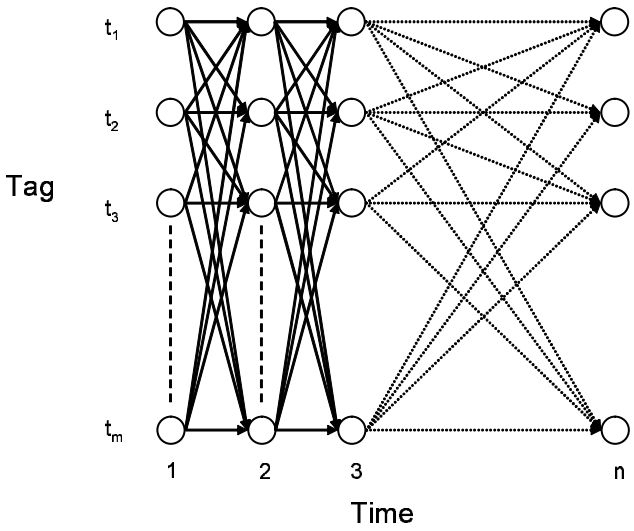

図 2: Lattice

Decoding の際はこれらの値を用いてタグ推定を行なう。次の式によって定義される値 Y を動的計画法の一種である *Viterbi algorithm* によって得る。

$$\begin{aligned}\hat{Y} &= arg\max_{Y} P(Y|X) \\ &= arg\max_{Y=Y_1,\ldots,Y_n} P(Y_1|Y_0 = BOS) \times \prod_{i=1}^{n}[P(X_i|Y_i) \times P(Y_i|Y_{i-1})]\end{aligned}$$

詳しくは 2.4 節で述べる。(See also [3])

2.3 タグなしコーパスからのモデル推定

求めたい確率値の 3 つ組を $\mu = (\Pi, A, B)$ で表現しモデルと呼ぶ。タグなしコーパスの情報 $X_{training}$ を用いて、モデル μ を推定することを考える。最尤推定法を使って、 $P(X_{training}|\mu)$ を最大化するようなパラメータ集合を見つける：

$$arg\max_{\mu} P(X_{training}|\mu)$$

解析的に μ を求める方法は知られていないが、*iterative hill-climbing algorithm* を用いて、局所的に最大化することができる。*Baum-Welch* もしくは *Forward-Backward algorithm* として知られている。

$$P(X|\mu) = \sum_{Y} P(X|Y,\mu)P(Y|\mu)$$

ここで、

$$\begin{aligned}P(X|Y,\mu) &= \prod_{i}^{n} P(X_i|Y_i,\mu) \\ &= b_{Y_1,X_1} b_{Y_2,X_2} \cdots b_{Y_n,X_n} \\ P(Y|\mu) &= \pi_{Y_1} a_{Y_1,Y_2} a_{Y_2,Y_3} \cdots a_{Y_n,Y_{n+1}}\end{aligned}$$

を代入すると

$$\begin{aligned} P(X|\mu) &= \sum_Y P(X|Y,\mu)P(Y|\mu) \\ &= \sum_Y \pi_{Y_1} \prod_i^n a_{Y_i,Y_{i+1}} b_{Y_i,X_i} \end{aligned}$$

ここで $\sum_Y$ は、Y の全ての値割り当てを考える必要がある。全ての値割り当ての枚挙を回避するために、forward probability α と backward probability β の 2 つを導入する。

Forward probability を推定する手続きを forward procedure と呼ぶ。推定したい forward probability α は以下で定義される：

$$\alpha_{t^j}(i) = P(X_1 X_2 \cdots X_{i-1}, Y_i = t^j|\mu)$$

この値は、系列の先頭から帰納的に推定される。

1. initialization
$$(\forall t^j \in T)\ \alpha_{t^j}(1) = \pi_{t^j}$$
2. induction
$$(\forall t^j \in T)\ \alpha_{t^j}(i+1) = \sum_{t^k \in T} \alpha_{t^k}(i) a_{t^k t^j} b_{t^k X_i}$$
3. total
$$P(X|\mu) = \sum_{t^k \in T} \alpha_{t^k}(n+1)$$

