FÖRELÄSNING I SPRÅKTEKNOLOGI

STATISTISKA METODER 2

- MARKOVPROCESSER
- HMM – GÖMDA MARKOVMODELLER
- INFORMATIONSTEORI (ENTROPI)
- TILLÄMPNING: ORDPREDIKTION
  - UTVÄRDERING (PERPLEXITET)
  - MARKOVMODELL
  - ORDFREKVENSER, ORDKLASSTATISTIK
  - HEURISTISKA FÖRBÄTTRINGAR
  - LAGRING AV DATA

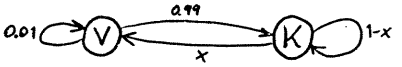
MARKOVPROCESS

EXEMPEL: STOKASTISK PROCESS $X_1, X_2, \ldots$ SOM LÄSER EN BOKSTAV I TAGET FRÅN ETT ORD OCH AVGÖR OM DEN ÄR EN VOKAL ELLER KONSONANT.

FREKVENSRÄKNING I KORPUS GER:

$P[X_i = \text{VOKAL}] = 0.36$ OCH $P[X_i = \text{VOKAL} \mid X_{i-1} \neq \text{VOKAL}] = 0.01$

MODELLERA MED MARKOVKEDJA:



$$(P_{ij}) = \begin{pmatrix} 0.01 & 0.99 \\ x & 1-x \end{pmatrix}$$

$P[X_i = V] = 0.36$, $P[X_i = K] = 0.64$

$$(0.36 \;\; 0.64)\begin{pmatrix} 0.01 & 0.99 \\ x & 1-x \end{pmatrix} = (0.36 \;\; 0.64)$$

$$\Rightarrow 0.36 \cdot 0.01 + 0.64x = 0.36 \Rightarrow x = \frac{0.36(1-0.01)}{0.64} \approx 0.56$$

EN MARKOVPROCESS HAR INGET MINNE.

MODELLEN ÄR DÄRFÖR LÄMPLIG BARA OM SANNOLIKHETEN FÖR VOKAL INTE PÅVERKAS AV MER ÄN FÖREGÅENDE BOKSTAV, OCH DET GÖR DEN!

HMM – GÖMDA MARKOVMODELLER

GIVET:
- TILLSTÅND $q_1, \ldots, q_n$
- ÖVERGÅNGSSANNOLIKHETSMATRIS $P = (p_{ij})$ DÄR $p_{ij} = P[\text{ÖVERGÅNG FRÅN } q_i \text{ TILL } q_j]$
- STARTSANNOLIKHETER $v = (v_1, \ldots, v_n)$
- SIGNALER $\sigma_1, \ldots, \sigma_m$
- SIGNALMATRIS $A = (a_{ij})$ DÄR $a_{ij} = P[\sigma_j \text{ GENERERAS} \mid \text{TILLSTÅNDET ÄR } q_i]$

I EN HMM ÄR TILLSTÅNDEN OSYNLIGA OCH BARA SIGNALERNA KAN OBSERVERAS.

EXEMPEL: TAGGNING AV EN MENING

TILLSTÅND ↔ TAGGAR, SIGNALER ↔ ORD

PROBLEM: HITTA TROLIGASTE PROCESSEN (TILLSTÅNDSFÖLJDEN) SOM GENERERAR MENINGEN

Informationsteori

Informationen hos en signal σ som har sannolikheten p är $I_\sigma = \log_2 \frac{1}{p}$

Exempel:

$P[\sigma] = 0.25 \Rightarrow I_\sigma = \log_2 \frac{1}{0.25} = \log_2 4 = 2$

$P[\sigma] = 1 \Rightarrow I_\sigma = \log_2 \frac{1}{1} = \log_2 1 = 0$

Informationsmängden mäts i **bitar**.

