Pertemuan 5

Sequence Labelling dan Hidden Markov Model

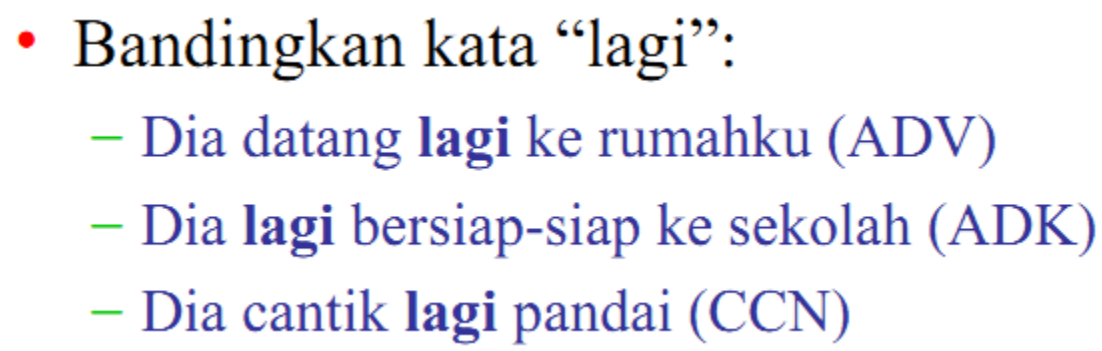
Kita sudah mempelajari tentang machine learning di kelas yang lain. Machine learning bisa berfungsi untuk regresi maupun klasifikasi. Dengan memasukkan data yang cukup dan algoritma tertentu, kita bisa mendapatkan model yang sesuai untuk memprediksi data yang baru.

Machine learning yang kategorinya klasifikasi memetakan **feature** ke sejumlah **kelas**. Jadi misalnya kita punya data yang menggambarkan huruf A, B, C, dan seterusnya untuk training OCR. Algoritma machine learning akan mencari feature dari masing masing data tersebut yang menggambarkan huruf-huruf yang bersesuaian. Setelah melewati masa training, didapatkan modelnya (dalam bentuk matriks) yang kalau dibahasakan mungkin akan berbunyi : “kalau ada garis miring dari bawah ke atas, terus turun lagi itu masuk kelas A”, “kalau ada lengkung yang bukaannya menghadap ke kanan itu C, G, O, Q, atau S”, kalau ada lengkung yang bukaannya menghadap ke kiri itu B, D, O, P, Q, R, atau S), dst.

Algoritma klasifikasi mengasumsikan bahwa setiap kasus (misal data untuk A yang pertama, data untuk A yang kedua, data untuk B, dst) adalah independen. Artinya tidak terkait antara satu sama lain. Jadi pemodelannya cukup memperhitungkan masing masing saja.

Tetapi regresi maupun klasifikasi saja kurang tepat digunakan untuk banyak kasus NLP, karena permasalahan dalam NLP banyak yang terkait dengan barisan (sequence). Data pada barisan saling terkait satu sama lain (mutually dependent). Contohnya pada waktu kita mau melabeli kalimat dengan POS Tag, label kata ke-n akan terkait dengan label sebelum dan sesudahnya.

Simak kembali kalimat pada slide halaman 12.



untuk melabeli kata “lagi”, tergantung dengan label kata sebelum dan sesudahnya. Misalnya kata sebelumnya dan sesudahnya sama, kemungkinan besar label dari kata “lagi” adalah kata penghubung (conjunction), yaitu CCN. Kalau kata setelah kata “lagi” adalah kata kerja, kemungkinan labelnya adalah ADK. Kalau kata sebelum kata “lagi” adalah kata kerja, kemungkinan labelnya adalah ADV, dan seterusnya. Jadi kita tidak bisa melabeli katanya hanya berdasarkan “lagi” saja, karena terlalu banyak kemungkinannya.

Hal-hal seperti ini disebut dengan **sequence labelling problem**. Banyak problem yang bisa kita selesaikan menggunakan cara yang sama.

