Föreläsning 13

Algoritmkonstruktion 4: mer dynamisk programmering

Ordklasstaggning

Ett viktigt problem inom språkteknologin är problemet att märka (tagga) varje ord i en mening med dess ordklass. Detta kallas för ordklasstaggning. En enkel men väl fungerande metod för detta bygger på statistik om hur ordklasser förekommer efter varandra. Mer om taggning och dess tillämpningar ges i den valfria kursen DH2418 Språkteknologi.

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. $P(\textit{start}, x)$ är då sannolikheten för att en mening inleds med ordklassen x och $P(y, \textit{slut})$ 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 *kundkrets* bara kan vara substantiv men att *kära* både kan vara adjektiv och verb.

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

Anta att indata består av en mening med n ord och att det finns m stycken ordklasstaggar (inklusive *start* och *slut*). Anta vidare att tiden för en uppslagning i lexikonet är $O(1)$.

Uppgiften är att konstruera en effektiv algoritm (tiden ska vara polynomisk i n och m) som hittar och skriver ut den sannolikaste ordklasstaggningen av en mening. Om flera lösningar är precis lika sannolika så räcker det att algoritmen ger en av dom. Analysera också algoritmens tidskomplexitet.

Exempel

Mening: **Vi tappar aldrig vår kära kundkrets.**

Lexikon: Vi – pronomen; tappar – substantiv, verb; aldrig – adverb; vår – substantiv, pronomen; kära – adjektiv, verb; kundkrets – substantiv.

Sannolikhetsmatrisen $P(\textit{ordklass1,ordklass2})$, där raderna motsvarar den första ordklassen och kolumnerna den andra ordklassen:

P	**adv**	**adj**	**konj**	**subs**	**pron**	**prep**	**räkn**	**verb**	**slut**
start	0.131	0.043	0.085	0.262	0.299	0.115	0.032	0.033	0.000
adverb	0.144	0.116	0.057	0.093	0.108	0.112	0.015	0.302	0.052
adjektiv	0.020	0.047	0.086	0.657	0.009	0.085	0.005	0.029	0.063
konjunktion	0.096	0.078	0.021	0.259	0.227	0.051	0.013	0.253	0.002
substantiv	0.083	0.025	0.126	0.102	0.013	0.232	0.016	0.210	0.195
pronomen	0.106	0.216	0.024	0.248	0.025	0.062	0.019	0.269	0.030
preposition	0.034	0.093	0.064	0.488	0.242	0.030	0.030	0.004	0.015
räkneord	0.023	0.054	0.054	0.607	0.009	0.054	0.015	0.075	0.109
verb	0.148	0.070	0.061	0.140	0.238	0.186	0.009	0.117	0.032

Den korrekta ordklasstaggningen start-pronomen-verb-adverb-pronomen-adjektiv-substantiv-slut har sannolikhetsprodukten $0,299 \cdot 0,269 \cdot 0,148 \cdot 0,108 \cdot 0,216 \cdot 0,657 \cdot 0,195 = 3,6 \cdot 10^{-5}$ medan den felaktiga ordklasstaggningen start-pronomen-verb-adverb-pronomen-verb-substantiv-slut bara har produkten $0,299 \cdot 0,269 \cdot 0,148 \cdot 0,108 \cdot 0,269 \cdot 0,140 \cdot 0,195 = 0,9 \cdot 10^{-5}$.

Lösning till Ordklasstaggning
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

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

Vilket k-värde som ger maximum i rekursionen lagras i $\mathit{father}[i,j]$ 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 $O(nm^2)$. □