Vanlig signal $\Rightarrow$ liten informationsmängd

Entropi

Entropin är väntevärdet för informationen:

$$H[X] = E\left[\log_2 \frac{1}{p}\right] = \sum p_i \log_2 \frac{1}{p_i} = -\sum p_i \log p_i$$

där X är en stokastisk variabel, $P[X = \sigma_i] = p_i$

Exempel: Låt $Y = \begin{cases} a & \text{med sannolikhet } 0.5 \\ b & \text{—"— } 0.25 \\ c & \text{—"— } 0.25 \end{cases}$

$$H[Y] = 0.5 \cdot \log_2 \frac{1}{0.5} + 0.25 \cdot \log_2 \frac{1}{0.25} + 0.25 \cdot \log_2 \frac{1}{0.25} =$$
$$= 0.5 + 0.25 \cdot 2 + 0.25 \cdot 2 = 1.5$$

Entropin är högst då alla n signaler är lika sannolika:

$$H_{max} = \sum_{i=1}^{n} \frac{1}{n} \log_2 \frac{1}{1/n} = n \cdot \frac{1}{n} \log_2 n = \log_2 n$$

Exempel:

$n = 3 \Rightarrow H_{max} = \log_2 3 \approx 1.58$

Redundansen är den relativa skillnaden mellan H_{max} och H: $R = \frac{H_{max} - H}{H_{max}} = 1 - \frac{H}{H_{max}}$

Exempel: $R = 1 - \frac{1.5}{1.58} \approx 0.051 = 5.1\%$

Exempel: Entropin för svenska

Entropin för svenska bokstäver är

$$H = \sum_{\alpha=A}^{\ddot{o}} P[\alpha] \cdot \log_2 \frac{1}{P[\alpha]} = 4.34$$

Om alla bokstäver var lika vanliga vore entropin

$H_{max} = \log_2 29 \approx 4.86$

$\Rightarrow$ redundansen är $1 - \frac{4.34}{4.86} \approx 11\%$

Försök ta hänsyn till att bokstäverna inte förekommer helt slumpmässigt i svenska ord.

Entropin för svenska ord är

$$H = \sum_{x \in \{\text{svenska ord}\}} P[x] \cdot \log_2 \frac{1}{P[x]} = 8.3$$

Medelordlängden = 5.3 bokstäver

Entropin utslagen på bokstäverna är $\frac{8.3}{5.3} \approx 1.6$

Redundansen blir då $1 - \frac{1.6}{4.86} \approx 67\%$

Ordprediktion

Exjobb av Johan Carlberger 1997

Problem: Givet ord $w_1, w_2, \ldots, w_{i-1}$

Gissa ordet w_i

Idé: Hitta X som maximerar $P[X=w_i | w_1 w_2 \ldots w_{i-1}]$

Utvärdering av en ordprediktionsmetod:

Informationsteoretisk:

Perplexiteten för X är

$$Perp[X] = 2^{H[X]}$$

motsvarar antalet ord prediktorn har att välja mellan.

Om alla N ord är lika sannolika blir

$$Perp[X] = 2^{\log_2 N} = N.$$

Mål: minimera perplexiteten.

Praktisk:

Mät antalet tangenttryckningar som behövs för att generera en testtext med ordprediktorn.

Mål: maximera antalet sparade tangenttryckningar

> Testa inte ett program på samma text som det har tränats på!

Statistikinsamling

Samla in följande statistik från en stor korpus:

ordfrekvenser	$C_o(w)$
ordbigramsfrekvenser	$C_{obi}(w_1, w_2)$
taggfrekvenser	$C_t(t)$
taggbigramsfrekvenser	$C_{tbi}(t_1, t_2)$
taggtrigramsfrekvenser	$C_{ttri}(t_1, t_2, t_3)$
ordtaggparsfrekvenser	$C_{ot}(w, t)$
antal ord i korpusen	C