Contohnya speech recognition. Ada dua bagian dalam speech recognition yaitu modeling akustik dan modeling bahasanya. Keduanya sama sama bisa kita perlakukan seperti sequence labeling. Modeling akustik mengubah gelombang suara ke dalam fonem yang sesuai, sedangkan modeling bahasa mengubah rangkaian fonem menjadi kata-kata.

Pada modeling akustik, mesin akan memodelkan fitur suara yang didapat dari gelombang. Gelombang sekian hertz (Hz) jika sebelumnya sekian Hz dan sesudahnya sekian Hz, maka fonem yang cocok adalah x, dst. Seakan akan kita melabeli gelombang suara dengan fonem (simbol suara).

Pada modeling bahasa, mesin mengubah fonemnya ke kata yang sesuai. Sama sama terdengar fonem "**aa**". Tapi ditulisnya bisa huruf "U" (seperti pada saat mendengar kata **U**mbrella) atau huruf "A" (saat mendengar kata indonesi**A**). Seakan akan kita melabeli barisan fonem dengan huruf yang sesuai untuk kata tersebut.

Contoh lain, motion detection pada video. Misal kita punya 3 kelas label untuk mewakili gerakan pada video, yaitu "berdiri", "duduk/jongkok", dan "rebahan/tiduran". Label tersebut akan digunakan untuk mendeteksi gerakan senam, misalnya. Maka gambar pada video tersebut dicapture scene by scene, dideteksi objeknya (manusia), misalnya dengan membuat bounding box (kotak yang melingkupi objek), dan dideteksi gerakannya (kelas) menggunakan informasi lebar dan panjang dari bounding box (feature). Sama ukurannya, belum tentu gerakannya sama. Misalnya pada saat dia jongkok tapi kakinya agak melebar, jadi boxnya lebih besar ukuran horizontalnya (yang ke kanan kiri) dibanding vertikal (yang ke atas bawah). Bisa saja salah deteksi sebagai rebahan karena boxnya melebar. Atau sebaliknya, posisinya rebahan, sedang push up misalnya, tetapi gambarnya diambil dari depan, bisa salah deteksi sebagai jongkok (dengan catatan bahwa fitur yang digunakan hanya lebar dan tinggi boxnya saja. Makin banyak fitur berbeda yang digunakan akan semakin akurat pendeteksiannya).

Tapi jika kita melihat gerakan sebelumnya, maka peluang untuk salah deteksinya bisa mengecil. Misalnya gerakan sebelumnya adalah "berdiri", jarang gerakan selanjutnya adalah "rebahan", kemungkinan besar "jongkok" dulu. Ini juga adalah sequence labelling problem. Kita melabeli barisan gerakan (yang diwakili oleh data lebar dan tinggi kotak) dengan 3 kelas yang didefinisikan tadi.

Sequence Labeling Application

- Information Extraction
- Speech Recognition
- POS Tagging
- Shallow Parsing
- Handwriting Recognition
- Protein Secondary Structure Prediction
- Video Analysis
- Facial Expression Dynamic Modelling

Ini beberapa problem di NLP yang memanfaatkan sequence labelling. Baca detailnya di slide untuk contoh IE, SRL, dan Bioinformatic.

Sulit untuk membuat hasil dari sequence labeling menjadi deterministik (jika x maka y) karena kemungkinannya banyak dan ketidakpastian kategori (kelas) mana yang paling tepat untuk token (fitur/ kata) tertentu. Karena itu kita gunakan **probabilistic sequence model**. Model ini memanfaatkan peluang statistik untuk menentukan mana kategori yang paling tepat.

Standard yang digunakan biasanya :
1. hidden markov model (HMM)
2. conditional random field (CRF)

Yang kita pelajari pada kelas ini hanya yang HMM saja. CRF dapat dipelajari masing masing terlebih dahulu. Nanti bisa kita diskusikan juga apabila ada yang ditanyakan.

 # Hidden Markov Model

HMM adalah sebuah finite state machine (automata) dengan transisi state probabilistik. Kalau diingat kembali bahasan tentang automata, transisi dilakukan pada saat state tertentu bertemu dengan karakter tertentu, maka statenya berubah ke state lain. Ini deterministik. NFA (non deterministik finite automata) mengubah state ke beberapa kemungkinan state yang ada. Di sini statenya bisa berubah ke beberapa state lain (seperti NFA) tetapi dengan peluang masing-masing.