図 3 に induction 時の確率値の推定を示す。

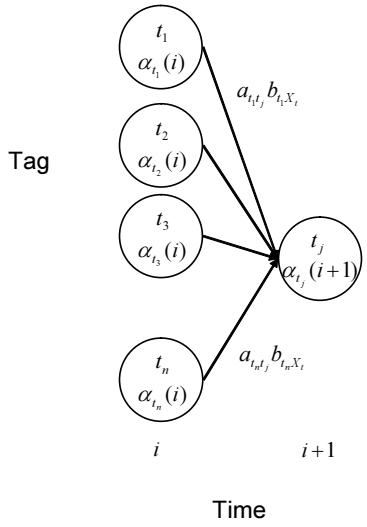

図 3: Forward procedure: 前ノードの forward probability と各 arc の確率値の積の総和により $\alpha_{t^j}(i+1)$ を計算する。

同様に backward probability を推定する手続きを backward procedure と呼ぶ。推定したい backward probability β は以下で定義される：

$$\beta_{t^k}(i) = P(X_i \cdots X_n | Y_i = t^k, \mu)$$

この値は、系列の末尾から帰納的に推定される。

1. initialization

$$(\forall t^k \in T)\ \beta_{t^k}(n) = 1$$

2. induction

$$(\forall t^k \in T)\ \beta_{t^k}(i) = \sum_{t^j \in T} a_{t^k t^j} b_{t^k X_i} \beta_{t^j}(i+1)$$

3. total

$$P(X|\mu) = \sum_{t^j \in T} \pi_{t^j} \beta_{t^j}(1)$$

ここまでは、$P(X|\mu)$ を forward variable α もしくは backward variable β で表現する方法を示した。$P(X|\mu)$ は、系列のある地点 i までの $\alpha_{Y_i}(i)$ と $\beta_{Y_i}(i)$ を用いても表現することができる。$Y_i = t^j$ である場合の $P(X, Y_i = t^j|\mu)$ は、次のようになる。

$$\begin{aligned}
P(X, Y_i = t^j|\mu) &= P(X_1 \cdots X_n, Y_i = t^j|\mu) \\
&= P(X_1 \cdots X_{i-1}, Y_i = t^j, X_i \cdots X_n|\mu) \\
&= P(X_1 \cdots X_{i-1}, Y_i = t^j|\mu) \times P(X_i \cdots X_n|X_1 \cdots X_{i-1}, Y_i = t^j, \mu) \\
&= P(X_1 \cdots X_{i-1}, Y_i = t^j|\mu) \times P(X_i \cdots X_n|Y_i = t^j, \mu) \\
&= \alpha_{t^j}(i)\beta_{t^j}(i)
\end{aligned}$$

このことから

$$P(X|\mu) = \sum_{t^j \in T} \alpha_{t^j}(i)\beta_{t^j}(i)$$

モデル推定の話に戻る。今定義した α と β を用いて、$p_i(t^k, t^j)$ を定義する。

$$\begin{aligned}
p_i(t^k, t^j) &= P(Y_i = t^k, Y_{i+1} = t^j|O, \mu) \\
&= \frac{P(Y_i = t^k, Y_{i+1} = t^j, O|\mu)}{P(O|\mu)} \\
&= \frac{\alpha_{t^k}(i) a_{t^k t^j} b_{t^k X_i} \beta_{t^j}(i+1)}{\sum_{t_u \in T} \alpha_{t_u}(i)\beta_{t_u}(i)} \\
&= \frac{\alpha_{t^k}(i) a_{t^k t^j} b_{t^k X_i} \beta_{t^j}(i+1)}{\sum_{t_u \in T} \sum_{t_v \in T} \alpha_{t_u}(i) a_{t_u t_v} b_{t_u X_i} \beta_{t_v}(i+1)}
\end{aligned}$$

ここで $\gamma_{t^k}(i) = \sum_{t^j \in T} p_i(t^k, t^j)$ とする。この $\gamma_{t^k}(i)$ と $p_i(t^k, t^j)$ はそれぞれ時間軸の index で 総和をとると以下のような意味になる。

- $\sum_{i=1}^{n} \gamma_{t^k}(i)$ = 状態 t^k から遷移する推定頻度
- $\sum_{i=1}^{n} p_i(t^k, t^j)$ = 状態 t^k から状態 t^j へ遷移する推定頻度

ここで、何らかのモデル μ（多くの場合ランダムに分布させたもの）からはじめ、現在のモデルを用いて、観測列（タグなしコーパス）を走らせ、各モデルパラメータの期待値を推定する。次に**よく使われるパス**を最大化するように、モデルを変化させる。これをモデルパラメータ μ を収束するまで繰り返す。

