Part-of-Speech (POS) Tagging

1. Perceptron Algorithm.

- Given word sequence $X^{1\ldots S}$, POS tagging $Y^{1\ldots S}$, feature ϕ

  Initialize $w \leftarrow 0$

  Repeat till converge:

  For sentence $s = 1$ to S do

  $V^s = \arg\max_{Y} \sum_{j=1}^{I} w_j \sum_{i=1}^{N_s} \phi_i(X^s, Y, j) \Rightarrow$ Using Viterbi Decoding

  if $V^s \neq Y^s$ do

  $f(X^s, Y^s) \leftarrow \sum_{i=1}^{N_s} \phi(X^s, Y^s, i)$

  $f(X^s, V^s) \leftarrow \sum_{i=1}^{N_s} \phi(X^s, V^s, i)$

  $\hat{w} \mathrel{+}= f(X^s, Y^s) - f(X^s, V^s)$

- Viterbi Decoding.

  $\cancel{V^s_{1\ldots N} = \infty}$ Allocate δ with size$(|T| N)$ T-#tags

  For $i = 1$ to N do

  For $t \in T$ do

  $\delta(i, t) = -\infty$

  For $t' \in T$ do

  $\delta(i,t) = \max\{\delta(i,t),\ \delta(i-1,t') + \sum_{j=1}^{I} w_j \phi_j(X, Y, i)\}$

  return δ

2. HMM: $P(y_1^N, x_1^N) = \prod_{i=1}^{N} P(y_i | y_{i-1}) P(x_i | y_i)$ y - hidden state, x — observation.

- Viterbi Decoding: find the best tagging (hidden) sequence, given sentences of words.

$$y^* = \arg\max_{y_1^N} P(y_1^N | x_1^N) \Leftrightarrow P(y_1^N, x_1^N)$$

Allocate δ with size $|T|N$, $\delta = -\infty$

```
For i = 1 to N do
    For t = 1 to T do
        For t' = 1 to T do
            δ(i,t) = max{ δ(i,t), δ(i-1,t') P(t|t') P(x_i|t) }
return δ
```

- Minimum Bayes Risk: $\sum_{i=1}^{N} \max_{\hat{y}_1^N} P(\hat{y}_i = y_i | x_1^N)$.

Forward-Backward Algorithm.

define: $\alpha(i,t) = P(x_1^i, y_i = t)$

$\beta(i,t) = P(x_{i+1}^N | y_i = t)$

Allocate α with size $|T|N$, $\alpha = 0$, $\alpha(1, START) = 1$

```
For i = 1 to N do
    For t = 1 to T do
        For t' = 1 to T do
            ~~α(i,t) = α(i,t') +~~
            α(i,t) += P(y_i = t | y_{i-1} = t') P(x_i | y_i) α(i-1, t')
```

Allocate β with size $|T|N$, $\beta = 0$, $\beta(N,:) = 1$.

```
For i = N to 1 do
    For t = 1 to T do
        For t' = 1 to T do.
            β(i,t) += P(y_{i+1} = t' | y_i = t) P(x_{i+1} | y_{i+1}) β(i+1, t')
```

Best tagging sequence:

$$\left[\arg\max_{\hat{y}_i} \alpha(i, \hat{y}_i)\beta(i, \hat{y}_i)\right]_{i=1}^{N}$$

- EM Algorithm for parameters learning.

First compute α and β for all sequences in data.

E-step: expected count:

$$\begin{cases} ec(t,x) = \sum_{i, s.t. X_i = x} P(Z_i = t \mid X_1^N) = \underline{\sum \frac{1}{Z} \alpha(i,t)\beta(i,t)} \\ ec(t,t') = \sum_i P(Z_i = t \wedge Z_{i+1} = t' \mid X_1^N) \end{cases}$$

$$= \underline{\sum_i \frac{1}{Z} \alpha(i,t) P(t' \mid t) P(x_{i+1} \mid t') \beta(i+1, t')}$$

M-step:

$$\theta_{t,x} = \frac{ec(t,x)}{\sum_{x'} ec(t,x')}$$

$$A_{t,t'} = \frac{ec(t,t')}{\sum_{t''} ec(t,t'')}$$

3. Conditional Random Field (CRF). $P(Y|X) = \frac{1}{Z} e^{\sum_j w_j f_j(Y,X)}$. (max entropy).

- Learn the weights of CRF:

  First calculate α and β using sentences in data.