HMM menggunakan asumsi markov, yaitu state berikutnya hanya tergantung dari state saat ini dan independen terhadap apa yang terjadi sebelumnya. Markov model orde ke berapa kah ini? Bisa dilihat kembali ke penjelasan tentang markov model di slide tentang n-gram.

Disebut **hidden** markov model karena ada state machine yang tersembunyi.

Kita definisikan simbol-simbol yang akan kita gunakan :
x = observasi , contohnya kata kata yang muncul dalam satu rangkaian kalimat, atau gerakan gerakan yang ada dalam video. Kita beri index dari 1 sampai T (banyaknya token yang muncul dalam barisan tersebut)

y = label , contohnya POS Tag untuk kalimat, atau set kategori (“berdiri”, “duduk”, “tiduran”) untuk video. Masing masing berkorespondensi dengan x sesuai indexnya. y1 adalah label dari x1, dst.

Y = himpunan label , sekumpulan label yang diperbolehkan, contohnya tagset. Banyaknya label harus berhingga.

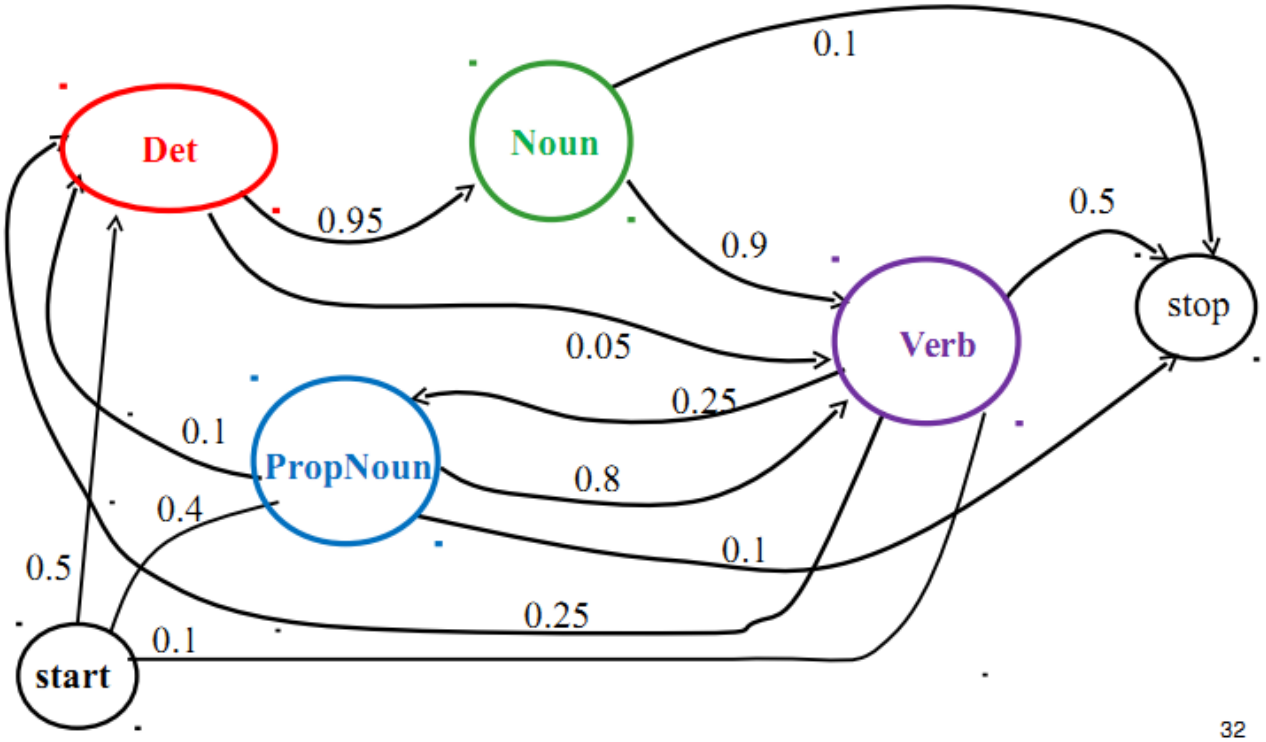
Kita memanfaatkan markov assumption, jadi peluang label sebuah token hanya tergantung dengan sebuah label token sebelumnya.