Beräkna sedan (och lagra):

$$f_o(w) = \frac{C_o(w)}{C}, \quad f_{obi}(w_2|w_1) = \frac{C_{obi}(w_1, w_2)}{C_o(w_1)}$$

$$f_t(t) = \frac{C_t(t)}{C}, \quad f_{tbi}(t_2|t_1) = \frac{C_{tbi}(t_1, t_2)}{C_t(t_1)}, \quad f_{ttri}(t_3|t_1, t_2) = \frac{C_{ttri}(t_1, t_2, t_3)}{C_{tbi}(t_1, t_2)}$$

$$f_{ot}(t|w) = \frac{C_{ot}(w, t)}{C_o(w)}$$

Två Markovmodeller

Taggmodellen

Andra ordningens markovmodell för taggar.

Tillstånden beskrivs av par av taggar (t_1, t_2).

$$P_{(t_1,t_2),(t_2,t_3)} = P[t_3 \mid t_1, t_2] = q_0 f_{ttri}(t_3|t_1,t_2) + q_1 f_{tbi}(t_3|t_2) + q_2 f_t(t_3)$$

där q_0, q_1, q_2 är lämpligt valda konstanter

Ordmodellen

Första ordningens markovmodell för ord.

Tillstånden beskrivs av orden i korpusen.

$$P_{w_1,w_2} = P[w_2|w_1] =$$
$$= r_0 f_{obi}(w_2|w_1) + r_1 f_o(w_2) + \left(r_2 f_{obi}(w_2|w_1) + r_3 f_o(w_2)\right) \cdot p_t(w_2)$$

där r_0, r_1, r_2, r_3 är lämpligt valda konstanter och

$$p_t(w_2) = f_{ot}(t_2|w_2) \cdot P[t_2|t_0, t_1]$$

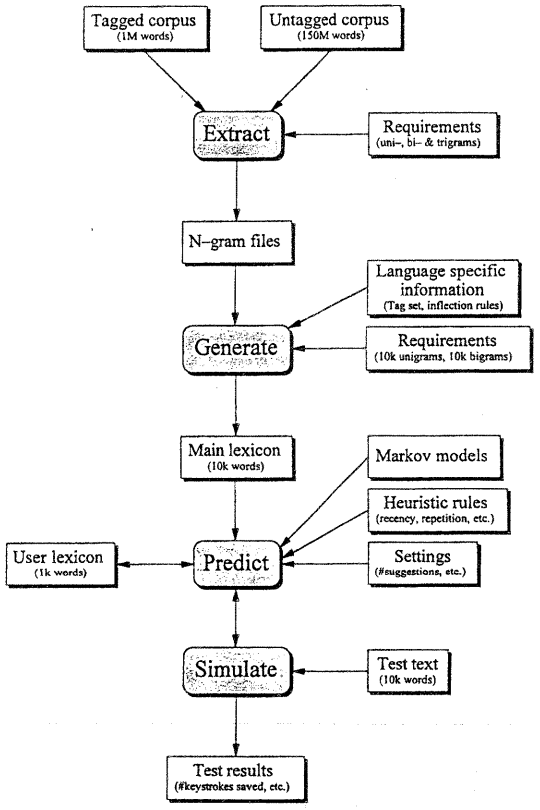


Figure 3. A system for word predictor construction.

LATMANNAHASHNING

HASHA BARA PÅ DOM TRE FÖRSTA BOKSTÄVERNA I SÖKNYCKELN. ANVÄND SEDAN BINÄRSÖKNING.

LÄMPLIGT FÖR SÖKNING MED FÅ DISKACCESSER I STOR TEXT NÄR INDEXET INTE KAN LIGGA I PRIMÄRMINNET.

EXEMPEL: INDEXERA STORT SVENSK-ENGELSKT LEXIKON.

GIVET: • STOR FIL L MED HELA LEXIKONTEXTEN.
• INDEX I DÄR VARJE RAD ÄR: SÖKORD POSITION I L

SKAPA SÖKINDEX: SORTERA I MED sort.
SKAPA EN INDEXARRAY A[abc] SOM FÖR VARJE abc ANGER VAR I I FÖRSTA ORDET SOM BÖRJAR SÅ FINNS.