  where the update rule of α is modified as:

  $$\alpha(i,t) += \alpha(i-1,t')\, e^{\sum_j w_j \phi_j(x, y_{i-1}, y_i)}$$

  $$\beta(i,t) += \beta(i+1,t')\, e^{\sum_j w_j \phi_j(x, y_i, y_{i+1})}.$$

key advantage over HMM: flexibility in the features can be used

Upate w through gradiant dascent: (CRF has concave loss function).

$$\frac{\partial L}{\partial w} = f(Y,X) - \mathbb{E}_{P_w(Y|X)}[f]$$

where $\mathbb{E}_{P_w(Y|X)}[f_j] = \sum_i \sum_{t,t'} f_j(Y_i = t, Y_{i+1} = t', X_1^N)\, \underbrace{\alpha(i,t)\beta(i+1,t')\frac{1}{Z} e^{w^T f(Y_i, Y_{i+1}, X_1^N)}}_{P_w(Y_i = t,\, Y_{i+1} = t' | X_1^N)}$.

$$w += \eta \frac{\partial L}{\partial w}$$

- Latent CRF: $P(Z,Y|X) = \frac{1}{Z(X)} e^{\sum_j w_j f_j(Z,Y,X)}$.

  $$\frac{\partial L}{\partial w} = \mathbb{E}_{P(Z|Y,X)}[f] - \mathbb{E}_{P(Z,Y|X)}[f]$$

4. Structured SVM: max margin as a metric for structured prediction.

- $\min \frac{1}{2}\|w\|^2 + C \sum_n \xi_n$.

  s.t. $w^T(f(x_n,y_n) - f(x_n,y)) \geq L(y_n,y) - \xi_n \quad \forall n, y$ $\rightsquigarrow$ to all possible y

  where $L(y_n,y) = \sum_{i=1}^{|x_n|} I(y_{n,i} \neq y_i)$ is loss function.

- Training (compared with perceptron for tagging).

  update rule: $w \leftarrow w - \eta\left(\frac{w}{N} - C(f(x_n,y_n) - f(x_n,\hat{y}))\right)$

  where $\hat{y} = \arg\max_y L(y_n,y) - w^T(f(x_n,y_n) - f(x_n,y))$.

- Solve $\hat{y}$ through Loss Augmented Decoding:

  For $i = 1$ to $|x_n|$ do

    For $t = 1$ to T do

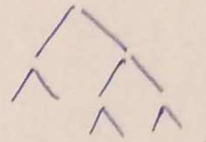
      For $t' = 1$ to T do

  $$\delta(i,t) = \max\{\delta(i,t),\ \delta(i-1,t') + \ell(x_n, y_{n,i}, t) - w^T(f(x, y_{n,i-1}, y_{n,i}) - f(x_n, t', t))\}$$

Parsing

Key Points:
- Contex Free ~~Grammar~~ (CFG) / Probabilistic CFG (PCFG)
- Viterbi Decoding for parsing
- Posterior Decoding for parsing (Inside-Outside Algorithm).

1. CFG.



- Parsing Problem: to define a syntatic structure (parsing tree)
- CFG: $G=(S,N,T,R)$: S – start symbol, N – non-terminals
  T – terminals, R – rules ($X \to Y_1 Y_2 \cdots Y_n$, for $n \geq 0$, $X \in N$, $Y_i \in (N \cup T)$).
- PCFG: $\sum_B P(A \to B) = 1 \; . \; \forall A$
  $P(\text{derivation}) = \prod_{r \in d} P(r) \Rightarrow$ probability of a tree: $p(t) = \prod_{i=1}^{n} q_\theta(\alpha_i \to \beta_i)$.
- HMM is a special case of PCFG:

2. Viterbi Decoding: With probabilities of all rules in CFG, find the best parsing tree

- $tree^* = \arg\max_{tree} P(\text{sentence}, \text{tree}) = \arg\max_{tree} \prod_{r \in tree} P(r)$
- Dynamic Programming: $O(N^3|R|^3)$
  define $\delta(i,j,A)$ = maximum prob. of a constituent with non-terminal A ~~spanning~~ spanning words $i \cdots j$ inclusive

- Chomsky Normal Form (CNF).
  $\begin{cases} A \to BC \\ A \to a \end{cases}$
- every CFG can be transformed into CNF

Allocate δ with $\text{size}(|R| \cdot N^2)$, all $\delta = -\infty$

```
For i = 1 to N do
    For all X ∈ N
        δ(i, i+1, X) = { P(X → x_i)   if X → x_i ∈ R
                       { 0            otherwise

For span = 2 to N do
    For i = 1 to N+1-span do
        j = i + span
        For k = i to j do
            For A, B, C ∈ N do
                δ(i, j, A) = max{ δ(i, j, A), P(A → BC) δ(i, k, B) δ(k, j, C) }

return δ(1, N+1, S)
```

3. Posterior Decoding: despite of maximizing tree prob., also maximize the correctness of each node (Minimum Baysian Risk, MBR).

$$\max_{\hat{t}} \sum_{n \in \hat{t}} P(n \in t) \Rightarrow P(n \in t) = \frac{\alpha(i,j,A)\,\beta(i,j,A)}{\beta(1,N+1,S)}$$

- Inside-Outside Algorithm.