$$\text{Markov assumption } p(y_t|y_1,y_2,\ldots,y_{t-2},y_{t-1}) = p(y_t|y_{t-1})$$

Transisi yang terjadi adalah labelnya dulu (dari y1 ke y2 ke y3, dst) baru setelah itu dimunculkan observasi x-nya. Observasi x dihidden sementara dari transisi statenya.

Kita butuh 3 buah parameter untuk model HMM ini :
- probabilitas transisi state
- probabilitas kemunculan observasi dari state tertentu
- probabilitas mulai dari state tertentu (initial state) dan berakhir di state tertentu (final state)

Parameter tersebut dapat kita tuliskan dalam bentuk tabel ataupun diagram (state transition diagram atau lattice diagram).



Pada halaman 32, ada contoh model sederhana untuk bahasa inggris. Dalam hal ini label yang ada hanya 4 macam, yaitu determiner, proper noun, noun, dan verb.

Y = { determiner, proper noun, noun, verb }

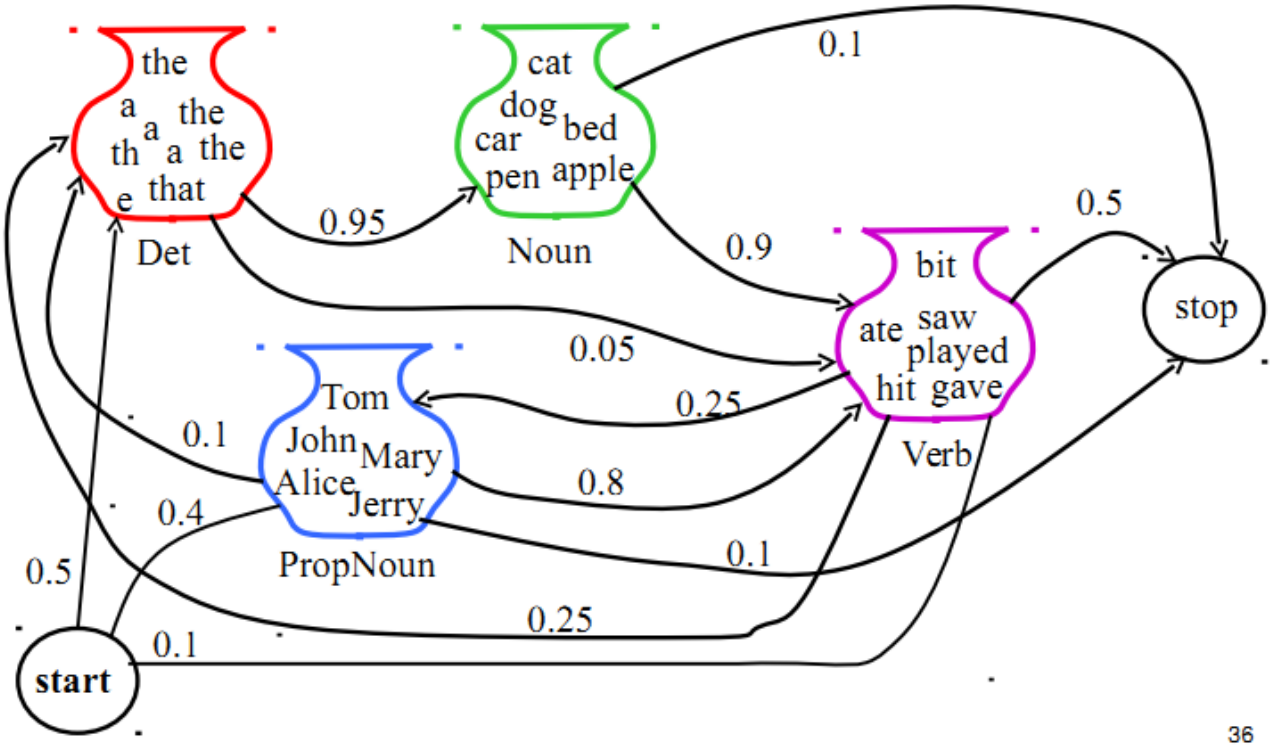
peluang masing masing state dapat terlihat dari gambar tersebut (meskipun beberapa panah hilang). Contohnya peluang state verb (**V**) berpindah ke determiner (**D**) sebesar 0.25 . Peluang dari determiner ke noun (**N**) sebesar 0.95, dst. Di sini labelnya saya singkat jadi D, P, N, dan V untuk memudahkan penulisan ya.

