COMP90042 LECTURE 7

SEQUENCE TAGGING: UNSUPERVISED HMMS

HMMS FOR POS TAGGING - RECAP

- Previous lecture: **supervised** POS tagging using HMMs
  - Assume we have a corpus with annotated POS tags (for instance, Penn Treebank)
  - Learning via MLE (counting)
  - At test/prediction time, uses Viterbi algorithm to find most probably sequence

HMMS FOR POS TAGGING - RECAP

- Previous lecture: **supervised** POS tagging using HMMs
  - Assume we have a corpus with annotated POS tags (for instance, Penn Treebank)
  - Learning via MLE (counting)
  - At test/prediction time, uses Viterbi algorithm to find most probably sequence
- Sometimes we do not have such a corpus
  - Different domains (Twitter, TED talks)
  - Different languages, especially low-resource ones
- Solution: **unsupervised** learning

UNSUPERVISED HMMS

- Suppose you have a corpus of (unannotated) tweets.
- Simple idea: start with an HMM with **random** parameters (emission and transition matrices)
  - Using Viterbi, tag the data using this model.
  - Pretend this is training data. Obtain counts via MLE.
  - Repeat

UNSUPERVISED HMMS

- Suppose you have a corpus of (unannotated) tweets.
- Simple idea: start with an HMM with **random** parameters (emission and transition matrices)
  - Using Viterbi, tag the data using this model.
  - Pretend this is training data. Obtain counts via MLE.
  - Repeat
- Rationale: the model will learn the actual POS distribution based on the word patterns seen in the data
  - This is a simple version of the **Expectation-Maximisation (EM)** algorithm

EXPECTATION MAXIMIZATION

- Janet/NNP will/MD back/VB the/DT bill/NN

EXPECTATION MAXIMIZATION

- ~~Janet/NNP will/MD back/VB the/DT bill/NN~~
- Janet/NNP will/MD back/**RB** the/DT bill/NN
- $P(back|\boldsymbol{t}) = \begin{bmatrix} P(back|RB) \\ P(back|VB) \\ P(back|DT) \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{c(will,RB)}{c(RB)} \\ \frac{c(will,VB)}{c(VB)} \\ \frac{c(will,DT)}{c(DT)} \\ \vdots \end{bmatrix} = \begin{bmatrix} {}^{1}/_{2} \\ {}^{0}/_{1} \\ {}^{0}/_{1} \\ \vdots \end{bmatrix}$

EXPECTATION MAXIMIZATION

- This approach, sometimes referred as **hard EM**, can perform well depending on the setting.

EXPECTATION MAXIMIZATION

- This approach, sometimes referred as **hard EM**, can perform well depending on the setting.
- However, it is still too naïve, as it does not take into account **distributions** on each word and each tag transition.

EXPECTATION MAXIMIZATION

- This approach, sometimes referred as **hard EM**, can perform well depending on the setting.
- However, it is still too naïve, as it does not take into account **distributions** on each word and each tag transition.
- A better approach would be to incorporate such distributions by:
  - Obtaining **marginal** emission and transition distributions.
  - Using **weighted** (expected) counts to train via MLE.

VITERBI EXAMPLE REVISITED

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	**2.0e-15**
RB	0	0	5.3e-11	0	0
DT	0	0	0	1.8e-12	0

VITERBI EXAMPLE REVISITED

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	2.5e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.0e-20
JJ	0	0	5.1e-15	5.2e-16	0
NN	0	1.034e-10	5.4e-15	5.9e-18	**2.0e-15**
RB	0	0	**5.3e-11**	0	0
DT	0	0	0	1.8e-12	0

MLE WITH EXPECTED COUNTS

- $P(back|\boldsymbol{t}) = \begin{bmatrix} P(back|RB) \\ P(back|VB) \\ P(back|DT) \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{\hat{c}(back,RB)}{\hat{c}(RB)} \\ \frac{\hat{c}(back,VB)}{\hat{c}(VB)} \\ \frac{\hat{c}(back,DT)}{\hat{c}(DT)} \\ \vdots \end{bmatrix} = ?$

