ILLINOIS INSTITUTE OF TECHNOLOGY

Maximum entropy models and structured prediction

CS-585

Natural Language Processing

Derrick Higgins

HMMs, MEMMs and CRFs

- Hidden Markov Models
  - Generative sequence labeling models
- Maximum Entropy Markov Models
  - Like HMMs, but discriminative
  - Logistic regression
- Conditional Random Fields
  - Like MEMMs, but in structured prediction paradigm

MAXIMUM ENTROPY MARKOV MODELS

Maximum Entropy

- In NLP, logistic regression models are often called *maximum entropy* (or *maxent*) models
- Idea
  - when selecting a probability distribution to model observed data, we want the distribution to model the statistics of the data, but otherwise reserve judgement
  - Model should be as uncertain as possible, while still capturing the data
  - Uncertainty=entropy, so select the model with maximal entropy subject to data-fitting constraints

Maximum Entropy

- In a supervised learning context, the data fitting constraints have to do with the frequencies of feature functions $f_k(X, Y)$ in the training data:

$$\sum_n f_k(X_n, Y_n) P(X_n, Y_n) = c_k$$

- It can be shown that the distribution $P(Y_n|X_n)$ with maximum entropy subject to these constraints has the form

$$P(Y_n|X_n) = \frac{e^{\sum_k \lambda_k f_k(X_n, Y_n)}}{Z}$$

- Where Z is a normalizing constant
- Just logistic regression….

Maximum Entropy Markov Models (MEMMs)

- Cousins of HMMs
  - Still based on the Markov assumption:

$$P(T|W) = \prod_i P(T_i|T_{i-k..i-1}, W)$$

- Because they are discriminative, MEMMs cannot be trained in an unsupervised way like HMMs
  - HMMs are generative (model $P(W, T)$)
  - MEMMs are discriminative (model $P(T|W)$)
- Based on maximum entropy (logistic regression) models to predict tag for each word given local context
  - Can be trained using gradient descent or model-specific algorithms (GIS, IIS)

Maximum Entropy Markov Models (MEMMs)

Estimate conditional probability of tags given text

$$P_m(T|W) = \prod_i P(T_i|T_{i-k..i-1}, W)$$

(Markov assumption)

$$= \prod_i \frac{e^{\sum_k \lambda_k f_k(W,T_{i-k..i})}}{\sum_{t\in\mathfrak{T}} e^{\sum_k \lambda_k f_k(W,T_{i-k..i-1},t)}}$$

(maxent)

- Can be t
  - Either d likelihood (regula
- Similar dynamic programming tricks to HMMs for efficiently computing sum across all labelings

Locally normalized

MEMMs: POS feature functions

- Remember Transformation-Based Learning

The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word two before (after) is tagged **w**.

#	Change tags From	To	Condition	Example
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

MEMMs: POS feature functions

- Maxent feature functions

Condition	Features	
w_i is not rare	$w_i = X$	$\& \ t_i = T$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$	$\& \ t_i = T$
	X is suffix of w_i, $\|X\| \leq 4$	$\& \ t_i = T$
	w_i contains number	$\& \ t_i = T$
	w_i contains uppercase character	$\& \ t_i = T$
	w_i contains hyphen	$\& \ t_i = T$
$\forall \ w_i$	$t_{i-1} = X$	$\& \ t_i = T$
	$t_{i-2}t_{i-1} = XY$	$\& \ t_i = T$
	$w_{i-1} = X$	$\& \ t_i = T$
	$w_{i-2} = X$	$\& \ t_i = T$
	$w_{i+1} = X$	$\& \ t_i = T$
	$w_{i+2} = X$	$\& \ t_i = T$

Table 1: Features on the current history h_i

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. EMNLP.

