CMPT-882: Statistical Learning of Natural Language

Lecture #4

Anoop Sarkar
anoop@cs.sfu.ca
http://www.sfu.ca/~anoop

Mailing List

- `cmpt-882-fall-2002@sfu.ca`
  - Subscribe using online services: `http://www.sfu.ca/onlineserv.htm`
  - For course web page updates
  - Also a forum for questions you might have: remember to use `fraser.sfu.ca` to post to the list

- Hidden Markov Models (HMMs)
  - Distinction between Markov Chains and HMMs
  - Application of HMMs to Part of Speech tagging
  - Algorithms for HMMs: Viterbi decoding and the Forward-Backward Algorithm (Baum-Welch)

Trigram Models

- $P(w_{1,2,\ldots,m}) =$
  $P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_{1,2}) \times P(w_4 \mid w_{1,3}) \times \ldots$
- $P(w_k \mid w_{1,k-1}) = P(w_k \mid w_{k-n-1,k-1})$
- hence the name n-grams; $n = 2$: bigrams, $n = 3$: trigrams
- also possible to define variable length n-grams
- Best performing language model when used with *smoothing*

Trigram Models and Markov Chains

	aa	ab	ba	bb
aa	a:p(a| aa)	b:p(b| aa)		
ab			a:p(a| ab)	b:p(b| ab)
ba	a:p(a| ba)	b:p(b| ba)		
bb			a:p(a| bb)	b:p(b| bb)

Trigram Models and Smoothing

- Let vocabulary be V
- worst case size of a trigram model: $\mid V \mid^{?}$; for a given corpus: $\mid V \mid^{?}$
- $P(w_n \mid w_{n-2,n-1}) =$
  $\lambda_1(w_{n-2,n-1})P_e(w_n) + \lambda_2(w_{n-2,n-1})P_e(w_n \mid w_{n-1}) +$
  $\lambda_3(w_{n-2,n-1})P_e(w_n \mid w_{n-2,n-1})$
- $\lambda_1(w_{n-2,n-1}) + \lambda_2(w_{n-2,n-1}) + \lambda_3(w_{n-2,n-1}) = 1$

Trigram Models

- Using cross-entropy as a model evaluator
- Removing the ”correct” model from the equation was possible when the model was *ergodic*
- Samples (when large enough) correspond to the true probabilities in the model
- Also, the model has to be *stationary*: for our purposes, it means that the values for a given context do not vary

Trigram Models: A Generative Model

```
In_IN 1994_CD ,_, Hartnett_NNP said_VBD

THE_DT BONEYARD_NNP Northrop_NNP  Grumman_NNP ’s_POS modest_JJ
flight_NN museum_NN occupies_VBZ a_DT corner_NN of_IN one_CD of_IN
its_PRP$ power-seat_NN adjusters_NNS ,_, door_NN trim_JJ now_RB
made_VBN in_IN South_NNP Korea_NNP ’s_POS antiquated_JJ coal-fired_JJ
power_NN plant_NN in_IN Canada_NNP ,_, to_TO a_DT 11.9_CD million_CD
mark_NN investment_NN in_IN Samsung_NNP ’s_POS Sachon_NNP plant_NN
in_IN Taiwan_NNP as_IN part_NN of_IN a_DT steam_NN turbine_NN ,_,
a_DT new_JJ high-yielding_JJ rice_NN plant_NN was_VBD reorganized_VBN
into_IN a_DT big_JJ expansion_NN of_IN a_DT fuel-fabrication_NN
plant_NN near_IN Nagoya_NNP in_IN Aichi_NNP Prefecture_NNP

From_IN October_NNP ,_, when_WRB they_PRP
do_VBP not_RB need_VB it_PRP
```