MLE WITH EXPECTED COUNTS

- $P(back|\boldsymbol{t}) = \begin{bmatrix} P(back|RB) \\ P(back|VB) \\ P(back|DT) \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{\hat{c}(back,RB)}{\hat{c}(RB)} \\ \frac{\hat{c}(back,VB)}{\hat{c}(VB)} \\ \frac{\hat{c}(back,DT)}{\hat{c}(DT)} \\ \vdots \end{bmatrix} = ?$

- $\hat{c}(back, RB) = \sum_{\boldsymbol{w}} \sum_{i=1;i=back}^{l\boldsymbol{w}} \hat{P}(t_i = RB|\boldsymbol{w})$
  - "i = back" means we iterate only when the observed word is "back"

MLE WITH EXPECTED COUNTS

- $P(back|\boldsymbol{t}) = \begin{bmatrix} P(back|RB) \\ P(back|VB) \\ P(back|DT) \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{\hat{c}(back,RB)}{\hat{c}(RB)} \\ \frac{\hat{c}(back,VB)}{\hat{c}(VB)} \\ \frac{\hat{c}(back,DT)}{\hat{c}(DT)} \\ \vdots \end{bmatrix} = ?$

- $\hat{c}(back, RB) = \sum_{\boldsymbol{w}} \sum_{i=1;i=back}^{l\boldsymbol{w}} \hat{P}(t_i = RB|\boldsymbol{w})$
  - "i = back" means we iterate only when the observed word is "back"

- $\hat{c}(RB) = \sum_{\boldsymbol{w}} \sum_{i=1}^{l\boldsymbol{w}} \hat{P}(t_i = RB|\boldsymbol{w})$

MLE WITH EXPECTED COUNTS

- $P(back|\boldsymbol{t}) = \begin{bmatrix} P(back|RB) \\ P(back|VB) \\ P(back|DT) \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{\hat{c}(back,RB)}{\hat{c}(RB)} \\ \frac{\hat{c}(back,VB)}{\hat{c}(VB)} \\ \frac{\hat{c}(back,DT)}{\hat{c}(DT)} \\ \vdots \end{bmatrix} = ?$
- $\hat{c}(back, RB) = \sum_{\boldsymbol{w}} \sum_{i=1;i=back}^{l\boldsymbol{w}} \hat{P}(t_i = RB|\boldsymbol{w})$
  - "i = back" means we iterate only when the observed word is "back"
- $\hat{c}(RB) = \sum_{\boldsymbol{w}} \sum_{i=1}^{l\boldsymbol{w}} \hat{P}(t_i = RB|\boldsymbol{w})$
- $\hat{P}(t_i = RB|\boldsymbol{w}) = \frac{\hat{P}(t_i=RB,\boldsymbol{w})}{\hat{P}(\boldsymbol{w})}$

MLE WITH EXPECTED COUNTS

- Key quantities for emission probabilities:
  - $\hat{P}(\boldsymbol{w})$
  - $\hat{P}(t_i = RB, \boldsymbol{w})$

MLE WITH EXPECTED COUNTS

- Key quantities for emission probabilities:
  - $\hat{P}(\boldsymbol{w})$
  - $\hat{P}(t_i = RB, \boldsymbol{w})$
- Let's start with $\hat{P}(\boldsymbol{w})$
  - $\hat{P}(\boldsymbol{w}) = \sum_{\boldsymbol{t}} \hat{P}(\boldsymbol{w}, \boldsymbol{t}) = \sum_{\boldsymbol{t}} \hat{P}(\boldsymbol{w}|\boldsymbol{t}) \hat{P}(\boldsymbol{t})$
  - Exponential number of tag sequences: can we use DP again?

THE FORWARD ALGORITHM

- Exactly like Viterbi, but summing scores instead of taking the max.
  - Also no backpointers since the goal is not prediction.