再推定される各パラメータの式は以下の通り：

$$\begin{aligned}
\hat{\pi}_{t^j} &= \text{時間 } t=1 \text{ の時の状態 } t^j \text{ の推定頻度} \\
&= \gamma_{t^j}(1) \\
\hat{a}_{t^k t^j} &= \frac{\text{状態 } t^k \text{ から状態 } t^j \text{ へ遷移する推定頻度}}{\text{状態 } t^k \text{ から遷移する推定頻度}} \\
&= \frac{\sum_{i=1}^{n} p_i(t^k, t^j)}{\sum_{i=1}^{n} \gamma_{t^k}(i)} \\
\hat{b}_{t^k w^l} &= \frac{\text{状態 } t^k \text{ で } w^l \text{ が観測される推定頻度}}{\text{状態 } t^k \text{ から遷移する推定頻度}} \\
&= \frac{\sum_{i:X_i=w^l, 1\leq i\leq n} \gamma_{t^k}(i)}{\sum_{i=1}^{n} \gamma_{t^k}(i)}
\end{aligned}$$

このようにして、$\mu = (A, B, \Pi)$ から $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$ を導出できる。Baum らにより $P(O_{training}|\hat{\mu}) \geq P(O_{training}|\mu)$ が証明されている。

詳しくは FSNLP の Chapter 9 [4] を参照[2] 。

2.4 最適パスの探索

ラティス上の最適なパスの探索は次を満たす Y を見つければよい：

$$arg \max_{Y} P(Y|X, \mu)$$

これを最大化するには、固定された X に対して、以下を最大化するので十分である：

$$arg \max_{Y} P(Y, X|\mu)$$

このパスをラティス上で探索する効率的なアルゴリズム (Viterbi algorithm) がある。

$$\delta_{t^j}(t) = \max_{Y_1,\cdots,Y_{i-1}} P(Y_1, \cdots, Y_{i-1}, X_1, \cdots, X_{i-1}, X_i = t^j|\mu)$$

この変数 $\delta_{t^j}(i)$ は、ラティス上の各点の最尤パスへ導く確率値を格納する。$\psi_{t^j}(i)$ にどこのパスから来たか (backtrace) を格納する。これらの変数は、次の動的計画法により求められる：

1. initialization:

$$\delta_{t_j}(1) = \pi_j \quad (t_j \in T)$$

2. induction:

$$\delta_{t_j}(i+1) = \max_{t^k \in T} \delta_{t_k}(i) a_{t_k t_i} b_{t_k X_i} \quad (t_j \in T)$$

[2]記号類は FSNLP の Chapter 10 に合わせた。また、Chapter 10 では state が symbol を出力するが、Chapter 9 では arc が symbol を出力するモデル。ここでは、混乱を避けるため Chapter 9 の内容を state が symbol を出力するモデルに書き直した。

backtrace を格納する：

$$\psi_{t^j}(i+1) = arg\max_{t^k \in T} \delta_{t_j}(i) a_{t_k t_j} b_{t_k X_i} \quad (t_j \in T)$$

3. 末端まで行い、後ろから ψ をたどることにより最適パスの認定を行う:

$$\hat{X}_{n+1} = arg\max_{t_k \in T} \delta_{t_k}(n+1)$$

$$\hat{X}_i = \psi_{\hat{X}_{i+1}}(i+1)$$

$$P(\hat{X}) = \max_{t_k \in T} \delta_{t_k}(n+1)$$

3 Maximum Entropy Markov Model

Maximum entropy markov model (MEMM)[5] は HMM で入れられなかった観測列に対する overlapping feature を導入することを可能にした。例えば、出現する単語を特徴づけるために capitalization, 語尾の情報, 品詞などを自由に設計することができる。

他の HMM との差異として、HMM は観測列の尤度を最大化するのに対し、MEMM は観測列が与えられて状態列を予測する。HMM では conditional problem を解くために、不適切な joint model (図 4(a)) を適用してきた。これに対して、MEMM は $P(Y_i|Y_{i-1}, X_i)$ を推定することを目標とする (図 4(b))。