Part of Speech Tagging using Trigram Models

- $P(w_{1,n}) = \sum_{t_{1,n+1}} P(w_{1,n}, t_{1,n+1})$
- $\underset{t_{1,n+1}}{\text{arg max}}\ P(t_{1,n+1} \mid w_{1,n}) = \underset{t_{1,n+1}}{\text{arg max}}\ \frac{P(w_{1,n},\, t_{1,n+1})}{P(w_{1,n})}$
- $\underset{t_{1,n+1}}{\text{arg max}}\ P(w_{1,n},\ t_{1,n+1})$

Hidden Markov Models

- A set of states, each state is hidden, i.e. not visible in the training data. The number of states is arbitrary and set in advance (see paper by A. Stolcke and S. M. Omohundro)

- Assume each state is connected to every other state

- Numerous applications in speech, language processing, cryptography, modelling continuous fns.

Hidden Markov Models

- At each time tick t, we traverse from one state to another and emit an output symbol
- $P(s^i \xrightarrow{w} s^j) = P(S_{t+1} = s^j, W_t = w, \mid S_t = s^i)$
- $P(s^i \xrightarrow{w} s^j) = P(w, s^j \mid s^i) = P(w \mid s^i)P(s^j \mid s^i)$
  — the Markov assumption
- transition probability: $P(s_j \mid s_i)$
- output probability: $P(w \mid s_i)$

Hidden Markov Models

- $P(w_{1,n}) = \sum_{s_{1,n+1}} P(w_{1,n},\ s_{1,n+1})$
- $\sum_{s_{1,n+1}} \prod_{i=1}^{n} P(w_i,\ s_{i+1} \mid s_i)$
- $\sum_{s_{1,n+1}} \prod_{i=1}^{n} P(s^i \overset{w_i}{\rightarrow} s^{i+1})$
- Best path (Viterbi algorithm): $\underset{s_{1,n+1}}{\arg\max} \prod_{i=1}^{n} P(s^i \overset{w_i}{\rightarrow} s^{i+1})$

Forward-Backward Algorithm: Baum-Welch

- How can we compute transition and output probabilities, when the state sequences are ”hidden”
- Intuitively, probability of taking a transition from a state s^i to s^j is

$$P_e(s^i \xrightarrow{w} s^j) = \frac{C(s^i \xrightarrow{w} s^j)}{\sum_{k,w'} C(s^i \xrightarrow{w'} s^k)}$$

- So once we have a method for computing $C(s^i \xrightarrow{w} s^j)$ we can re-estimate each transition probability
- Note that number of times you take a transition also depends on the initial setting of the transition probability

Forward-Backward Algorithm

- Hence, the probability of a transition is the number of times it was used in a path (state sequence) times the probability of that path (for all paths)

- $C(s^i \xrightarrow{w} s^j) = \sum_{s_{1,n+1}} P(s_{1,n+1} \mid w_{1,n}) \cdot \eta(s^i \xrightarrow{w} s^j, s_{1,n+1}, w_{1,n})$

- η counts number of times $s^i \xrightarrow{w} s^j$ appears in the path $s_{1,n+1}$ when the output is $w_{1,n}$

Forward-Backward Algorithm

$$
\begin{aligned}
C(s^i \xrightarrow{w} s^j) &= \sum_{s_{1,n+1}} P(s_{1,n+1} \mid w_{1,n}) \cdot \eta(s^i \xrightarrow{w} s^j, s_{1,n+1}, w_{1,n}) \\
C(s^i \xrightarrow{w} s^j) &= \frac{1}{P(w_{1,n})} \times \\
&\quad \sum_{t=1}^{n} \sum_{s_{1,n+1}} P(s_{1,n+1}, w_{1,n}, S_t = s^i, S_{t+1} = s^j, W_t = w) \\
P(w_{1,n}) &= \sum_{s_{1,n+1}} P(s_{1,n+1}, w_{1,n})
\end{aligned}
$$