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP					
MD					
VB					
JJ					
NN					
RB					
DT					

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	P(Janet|NNP) * P(NNP|<s>)				
MD	P(Janet|MD) * P(MD|<s>)				
VB	…				
JJ	…				
NN	…				
RB	…				
DT	…				

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06				
MD	0				
VB	0				
JJ	0				
NN	0				
RB	0				
DT	0				

THE FORWARD ALGORITHM

	Janet	**will**	**back**	**the**	**bill**
NNP	8.8544e-06	P(will|NNP) * P(NNP|t_{Janet}) * a(t_{Janet}|Janet)			
MD	0	…			
VB	0	…			
JJ	0	…			
NN	0	…			
RB	0	…			
DT	0	…			

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	P(will|NNP) * P(NNP|t_{Janet}) * a(t_{Janet}|Janet)			
MD	0	...			
VB	0				
JJ	0	...			
NN	0	...			
RB	0	...			
DT	0	...			

Calculate this for all tags, take the **sum**.

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	0			
MD	0	3.004e-8			
VB	0	2.231e-13			
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

THE FORWARD ALGORITHM

	Janet	**will**	**back**	**the**	**bill**
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	P(back|VB) * P(VB|t_{will}) * s(t_{will}|will)		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	**MD: 1.6e-11** **VB: 7.5e-19** **NN: 9.7e-17**		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	1.6e-11		
JJ	0	0			
NN	0	1.034e-10			
RB	0	0			
DT	0	0			

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0		
MD	0	3.004e-8	0		
VB	0	2.231e-13	1.6e-11		
JJ	0	0	5.42e-15		
NN	0	1.034e-10	8.17e-15		
RB	0	0	5.33e-11		
DT	0	0	0		

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	4.21e-17	0
MD	0	3.004e-8	0	0	0
VB	0	2.231e-13	1.6e-11	0	1.74e-20
JJ	0	0	5.42e-15	6.53e-16	0
NN	0	1.034e-10	8.17e-15	9.76e-18	3.44e-15
RB	0	0	5.33e-11	0	0
DT	0	0	0	3.1e-12	0

THE FORWARD ALGORITHM

	Janet	will	back	the	bill
NNP	8.8544e-06	0	0	4.21e-17	0
					0
					1.74e-20
					0
					3.44e-15
					0
DT	0	0	0	3.1e-12	0

- Final probability also needs to take into account the end state “</s>”
- P(**w**) = α(bill,VB) * P(<end>|VB) + α(bill,NN) * P(<end>|NN)
- P(**w**) = 1.74e-20 * 0.0004 + 3.44e-15 * 0.237
- P(**w**) = 8.15e-16

THE FORWARD PROCEDURE

- Pseudocode

```
alpha = np.zeros(M, T)
for t in range(T):
    alpha[1, t] = pi[t] * O[w[1], t]

for i in range(2, M):
    for t_i in range(T):
        for t_last in range(T):    # t_last means t_{i-1}
            s = alpha[i-1, t_last] * A[t_last, t_i] * B[w[i], t_i]
            alpha[i,t_i] += s

total = np.sum(alpha[M-1,:] * A[:,<end>])
Return total, alpha
```

MLE WITH EXPECTED COUNTS

- We got $\hat{P}(\boldsymbol{w})$ using the forward algorithm. Now we need $\hat{P}(t_i = RB, \boldsymbol{w})$.