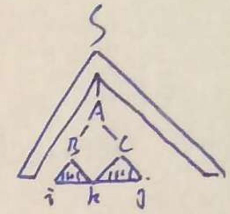
define: $\cancel{\alpha(i,k,X) = P(S \to^* x_1 \cdots X x_{k+1} \cdots x_{N+1})}$

$$\alpha(i,j,X) = P(S \to^* x_1 \cdots x_i X x_j \cdots x_{N+1})$$

$$\beta(i,k,X) = P(X \to^* x_i \cdots x_k)$$

$$S(A \to BC, i,j,k) = \frac{1}{Z}\alpha(i,j,A)\,\beta(i,k,B)\,\beta(k,j,C)\,P(A \to BC).$$

$\alpha(i,j,A)$



$\beta(i,k,B), \beta(k,j,C)$

```
Allocate α and β with size |R| N^2
  α(1, N, S) = 1
  # calculate β first.
  For i = 1 to N do
    For all X ∈ N
      β(i, i+1, X) = { P(X → x_i), if X → x_i in R
                     { 0           otherwise.

  For span = 2 to N do
    For i = 1 to N+1-span do
      j = i + span
      For k = i to j do
        For A, B, C ∈ N do
          β(i, j, A) += β(i, k, B) β(k, j, C) P(A → BC)

  # then calculate α
  For span = N to 2 do
    For i = 1 to N+1-span do
      j = i + span
      For k = i to j do
        For A, B, C ∈ N do
          α(i, k, B) += α(i, j, A) β(k, j, C) P(A → BC)
          α(k, j, C) += α(i, j, A) β(i, k, B) P(A → BC).

  return α and β
```

- Calculate δ with α and β: build the most likely tree, but need to have a valid tree structure

Allocate δ with size $|R|N^2$, $\delta = -\infty$

```
For span = 2 to N do
    For i = 1 to N+1-span do
        j = i+ span
        For k = i to j do
            For A,B,C ∈ N do
                δ(i,j,A) = max{δ(i,j,A), δ(i,k,B)+δ(k,j,C)+α(i,j,A)β(i,j,A)}

    return δ
```

Machine Translation.

Key Points:
- Word Alignment. (IBM Models)
- Phrased Based Translation
- Synchronous CFG.

1. Word Alignment.

Task: French $\rightarrow$ English.

- Noisy Channel Model.

$$p(e|f) = \frac{p(f|e)\,p(e)}{p(f)}.$$

$p(e)$: language model

$p(f|e)$: translation model (backwards!)

- IBM Model 1

alignment sequence: $a: \{a_1, a_2 \cdots a_m\}$ m — french sentence length. l — Eng. sentence length

$$p(f, a|e, m) = p(a|e, m)\,p(f|a, e, m) = \underbrace{p(a|e, m)}_{\text{alignment prob.}}\,\underbrace{p(f|e_a)}_{\text{transition prob.}}$$

- Assumption 1: $p(a|e, m) = \underbrace{\frac{1}{(l+1)^m}}_{\text{constant}}$ (every alignment is equally probable).

- Assumption 2: Each ~~target~~ source word corresponds to only one target word.

$p(f_j|e_{a_j})$ — learned using EM.

IBM Model 1: $\underline{p(f, a|e, m) = \frac{c}{(l+1)^m} \prod_j p(f_j|e_{a_j})}$.

- IBM Model 2
  - Introduce distortion parameters: $q(i|j, l, m)$ — i: index of Eng., j: index of French.

$$p(a|e, m) = \prod_{j=1}^{m} q(a_j|j, l, m).$$

IBM Model 2: $p(f, a|e, m) = \prod_{j=1}^{m} p(f_j|e_{a_j})\,q(a_j|j, l, m)$.

- HMM Model.

$$p(f, a|e, m) = \prod_{j=1}^{m} p(f_j|e_{a_j})\,p(a_j|a_{j-1}, l, m).$$

- EM for IBM Model 2.