Forward-Backward Algorithm

$$
\begin{aligned}
C(s^i \xrightarrow{w} s^j) &= \frac{1}{P(w_{1,n})} \times \\
&\quad \sum_{t=1}^{n} P(w_{1,n}, S_t = s^i, S_{t+1} = s^j, W_t = w) \\
&= \frac{1}{P(w_{1,n})} \times \\
&\quad \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i, S_{t+1} = s^j, W_t = w, w_{t+1,n}) \\
&= \frac{1}{P(w_{1,n})} \times \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i) \cdot \\
&\quad P(S_{t+1} = s^j, W_t = w \mid w_{1,t-1}, S_t = s^i) \cdot \\
&\quad P(w_{t+1,n} \mid w_{1,t}, S_t = s^i, S_{t+1} = s^j)
\end{aligned}
$$

Forward-Backward Algorithm

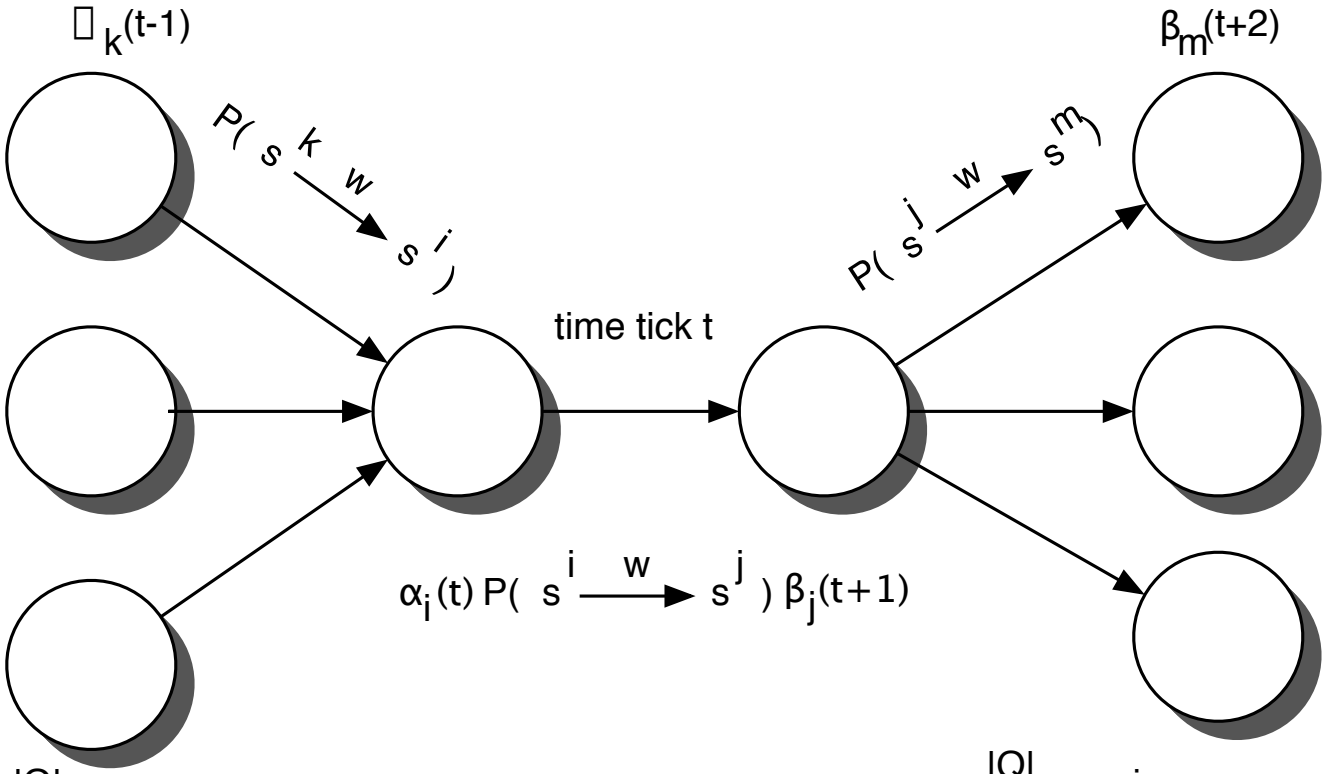
$$
\begin{aligned}
C(s^i \xrightarrow{w} s^j) &= \frac{1}{P(w_{1,n})} \times \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i) \cdot \\
&\quad P(S_{t+1} = s^j, W_t = w \mid w_{1,t-1}, S_t = s^i) \cdot \\
&\quad P(w_{t+1,n} \mid w_{1,t}, S_t = s^i, S_{t+1} = s^j) \\
&= \frac{1}{P(w_{1,n})} \times \sum_{t=1}^{n} P(w_{1,t-1}, S_t = s^i) \cdot \\
&\quad P(S_{t+1} = s^j, W_t = w \mid S_t = s^i) \cdot \\
&\quad P(w_{t+1,n} \mid S_{t+1} = s^j) \\
&= \frac{1}{P(w_{1,n})} \times \sum_{t=1}^{n} \alpha_i(t) \cdot P(s^i \xrightarrow{w} s^j) \cdot \beta_j(t+1)
\end{aligned}
$$