Dari diagram di atas dapat diubah ke dalam tabel untuk menuliskan parameter transisi statenya. Tabel transisi state dapat digabungkan dengan peluang initial state dan final state. Bisa juga dipisah. Kalau dipisah berarti initial state probabilitynya dapat dituliskan seperti ini :

D = 0.5, P = 0.4, V= 0.1

Kalau digabung berarti ada state baru (selain 4 state yang didefinisikan tadi), yaitu initial state dan final state) dalam tabelnya.

Bagaimana dengan parameter nomor 2 tentang peluang kemunculan observasi tertentu? Dalam diagram di atas tidak terlihat. Tetapi, setelah statenya berpindah, kita dapat cari peluang kemunculan observasi/token/kata (x) pada state/kategori/kelas (y) tersebut.



Pada halaman 36 ada gambar representasi dari observasi. Digambarkan berbentuk Guci / Urn (analogi urn dan ball). Maksudnya adalah. Pada saat kita masuk ke state Determiner misalnya, maka observasi yang mungkin adalah the, a, dan that. Peluang masing masing (dilihat dari banyaknya token yang tertulis) adalah the = 0.5, a = 0.375, dan that = 0.125.

Kita dapat gunakan untuk classification dan generation, sama seperti finite state automata yang lainnya.

 # Generation

Misalnya kita mau menggenerate (membuat) sebuah kalimat dari model tersebut. Kita bisa memanfaatkan angka random untuk memindahkan dari satu state ke state lainnya. Dalam slide dicontohkan :
- kita berangkat dari state awal (initial state/start)
- dari angka random yang didapat, ternyata jatuh ke state P (y1 = P)
- di state P tersebut dilihat observasi yang mungkin (random lagi), ternyata jatuh ke kata John (x1 = John)
- dari state y1, kita random lagi transisi statenya, masuk ke V (y2 = V)
- di state V, ternyata observasinya adalah kata bit (x2 = bit)
- dan seterusnya, hingga akhirnya kita dapatkan kata John bit the apple.

Kalau kita random lagi, bisa jadi kita akan mendapatkan kalimat yang lain seperti :
- Mary played the dog hit
- Jerry that car hit the bed saw Tom

dll.

Baik itu kalimat yang masuk akal maupun yang tidak. Tetapi secara struktur akan lebih baik daripada model ngram yang kita gunakan (dengan contoh kalimat yang digenerate di akhir slide praktikum). Karena pada model ini kita sudah mempertimbangkan peran dari kata yang terlibat.

Pada model ngram, mungkin saja kalimat dimulai dengan salah satu kata pada kelas Noun, apabila peluang kemunculannya di korpus memang tinggi. Padahal Noun tidak bisa berdiri sendiri tanpa diawali determiner. Tapi pada model HMM ini, hal tersebut sudah diperbaiki.