MEMMs: POS feature functions

Word	the	stories	about	well-heeled	communities	and	developers
Tag	DT	NNS	IN	JJ	NNS	CC	NNS
Position	1	2	3	4	5	6	7

$w_i = \texttt{about}$	$\wedge\, t_i = \texttt{IN}$
$w_{i-1} = \texttt{stories}$	$\wedge\, t_i = \texttt{IN}$
$w_{i-2} = \texttt{the}$	$\wedge\, t_i = \texttt{IN}$
$w_{i+1} = \texttt{well-heeled}$	$\wedge\, t_i = \texttt{IN}$
$w_{i+2} = \texttt{communities}$	$\wedge\, t_i = \texttt{IN}$
$t_{i-1} = \texttt{NNS}$	$\wedge\, t_i = \texttt{IN}$
$t_{i-2}t_{i-1} = \texttt{DT NNS}$	$\wedge\, t_i = \texttt{IN}$

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. EMNLP.

MEMMs: POS feature functions

Word	the	stories	about	well-heeled	communities	and	developers
Tag	DT	NNS	IN	JJ	NNS	CC	NNS
Position	1	2	3	4	5	6	7

$prefix(w_i) = \texttt{w}$	$\wedge\, t_i = \texttt{JJ}$
$prefix(w_i) = \texttt{we}$	$\wedge\, t_i = \texttt{JJ}$
$prefix(w_i) = \texttt{wel}$	$\wedge\, t_i = \texttt{JJ}$
$prefix(w_i) = \texttt{well}$	$\wedge\, t_i = \texttt{JJ}$
$suffix(w_i) = \texttt{d}$	$\wedge\, t_i = \texttt{JJ}$
$suffix(w_i) = \texttt{ed}$	$\wedge\, t_i = \texttt{JJ}$
$suffix(w_i) = \texttt{led}$	$\wedge\, t_i = \texttt{JJ}$
$suffix(w_i) = \texttt{eled}$	$\wedge\, t_i = \texttt{JJ}$
w_i contains hyphen	$\wedge\, t_i = \texttt{JJ}$

$w_{i-1} = \texttt{about}$	$\wedge\, t_i = \texttt{JJ}$
$w_{i-2} = \texttt{stories}$	$\wedge\, t_i = \texttt{JJ}$
$w_{i+1} = \texttt{communities}$	$\wedge\, t_i = \texttt{JJ}$
$w_{i+2} = \texttt{and}$	$\wedge\, t_i = \texttt{JJ}$
$t_{i-1} = \texttt{IN}$	$\wedge\, t_i = \texttt{JJ}$
$t_{i-2}t_{i-1} = \texttt{NNS IN}$	$\wedge\, t_i = \texttt{JJ}$

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. EMNLP.

Viterbi Tagging for MEMMs

Most probable tag sequence given text:

$$T^* = \underset{T}{\operatorname{argmax}} P_m(T|W)$$

$$= \underset{T}{\operatorname{argmax}} \prod_i P(T_i | T_{i-k..i-1}, W)$$

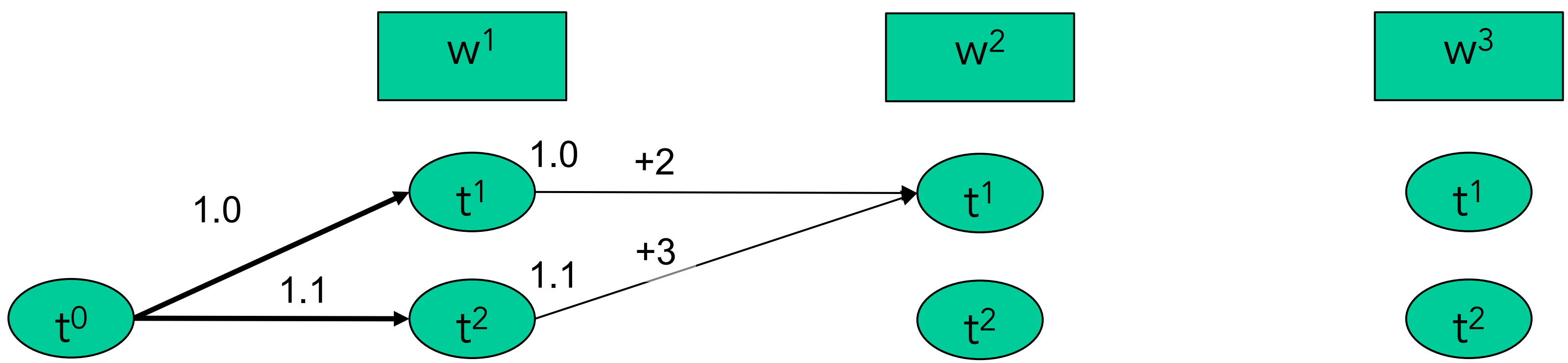
(Markov assumption)