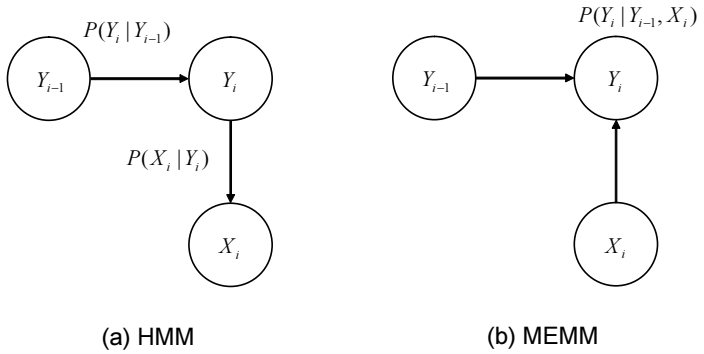

図 4: HMM (a) と MEMM (b) の依存構造グラフ

この model の相違のため HMM と MEMM とで、Viterbi steps が異なってくる。HMM の場合、forward procedure の induction は:

$$\alpha_{t^j}(i+1) = \sum_{t^k \in T} \alpha_{t^k}(i) P(t^k|t^j) P(t^k|X_i)$$

であるが、MEMM の場合 の forward procedure の induction は:

$$\alpha_{t^j}(i+1) = \sum_{t^k \in T} \alpha_{t^k}(i) P_{t^j}(t^k X_i)$$

となる。同様に、MEMM の場合の backward procedure の induction は

$$\beta_{t^k}(i) = \sum_{t^j \in T} P(t^j|t^k, X_i) \beta_{t^j}(i+1)$$

となる。

HMM では、transition function と observation function の 2 つを用いるが、MEMM では state-observation transition function $P(Y_i|Y_{i-1}, X_i)$ のみを使う。簡単化のために $P(Y_i|Y_{i-1}, X_i)$ を、$|T|$ 個の transition function に分割した、$P_{Y_{i-1}=t^k}(Y_i|X_i)$ を導入する。この transition 関数を、複数の独立でない素性に関してモデル化する。このために最大エントロピー法による Exponential model を用いる。

最大エントロピー法はデータから確率分布を確定するための枠組みであり、次のような principle を基にしている：**データに対する最も良いモデルは、訓練データから導出される「ある制約集合」に関して一貫性を持つ。**ここで求めたい分布は制約づけられたものの中で最も一様分布に近いものであり、言い換えると最も高いエントロピーを持った分布である。

「制約」を定義するために素性関数を定義する。$P_{Y_{i-1}=t^k}(Y_i = t^j|X_i = w_l)$ は、前状態が t^k、次状態が t^j、観測が w_l の場合の transition function である。これに対し、二値の出力を持つ素性関数 $f_a(X_i, Y_i)$ を定義する (便宜的に $a = \langle b, t^j \rangle$ とし、b を観測に対する二値の出力を持つ素性関数、t^j を次状態とする)：

$$f_a(X_i, Y_i) = f_{\langle b,t^j \rangle}(X_i, Y_i) = \begin{cases} 1 & \text{if } b(X_i) \text{ が真で かつ } Y_i = t^j \\ 0 & \text{otherwise} \end{cases}$$

観測に対する素性関数 $b(X_i)$ の例として、「X_i が大文字で始まる」、「X_i が *ing* で終わる」などが考えられる。

「制約」は、学習される分布中のこのように定義された各素性関数の期待値とラベル付きの訓練データ中の平均値とが同じであることをいう。形式的には、

$$\frac{1}{n}\sum_{i}^{n} f_a(X_i, Y_i) = \frac{1}{n}\sum_{i}^{n}\sum_{t^j \in T} P_{Y_{i-1}}(Y_i = t^j|X_i) f_a(X_i, Y_i = t^j)$$

この制約を満たしエントロピーを最大にする確率分布は、単一で、最尤解と同じであり、次の exponential form を持つことが知られている [6]：

$$P_{Y_{i-1}}(Y_i|X_i) = \frac{1}{Z(X_i, Y_{i-1})} \exp(\sum_a \lambda_a f_a(X_i, Y_i))$$

λ_a が求めたいパラメータ集合、$Z(X_i, Y_{i-1})$ は全次状態に関して分布を合計する正規化項である。これらの値は、Generalized Iterative Scaling (GIS), Improved Iterative Scaling (IIS), Limited-memory BFGS などのパラメータ推定アルゴリズムを用いて求めることができる。

4 Linear-chain Conditional Random Fields

次に Linear-chain Conditional Random Fields (L-CRF) [2] について解説する。このモデルは、MEMM がもつ Label bias problem を解決する。まず、Label bias problem について説明する。