Penggambarannya bisa juga dengan graphical model yang memanjang ke kanan ataupun trellis. Penggambaran dengan trellis lebih informatif karena terlihat arah perubahan statenya, meskipun observasi tidak ditampilkan dalam gambar tersebut. Mana yang digunakan disesuaikan saja dengan kebutuhannya untuk visualisasi. Yang jadi concern utama kita adalah modelnya dulu (dalam bentuk tabel atau matriks)

Definisi Formal HMM

Formal Definition of an HMM

- A set of $N+2$ states $S=\{s_0,s_1,s_2, \ldots s_N, s_F\}$
  - Distinguished start state: s_0
  - Distinguished final state: s_F
- A set of M possible observations $V=\{v_1,v_2 \ldots v_M\}$
- A state transition probability distribution $A=\{a_{ij}\}$

$$a_{ij}=P(q_{t+1}=s_j|q_t=s_i) \qquad 1\le i,j\le N \text{ and } i=0, j=F$$

$$\sum_{j=1}^{N} a_{ij}+a_{iF}=1 \quad 0\le i\le N$$

- Observation probability distribution for each state j $B=\{b_j(k)\}$

$$b_j(k)=P(v_k \text{ at } t|q_t=s_j) \qquad 1\le j\le N \quad 1\le k\le M$$

- Total parameter set $\lambda=\{A,B\}$

Ini definisi formal HMM yang tertulis di halaman 49. pada model ini, parameter untuk peluang state transition, peluang initial statenya, dan peluang final statenya digabung dalam satu parameter A, yang kita sebut sebagai **transtition probability distribution**.

Maksud dari persamaan di atas adalah :
- simbol aij berarti peluang perubahan dari state i ke state j.
- jumlah peluang transisi dari satu state ke state lain adalah 1

Probabilitas kemunculan observasi pada state tertentu dituliskan sebagai parameter B. kita sebut sebagai **observation probability distribution**.

Maksud persamaannya adalah :
- simbol bj(k) artinya peluang kemunculan observasi vk di state j. t digunakan sebagai index untuk counter urutan kemunculan (kata pertama, kedua, dst)

Parameter A dan B adalah himpunan parameter (parameter set), atau kita sebut sebagai modelnya, disimbolkan dengan huruf lambda.

Algoritma untuk menggenerate kata sebenarnya sama seperti yang tadi kita lakukan di awal, hanya saja di sini diformalkan penulisannya. Untuk menggenerate barisan observasi O, kita lakukan

$$O = o_1 \; o_2 \; \ldots \; o_T$$

Set initial state $q_1 = s_0$
For $t = 1$ to T
    Transit to another state $q_{t+1} = s_j$ based on transition distribution a_{ij} for state q_t
    Pick an observation $o_t = v_k$ based on being in state q_t using distribution $b_{qt}(k)$

Kita bisa melakukan beberapa hal dengan HMM :
- **Observation likelyhood**. Jika kita tahu model (lambda) dan observasinya (O), kita bisa hitung peluang observasi tersebut. Misalnya peluang kemunculan kalimat tertentu.
- **Most likely state sequence**. Jika kita tahu model (lambda) dan observasinya (O), kita bisa menentukan rangkaian state (Q) yang memiliki peluang tertinggi untuk menghasilkan O. Misalnya melabeli kalimat dengan POS Tag.
- **Maximum likelihood training**. Jika kita punya sejumlah observasi dan labelnya (state), kita bisa membuat sebuah model lambda yang memaksimalkan peluang untuk menghasilkan observasi tersebut. Ini adalah machine learning.