Forward-Backward Algorithm

$$
\begin{aligned}
\alpha_i(t) &= P(w_{1,t-1}, S_t = s^i) \\
\alpha_{s_1}(1) &= 1.0 \\
\alpha_j(t+1) &= P(w_{1,t}, S_{t+1} = s^j) \\
&= \sum_i P(w_{1,t}, S_t = s^i, S_{t+1} = s^j) \\
&= \sum_i P(w_{1,t-1}, S_t = s^i) \cdot \\
&\quad P(W_t = w, S_{t+1} = s^j \mid \cancel{w_{1,t-1}}, S_t = s^i) \\
&= \sum_i \alpha_i(t) \cdot P(s^i \xrightarrow{w} s^j)
\end{aligned}
$$

Forward-Backward Algorithm

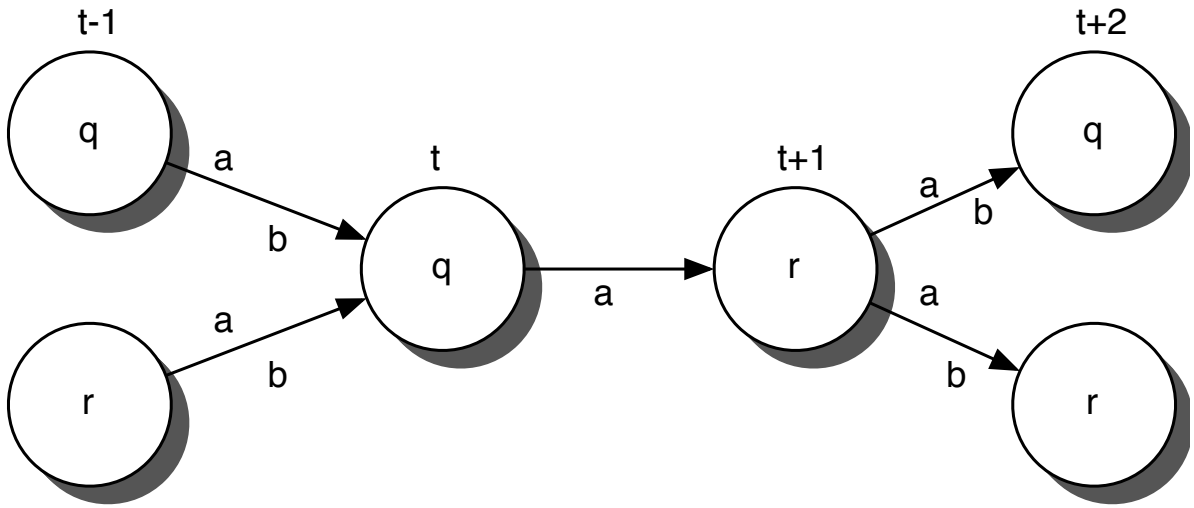
$$
\begin{aligned}
\beta_i(t) &= P(w_{t,n} \mid S_t = s^i) \\
\beta_i(n+1) &= P(\epsilon \mid S_{n+1} = s^i) = 1.0 \\
\beta_i(t-1) &= P(w_{t-1,n} \mid S_{t-1} = s^i) \\
&= \sum_j P(W_{t-1} = w, S_t = s^j \mid S_{t-1} = s^i) \cdot \\
&\quad P(w_{t,n} \mid \cancel{W_{t-1} = w}, S_t = s^j, \cancel{S_{t-1} = s^i}) \\
&= \sum_j P(W_{t-1} = w, S_t = s^j \mid S_{t-1} = s^i) \cdot \\
&\quad P(w_{t,n} \mid S_t = s^j) \\
&= \sum_j P(s^i \xrightarrow{w} s^j) \cdot \beta_j(t)
\end{aligned}
$$