4.1 Label bias problem

MEMM では、最大エントロピーモデルの順次適用により系列の各ラベルを決定する。元の系列ラベリング問題を個々のラベルを付与するという部分問題に分割し、個々のラベリング確率の積を全体のラベリングに対する確率と近似する。個々のラベルを付与する部分問題は、局所的な情報（2-gram の場合は、1 つ前の単語）のみを用いる。

図 5 に、典型的な label bias problem を示す。この例で、正解は t_1, t_1 という状態列とする。局所的な確率値では正しい系列を推定できるが、系列全体の確率値では t_3, t_3 が勝つ。これは、$Y_1 = t_3$ に後続するラベルが

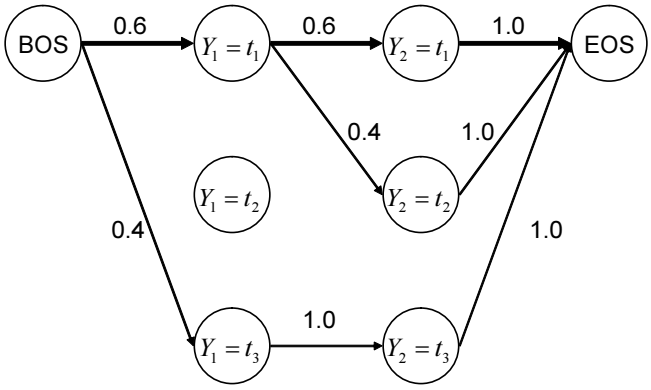

図 5: Label bias problem

$Y_2 = t_3$ のみであり、連接確率が常に 1 になってしまうことによる。系列の本質的な曖昧さがラティスを通る経路に依存し、実際には曖昧性の少ない（分岐の少ない）経路が選ばれやすくなる。これを label bias problem と呼ぶ。MEMM の学習時には、正解の経路のみを考慮し、個々の部分問題が独立に学習される。このため、学習時に考慮されなかった系列の確率値は、解析時に非常に不安定になる。結果として、経路上の曖昧性の影響を受けやすくなる。

4.2 Linear-chain CRF の定義

MEMM が元の系列ラベリング問題を部分問題に分割していたのに対し、CRF では 1 つの指数分布モデルによって各出力系列 Y の入力文 X に対する条件付き確率 $P(Y|X)$ を表現する。

$$P(Y|X) = \frac{1}{Z(X)} \exp(\sum_{i}^{n}(\sum_{a} \lambda_a f_a(X_i, Y_i)) + (\sum_{b} \nu_b g_b(Y_{i-1}, Y_i)))$$

$$Z(X) = \sum_{Y \in T^n} \exp(\sum_{i}^{n}(\sum_{a} \lambda_a f_a(X_i, Y_i)) + (\sum_{b} \nu_b g_b(Y_{i-1}, Y_i)))$$

ここで、f_a は、observation function に対する素性、g_b は、transition function に対する素性をさす[3]。$\lambda_a (\in \Lambda = \{\lambda_1, \ldots, \lambda_A\} \in \mathcal{R}^A)$ は素性関数 f_a に対する重み、$\nu_b (\in N = \{\nu_1, \ldots, \nu_B\} \in \mathcal{R}^B)$ は素性関数 g_b に対する重みであり、正しい出力系列を他の全出力候補と弁別するように選択される。$Z(X)$ は、他の全候補を考慮するための正規化項である。このように他の全候補を考慮する点が MEMM との違いであり、これにより label bias を低減することができる。

ここで記号の簡単化のために、大域素性ベクトル $F(Y,X) = \{F_1(Y,X), \ldots, F_A(Y,X)\}$ と $G(Y,X) = \{G_1(Y,X), \ldots, G_B(Y,X)\}$ を与える。但し $F_A = \sum_{i=1}^{n} f_a(X_i, Y_i)$ また $G_B = \sum_{i=1}^{n} g_b(Y_{i-1}, Y_i)$ とする。大域素性を用いると、$P(Y|X)$ は以下のように書ける：

$$P(Y|X) = \frac{1}{Z(x)} \exp(\Lambda \cdot F(Y,X) + N \cdot G(Y,X))$$

入力 X に対する最適なラベル列 $\hat{Y}$ は、

$$\hat{Y} = arg\max_{Y} P(Y|X) = arg\max_{Y} \Lambda \cdot F(Y,X) + N \cdot G(Y,X)$$