MLE WITH EXPECTED COUNTS

- We got $\hat{P}(\boldsymbol{w})$ using the forward algorithm. Now we need $\hat{P}(t_i = RB, \boldsymbol{w})$.
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB, \boldsymbol{w}_{i+1:l_{\boldsymbol{w}}})$

MLE WITH EXPECTED COUNTS

- We got $\hat{P}(\boldsymbol{w})$ using the forward algorithm. Now we need $\hat{P}(t_i = RB, \boldsymbol{w})$.
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB, \boldsymbol{w}_{i+1:l_w})$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}\left(\boldsymbol{w}_{i+1:l_w} | \boldsymbol{w}_{1:i}, t_i = RB\right)$

MLE WITH EXPECTED COUNTS

- We got $\hat{P}(\boldsymbol{w})$ using the forward algorithm. Now we need $\hat{P}(t_i = RB, \boldsymbol{w})$.
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB, \boldsymbol{w}_{i+1:l_w})$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}\left(\boldsymbol{w}_{i+1:l_w} | \boldsymbol{w}_{1:i}, t_i = RB\right)$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}(\boldsymbol{w}_{i+1:l_w} | t_i = RB)$

MLE WITH EXPECTED COUNTS

- We got $\hat{P}(\boldsymbol{w})$ using the forward algorithm. Now we need $\hat{P}(t_i = RB, \boldsymbol{w})$.
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB, \boldsymbol{w}_{i+1:l_w})$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}\left(\boldsymbol{w}_{i+1:l_w} | \boldsymbol{w}_{1:i}, t_i = RB\right)$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}(\boldsymbol{w}_{i+1:l_w} | t_i = RB)$
- $\hat{P}(\boldsymbol{w}_{1:i}, t_i = RB) = \alpha(i, RB)$
  - We got this from the forward algorithm!

MLE WITH EXPECTED COUNTS

- We got $\hat{P}(\boldsymbol{w})$ using the forward algorithm. Now we need $\hat{P}(t_i = RB, \boldsymbol{w})$.
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB, \boldsymbol{w}_{i+1:l_w})$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}\left(\boldsymbol{w}_{i+1:l_w} | \boldsymbol{w}_{1:i}, t_i = RB\right)$
  - $\hat{P}(t_i = RB, \boldsymbol{w}) = \hat{P}(\boldsymbol{w}_{1:i}, t_i = RB)\ \hat{P}(\boldsymbol{w}_{i+1:l_w} | t_i = RB)$
- $\hat{P}(\boldsymbol{w}_{1:i}, t_i = RB) = \alpha(i, RB)$
  - We got this from the forward algorithm!
- $\hat{P}\left(\boldsymbol{w}_{i+1:l_w} | t_i = RB\right) = \beta(i, RB)$
  - We will get these through the **backward** algorithm.

THE BACKWARD ALGORITHM

	Janet	will	back	the	bill
NNP					
MD					
VB					
JJ					
NN					
RB					
DT					

THE BACKWARD ALGORITHM

	Janet	will	back	the	bill
NNP					$P(NNP \vert <end>) * \beta(?,<end>)$

- $\beta(i, < end >) = \hat{P}\left(\boldsymbol{w}_{i+1:l_w} \middle| t_i =< end >\right)$
- $\beta(6, < end >) = \hat{P}(\boldsymbol{w}_{6:5} | t_6 =< end >)$
- This is the probability of an “empty” event given the end-of-sentence tag. In other words, the probability of not emitting anything given the end-of-sentence tag. This is always true so the result is 1 for all cells. This forms the base case for the recursion.

THE BACKWARD ALGORITHM

	Janet	will	back	the	bill
NNP				$\sum$ P(bill|t_{bill}) * P(NNP|t_{bill}) * ß(bill,t_{bill})	P(NNP|<end>)
MD				$\sum$ P(bill|t_{bill}) * P(MD|t_{bill}) * ß(bill,t_{bill})	P(MD|<end>)
VB				…	P(VB|<end>)
JJ				…	P(JJ|<end>)
NN				…	P(NN|<end>)
RB				…	P(RB|<end>)
DT				…	P(DT|<end>)

THE BACKWARD PROCEDURE

- Pseudocode

```
beta = np.zeros(M, T)
for t in range(T):
    beta[M, t] = A[:,<end>]    # initialise the last column

for i in range(M-1, 0, -1):   # range in reverse
    for t_i in range(T):
        for t_last in range(T):    # t_last means t_{i-1}
            s = beta[i+1, t_last] * A[t_i, t_last] * B[w[i+1], t_last]
            beta[i,t_i] += s

total = np.sum(beta[1,:] * A[<start>,:] * B[w[1],:])
Return total, beta
```