FÖRBEREDELSE INFÖR SÖKNINGAR:
LÄS IN A FRÅN FIL. ÖPPNA FILERNA I OCH L.

SÖKNINGSALGORITM(w)=

```
wprefix ← FÖRSTA 3 BOKST. I w
i ← A[wprefix]
j ← A[wprefix+1]
WHILE j-i > 1000 DO
    m ← ⌊(i+j)/2⌋
    GÅ TILL POSITION m I I
    LÄS IN NÄSTA ORD FRÅN I I s
    IF s≤w THEN i ← m
      ELSE j ← m
GÅ TILL POSITION i I I
WHILE TRUE DO
    LÄS IN NÄSTA ORD FRÅN I I s
    IF s=w THEN
        LÄS IN POSITIONEN I x
        RETURN x
    IF s>w THEN
        RETURN NOTFOUND
```

Språkteknologi 2000-11-21
Viggo Kann, Nada

Viterbis algoritm

Anta att vi vet sannolikheten $P(x, y)$ för att ordklassen x ska följas av ordklassen y i en mening. Vi har också två speciella ordklasser *start* för meningsbörjan och *slut* för meningsslut. $P(start, x)$ är då sannolikheten för att en mening inleds med ordklassen x och $P(y, slut)$ är sannolikheten för att en mening avslutas med ordklassen y.

Anta också att vi har ett lexikon som talar om vilka ordklasser varje ord kan ha. Exempelvis säger lexikonet att boken bara kan vara substantiv men att gula både kan vara adjektiv och substantiv.

Den troligaste ordklasstaggningen av en mening är den taggning som maximerar produkten av dom ingående sannolikheterna.

Indata består av en mening med n ord. Det finns m stycken ordklasstaggar (inklusive *start* och *slut*).

Viterbis algoritm löser detta problem i linjär tid i antalet ord n med hjälp av algoritmkonstruktionsmetoden dynamisk programmering.

Låt $M[i, j]$ vara lösningen (maximala produktsannolikheten) för delproblemet att tagga meningen fram till ord i så att ord i får taggen j. $M[i, j]$ kan definieras rekursivt som

$$M[i,j] = \begin{cases} 1 & \text{om } i = 0 \text{ och } j = start, \\ 0 & \text{om } i = 0 \text{ och } j \neq start, \\ 0 & \text{om } 1 \leq i \leq n \text{ och } j \notin Lexikon(w[i]), \\ \max\limits_{1 \leq k \leq m} M[i-1,k] \cdot P[k,j] & \text{annars.} \end{cases}$$

Vilket k-värde som ger maximum i rekursionen lagras i $father[i, j]$ så att algoritmen ska kunna följa den bästa taggningen tillbaka från *slut* till *start*. Den bästa taggningen lagras sedan i $POS[1..n]$.

Indata: $P[1..m, 1..m]$, $w[1..n]$, $Lexikon$: ord → **set of integer**

```
for j ← 1 to m do M[0, j] ← 0
M[0, start] ← 1
for i ← 1 to n+1 do
    if i ≤ n then S ← Lexikon(w[i]) else S ← {slut}
    for j ← 1 to m do M[i, j] ← 0
    foreach j ∈ S do
        for k ← 1 to m do
            tmp ← M[i-1, k] · P[k, j]
            if tmp > M[i, j] then M[i, j] ← tmp; father[i, j] ← k
c ← slut
for i ← n downto 1 do
    POS[i] ← c ← father[i+1, c]
for i ← 1 to n do
    write w[i], POS[i]
```

I värsta fallet kan varje ord ha m ordklasser. Dom tre nästlade forslingorna går då $n + 1$, m respektive m varv, vilket ger tidskomplexiteten $O(nm^2)$.