[3]CRF の定義では、素性は clique (最大部分完全グラフ) に対して与えられるため分割した。

となる。この最適系列は Viterbi アルゴリズムを用いて効率良く探索できる。ここで、$\Lambda \cdot F(Y,X) + N \cdot G(Y,X)$ を系列 Y のコストと呼ぶ。

4.3 パラメータ推定

CRF は、一般的な最尤推定を用いてパラメータを選択する。訓練データ $D = \{\langle X^u, Y^u \rangle\}_{u=1}^{\text{文の数}}$ に対する対数尤度 $\mathcal{L}_{\Lambda,N}$ の最大化を行う。

$$
\begin{aligned}
\mathcal{L}_{\Lambda,N} &= \sum_u \log(P(Y^u|X^u)) \\
&= \sum_u [\log(\sum_Y \exp(\Lambda \cdot [F(Y^u,X^u) - F(Y,X^u)] + N \cdot [G(Y^u,X^u) - G(Y,X^u)]))] \\
&= \sum_u [\Lambda \cdot F(Y^u,X^u) + N \cdot G(Y^u,X^u) - log(Z(X^u))] \\
\hat{\Lambda}, \hat{N} &= arg \max_{\Lambda \in \mathcal{R}^A, N \in \mathcal{R}^B} \mathcal{L}_{\Lambda,N}
\end{aligned}
$$

$\mathcal{L}_{\Lambda,N}$ を大きくするためには、各学習データ $\langle X^u, Y^u \rangle$ に対し、$\sum_Y \exp(\Lambda \cdot [F(Y^u,X^u) - F(Y,X^u)] + N \cdot [G(Y^u,X^u) - G(Y,X^u)])$ を大きくすればよい。これは、正解のパスコスト $\Lambda \cdot F(Y^u,X^u) + N \cdot G(Y^u,X^u)$ と、残りの全候補のコスト $\Lambda \cdot F(Y,X^u) + N \cdot G(Y,X^u)$ との差をできるだけ大きくすることに相当する。これにより、ラティス中の全候補系列から正解の系列のみを分別するような効果が生まれる。

目的関数の凸性から、最適点では以下が成立する：

$$
\begin{aligned}
\frac{\delta \mathcal{L}_{\Lambda,N}}{\delta \lambda_a} &= \sum_u (F_a(Y^u,X^u) - E_{P(Y|X^u)}[F_a(Y,X^u)]) \\
&= O_a - E_a = 0 \\
\frac{\delta \mathcal{L}_{\Lambda,N}}{\delta \nu_b} &= \sum_u (G_b(Y^u,X^u) - E_{P(Y|X^u)}[G_b(Y,X^u)]) \\
&= O_b - E_b = 0
\end{aligned}
$$

但し、$O_a = \sum_u F_a(Y^u,X^u)$ は素性 a の学習データ D における出現頻度、$O_b = \sum_u G_b(Y^u,X^u)$ は素性 b の学習データ D における出現頻度、$E_a = \sum_u E_{P(Y|X^u)}[F_a(Y,X^u)]$ は素性 a のモデル分布における出現期待値、$E_b = \sum_u E_{P(Y|X^u)}[G_b(Y,X^u)]$ は素性 b のモデル分布における出現期待値である。この期待値の計算は、単純には出力系列の全候補を陽に枚挙することにより実現できるが、この候補数は入力文の長さに対して指数的に増えるために、事実上困難である。しかし、forward-backward アルゴリズムの変種を用いることにより効率良く計算することができる (式中で $V(t^k,t^j)$ は、スペースの関係上便宜的に入れた関数)：

$$
\begin{aligned}
E_{P(Y|X)}[F_a(Y,X)] &= \sum_{t^k \in T, t^j \in T, 0 \le i \le n} \frac{\alpha_{t^k}(i) \cdot f_a(X_{i+1}, Y_{i+1} = t^j) \cdot V(t^k,t^j) \cdot \beta_{t^j}(i+1)}{Z(X)} \\
E_{P(Y|X)}[G_b(Y,X)] &= \sum_{t^k \in T, t^j \in T, 0 \le i \le n} \frac{\alpha_{t^k}(i) \cdot g_b(Y_i = t^k, Y_{i+1} = t^j) \cdot V(t^k,t^j) \cdot \beta_{t^j}(i+1)}{Z(X)} \\
V(t^k,t^j) &= \exp((\sum_a \lambda_a f_a(X_{i+1}, Y_{i+1} = t^j)) + (\sum_b \nu_b g_b(Y_i = t^k, Y_{i+1} = t^j)))
\end{aligned}
$$