$$\alpha_i(t) = \sum_{k=1}^{|Q|} \square_k(t-1)\, P(\, s^k \xrightarrow{w} s^i\,)$$

$$\beta_j(t+1) = \sum_{m=1}^{|Q|} P(\, s^j \xrightarrow{w} s^m)\, \beta_m(t+2)$$

$$C(\, s^i \xrightarrow{w} s^j\,) = \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} \alpha_i(t)\, P(\, s^i \xrightarrow{w} s^j\,)\, \beta_j(t+1)$$



$$
\begin{aligned}
\alpha_q(t) &= \alpha_q(t-1)P(a,q \mid q) + \alpha_q(t-1)P(b,q \mid q) + \\
&\quad \alpha_r(t-1)P(a,q \mid r) + \alpha_r(t-1)P(b,q \mid r) \\
\beta_r(t+1) &= P(a,q \mid r)\beta_q(t+2) + P(b,q \mid r)\beta_q(t+2) + \\
&\quad P(a,r \mid r)\beta_r(t+2) + P(b,r \mid r)\beta_r(t+2) \\
C(q \xrightarrow{a} r) &= \frac{1}{P(w_{1,n})} \sum_{t=1}^{n} \alpha_q(t)P(a,r \mid q)\beta_r(t+1)
\end{aligned}
$$

Forward-Backward Algorithm

- Set initial transition probabilities to appropriate values (usually random)
- Compute $C(s^i \xrightarrow{w} s^j)$ for each state i and then $P_e(s^i \xrightarrow{w} s^j) = \frac{C(s^i \xrightarrow{w} s^j)}{\sum_{k,w'} C(s^i \xrightarrow{w'} s^k)}$
- Compute likelihood $P(w_{1,n}) = \beta_{s^1}(1)$; iterate until likelihood is maximized (or entropy is minimized)
- Here we considered the case for one training sentence $w_{1,n}$. For a whole corpus, $\prod_k P(w^k_{1,n})$ is the likelihood of the entire corpus with k sentences

Forward-Backward Algorithm

- Likelihood is guaranteed to be non-decreasing due to the theorem by Baum (generalized by Dempster, Laird and Rubin)

  Maximum-likelihood from incomplete data via the EM algorithm. A. P. Dempster, N. M. Laird and D. B. Rubin. *Journal of the Royal Statistics Society*, 1977, 39:1, pp. 1–38

- Forward-Backward Algorithm is an example of purely unsupervised learning

- Applications of the Forward-Backward Algorithm

Next Time

- Using the Forward-Backward Algorithm to decrease human supervision

- Does Baum-Welch Re-estimation help taggers? (1994). David Elworthy. *Proceedings of 4th ACL Conf on ANLP*, Stuttgart. pp. 53-58.