OBTAINING THE PROBABILITIES

- After running forward and backward, we obtain two matrices containing **α's** and **β's**,
- $\hat{P}(\boldsymbol{w}) = \sum_t \alpha(l_{\boldsymbol{w}}, t) * \hat{P}(< end > |t)$
- $\hat{P}(t_i = RB, \boldsymbol{w}) = \alpha(i, RB) * \beta(i, RB)$

OBTAINING THE PROBABILITIES

- After running forward and backward, we obtain two matrices containing **α's** and **β's**,
- $\hat{P}(\boldsymbol{w}) = \sum_t \alpha(l_{\boldsymbol{w}}, t) * \hat{P}(< end > |t)$
- $\hat{P}(t_i = RB, \boldsymbol{w}) = \alpha(i, RB) * \beta(i, RB)$
- $\hat{c}(back, RB) = \sum_{\boldsymbol{w}} \sum_{i=1;i=back}^{l_W} \frac{\alpha(i,RB) * \beta(i,RB)}{\sum_t \alpha(l_{\boldsymbol{w}},t) * \hat{P}(<end>|t)}$
- $\hat{c}(RB) = \sum_{\boldsymbol{w}} \sum_{i=1}^{l_W} \frac{\alpha(i,RB) * \beta(i,RB)}{\sum_t \alpha(l_{\boldsymbol{w}},t) * \hat{P}(<end>|t)}$
- $P(back|RB) = \frac{\hat{c}(back,RB)}{\hat{c}(RB)}$

TRANSITION PROBABILITIES

- Transition probabilities are also updated using expected counts.
  - We can use the values from forward-backward as well.
- $\xi_i(NN, DT) = \frac{\alpha(i,DT) * a[DT,NN] * b(w_{i+1},NN) * \beta(i+1,NN)}{P(\boldsymbol{w})}$
- $\hat{c}(NN, DT) = \sum_{w} \sum_{i=1}^{l_w} \xi_i(NN, DT)$
- $\hat{c}(DT) = \sum_{w} \sum_{i=1}^{l_w} \sum_{t} \xi_i(t, DT)$
- $P(NN|DT) = \frac{\hat{c}(NN,DT)}{\hat{c}(DT)}$

EM - FINAL ALGORITHM

- Initialise emission and transition matrices
- **E-step:**
  - Run forward-backward, obtaining **α's** and **β's**
- **M-step:**
  - Update emission and transition matrices using the expected counts.

EM – IMPORTANT POINTS

- In our example, we initialised the parameters (matrices) at random. In practice initialisation matters so care must be taken.
  - Take values from a previous known tagger.
  - Use dictionaries to initialise some values ("the" is never a verb).

EM – IMPORTANT POINTS

- In our example, we initialised the parameters (matrices) at random. In practice initialisation matters so care must be taken.
  - Take values from a previous known tagger.
  - Use dictionaries to initialise some values ("the" is never a verb).
- Forward scores can be **conditional** instead of joint. Mitigates underflow.

EM – IMPORTANT POINTS

- In our example, we initialised the parameters (matrices) at random. In practice initialisation matters so care must be taken.
  - Take values from a previous known tagger.
  - Use dictionaries to initialise some values ("the" is never a verb).
- Forward scores can be **conditional** instead of joint. Mitigates underflow.
- How to evaluate?
  - Need some annotated data for that.

A FINAL WORD

- Unsupervised learning is key to generalise sequence tagging to other domains and languages
- HMMs can easily do that using EM.
- Some annotated data is always necessary for evaluation.

READINGS

- JM3 Ch 9.3, 9.5
- [Optional] Rabiner's HMM tutorial, for more details
  - http://tinyurl.com/2hqaf8