ここで、α と β は forward-backward コストで以下のように再帰的に定義される：

$$
\begin{aligned}
\alpha_{BOS}(0) &= 1 \\
\alpha_{t^j}(i+1) &= \sum_{t^k \in T} \alpha_{t^k}(i) \cdot \exp((\sum_a \lambda_a f_a(X_{i+1}, Y_{i+1} = t^j)) + (\sum_b \nu_b g_b(Y_i = t^k, Y_{i+1} = t^j))) \\
\beta_{EOS}(n+1) &= 1 \\
\beta_{t^k}(i) &= \sum_{t^j \in T} \exp((\sum_a \lambda_a f_a(X_{i+1}, Y_{i+1} = t^j)) + (\sum_b \nu_b g_b(Y_i = t^k, Y_{i+1} = t^j))) \cdot \beta_{t^j}(i+1)
\end{aligned}
$$

以上を用いると、正規化項は $Z(X) = \alpha_{EOS}(n+1) = \beta_{BOS}(0)$ となる。

5 To be written

- 用語の統一！
- GIS, IIS, L-BFGS, L-BFGS-B
- パラメータの正則化 (Laplacian Prior: L1-norm, Gaussian Prior: L2-norm)
- Semi-CRF
- D-CRF

6 実装

6.1 パッケージ

- MaxEnt (not MEMM)
  - openNLP maxent
    `http://maxent.sourceforge.net/`
- CRF
  - CRF package by Sunita Sarawagi
    `http://crf.sourceforge.net/`
  - CRF++
    `http://chasen.org/~taku/software/CRF++/`
  - FlexCRF
    `http://www.jaist.ac.jp/~hieuxuan/flexcrfs/flexcrfs.html`

6.2 Tips

CRF では以下の計算が多用される。

$$log(exp(x) + exp(y))$$

x, y が非常に大きい時, $exp(x)$ や $exp(y)$ を計算する時点でオーバフローが起きる。

http://crf.sourceforge.org のソースや CRF++ などでは、以下のような近似を用いている[4]：

```
double logsumexp (double x, double y, bool flg)
{
   if (flg) return y; // init mode
   if (x == y) return x + 0.69314718055;  // log(2)
    double vmin = std::min (x, y);
    double vmax = std::max (x, y);
    if (vmax > vmin + 50) {
      return vmax;
    } else {
      return vmax + std::log (std::exp (vmin - vmax) + 1.0);
    }
}
```

flg は初期値の設定のために用いる。
また, $x = log(exp(x) + exp(y))$ として、logsumexp (x,z) を計算すると

$$log(exp(log(exp(x) + exp(y))) + exp(y)) = log(exp(x) + exp(y) + exp(z))$$

となり、再帰的に適用することが可能である。
$Z = log(exp(a[0]) + exp(a[1]) + ...exp(a[n-1]))$ を計算する時には、以下のようにすることができる。

```
double Z = 0.0;
for (int i = 0; i < n; ++i)
  Z = logsumexp (Z,  a[i], (i == 0))
```

参考文献

[1] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Appliying conditional random fields to japanese morphological analysis. In *EMNLP-2004*, 2004.

[2] John Lafferty, Andrew McCallum, and Fernand Pereira. Conditional random fields: Probabilisitic models for segmenting and labeling sequence data. In *ICML-2001*, 2001.

[3] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing*, chapter Chapter 10. Part-of-Speech Tagging. The MIT Press, 1999.

[4] Christopher D. Manning and Hinrich Schuetze. *Foundations of Statistical Natural Language Processing*, chapter Chapter 9. Markov Models. The MIT Press, 1999.

[5] Andrew McCallum, Dayne Freitag, and Fernand Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML-2000*, 2000.

[6] S. Della Pietra, V. Della Pietra, and J. Laffety. Inducing features of random fields. *IEEE transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 4, 1997.

[4] $x > y$ とすると $e^x + e^y = e^x(1 + \frac{e^y}{e^x}) = e^x(1 + e^{y-x})$ ある程度 x が大きいとき、e^y を無視する。