Initial $q(\ )$, $p(\ )$ to be random values

Allocate δ with size $|E||F|$

E-step: For all sentence:

For $i = 1$ to m do.

For $j = 0$ to l do // 0 for NULL symbol.

$count(e_j, f_i) += p(f_i | e_j) q(j | i, l, m) \frac{1}{Z}$

$count(e_j) +=$ ..

$count(j, i, l, m) +=$ ..

$count(i, l, m) +=$ ..

~~Normalize~~

M-step: $p(f_i | e_j) = \frac{count(e_j, f_i)}{count(e_j)}$ $q(j | i, l, m) = \frac{count(j, i, l, m)}{count(i, l, m)}$.

2. Phrased Based Translation

① Alignment Matrix

Step 1: train IBM Model 2 for $p(f|e)$, find best alignment: $a_1^* = \arg\max_a p(f, a|e, m)$

Step 2: train . . . for $p(e|f)$, find best alignment: $a_2^* = \arg\max_a p(e, a|f, l)$

Step 3: Intersection of two most likely alignment.

Step 4: Growth algorithm to get alignment matrix.

② Extract Phrase Pairs from Alignment Matrix.

- Pair (e, f) is concistent if:
  - (1) at least one word in e aligned to a word in f
  - (2) no words in f aligned to words outside e.
  - (3) no words in e aligned to words outside f.

	Mary	did	not
Maria	o		
no		o	o

- Probabilities of Phrase Pairs

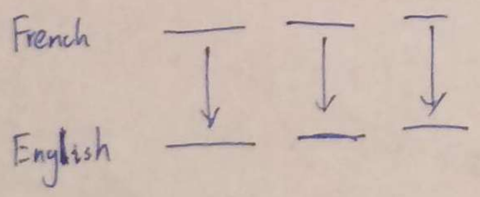
$$p(f|e) = \frac{count(f,e)}{count(e)}.$$

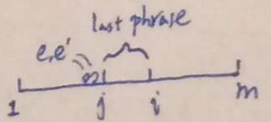
③ Decoding with Phrased Based Model:

- Do not consider reordering: only phrase model (transition prob.) and language model.

Dynamic Programming: state = $[i, \underbrace{e, e'}]$ last two english words, for tri-gram language model

For $i = 1$ to m do
    For $j = 1$ to i do   // determine the length of last phrase
        For phrase ph s.t. lexicon (p, f_j^i) do
            For e, e' do
                $\delta(i, u, u')$ max= $\delta(j, e, e') + \log p(ph|f_j^i) + P_{LM}(ph|e, e')$





- Consider <u>reordering</u> and distortion parameters.

Beam Search: state: $(e, e', b, r, \underbrace{a}_{\text{score}})$ initial state $(*, *, 0^m, 0, 0)$.

b: bit-string indicating covered position

r: end-point of last phrase

functions:

- $ph(q_e)$: possible states $\begin{cases} \text{must not overlap with } b \\ \text{distortion limit must not be violated.} \end{cases}$
- $next(q_e, p)$: combining state q_e with phrase p
- $eq(q_e, q_{e_2}^*)$: return true if $e = e_r$, $e' = e'_2$, $b = b_2$, $r = r_2$ (ignoring scores)
- $Add(Q, \hat{q}_e, q_e, p)$: If $eq(\hat{\hat{q}}_e, \hat{q})$:
  - If $a(\hat{q}_e) > a(\hat{\hat{q}}_e)$:
    - $Q = \{\hat{q}_e\} \cup Q \setminus \{\hat{\hat{q}}_e\}$
  - Else:
    - $Q = Q \cup \{\hat{q}_e\}$
- $beam(Q) = \{q_e \in Q : a(q_e) \geq a^* - \beta\}$

  $\beta \geq 0$: beam-width parameter

  $a^* = \max_{q_e \in Q} a(q_e)$.

```
Initialize Q_0 = {q_e0}, Q_i = {φ}
For i = 0 to m-1
    For each state q_e ∈ beam(Q_i) do
        For each phrase p ∈ ph(q_e) do
            q̂_e = next(q_e, p)
            Add(Q_i, q̂_e, q_e, p)    (where i = len(q̂_e))
return the highest score state in Q_n.
```

score: $a(\hat{q}_e) = a(q_e) + \underbrace{\log p(f|e)}_{\text{(phrase model)}} + \underbrace{\log P_{lm}(\hat{q}_e)}_{\text{(language model)}} + \underbrace{\eta \times |j+1 - r(q_e)|}_{\text{(distortion)}}$.