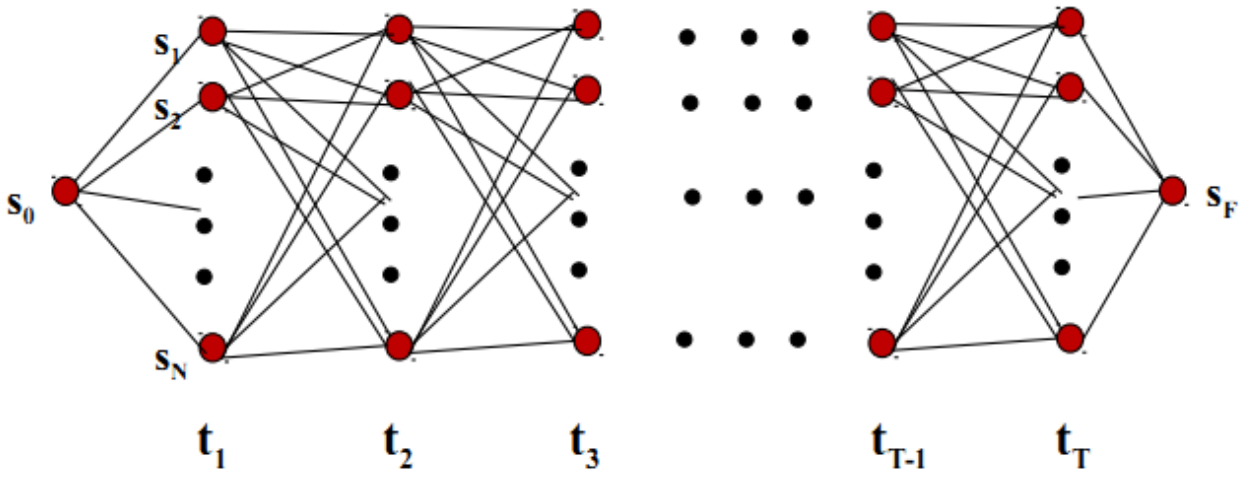
Penjelasan tentang ketiga hal tersebut dapat dilihat lebih jauh di slide. Silahkan ditanyakan jika ada yang kurang jelas.

 # Observation Likelihood

dari definisinya, kita akan mencari peluang dari observasi (O) pada model tertentu (lambda). Tetapi observasi tidak bisa kita hitung peluangnya tanpa kita tahu statenya terlebih dahulu. Sedangkan pada kasus ini kita tidak tahu state (Q) dari observasi tersebut apa. Kita tahu kalimatnya, tapi kita tidak tahu labelnya.

Maka kita perlu cari tahu peluang O di semua state Q yang mungkin. Ini tidak praktikal karena makin panjang kalimatnya, variasi state yang ada juga akan semakin banyak, eksponensial. Jadi kita perlu mencari cara lain yang bisa lebih terstruktur dan cepat. Pencarian dengan cara ini disebut dengan naive solution. Ordonya O(TN^T)

Kita bisa gunakan algoritma maju (forward) yang kompleksitasnya lebih rendah, yaitu berorde O(TN^2). Algoritma ini memanfaatkan dinamic programming. Jadi solusinya kita cari tahap per tahap dan kita bisa langsung mengeliminasi solusi yang tidak mungkin sehingga pencariannya tidak terlalu banyak.



Untuk visualisasi algoritma ini saya gunakan trellis. Kita akan berangkat dari sisi kiri, mengerjakan satu demi satu kolom hingga sampai di ujung sebelah kanan (akhir kalimat).

Simbol alpha digunakan sebagai nilai probabilitas pada titik tertentu. Alpha t (j) artinya peluang mesin kita berada pada state j setelah t observasi. Dapat dihitung dengan menghitung peluang dari t buah observasi pertama yang mengarah ke state j.

$$\alpha_t(j) = P(o_1, o_2, \ldots o_t, \quad q_t = s_j | \lambda)$$

Untuk mendapatkan nilai tersebut, kita perlu hitung terlebih dahulu peluang di waktu sebelumnya (t-1). Algoritmanya secara lengkap dapat dituliskan sbb.

- Initialization

$$\alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \le j \le N$$

- Recursion

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad 1 \le j \le N, \quad 1 < t \le T$$

- Termination

$$P(O|\lambda) = \alpha_{T+1}(s_F) = \sum_{i=1}^{N} \alpha_T(i)$$

Ada 3 tahapan : initialization, recursion, dan termination.

Pada tahap initialization kita menghitung peluang observasi kata pertama (o1) di setiap state yang mungkin (1<= j <=N). a0j dan bj(o1) didapat dari tabel A dan B.

tahap recursion kita menghitung peluang t buah observasi pertama yang mengarah ke kata ke t. Dari arah manapun. Karena itu peluangnya kita jumlahkan semuanya.

Tahap termination digunakan untuk mengakhiri kalimatnya (peluang state sebelum terakhir berpindah ke final state). Tidak ada observasinya lagi karena ini sudah di akhir kalimat.

Contoh penggunaan ada di slide halaman 62 s.d. 68.