$$= \underset{T}{\operatorname{argmax}} \prod_i \frac{e^{\sum_k \lambda_k f_k(W, T_{i-k..i})}}{\sum_{t \in \mathfrak{T}} e^{\sum_k \lambda_k f_k(W, T_{i-k..i-1}, t)}}$$

(maxent)

$$= \underset{T}{\operatorname{argmax}} \sum_i \left(\sum_k \lambda_k f_k(W, T_{i-k..i}) - \log \sum_{t \in \mathfrak{T}} e^{\sum_k \lambda_k f_k(W, T_{i-k..i-1}, t)} \right)$$

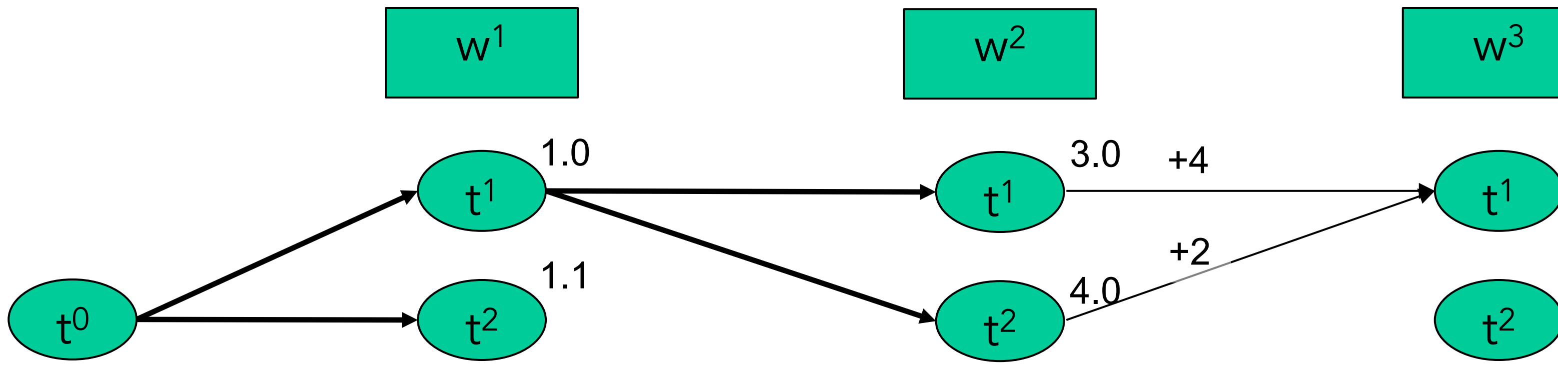
Viterbi algorithm



$$-\log P(t_i | t_{i-1}, w_i, w_{i-1})$$

t_{i-1}	t^0		t^1									t^2								
w_i	w^1	w^2	w^1			w^2			w^3			w^1			w^2			w^3		
w_{i-1}			w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3
$t_i=t^1$	1.0	1.1	2	4	2	2	2	3	2	4	5	2	2	2	3	2	4	5	2	4
$t_i=t^2$	1.1	2.1	3	5	3	3	2	4	2	4	3	3	3	2	4	2	4	3	4	5

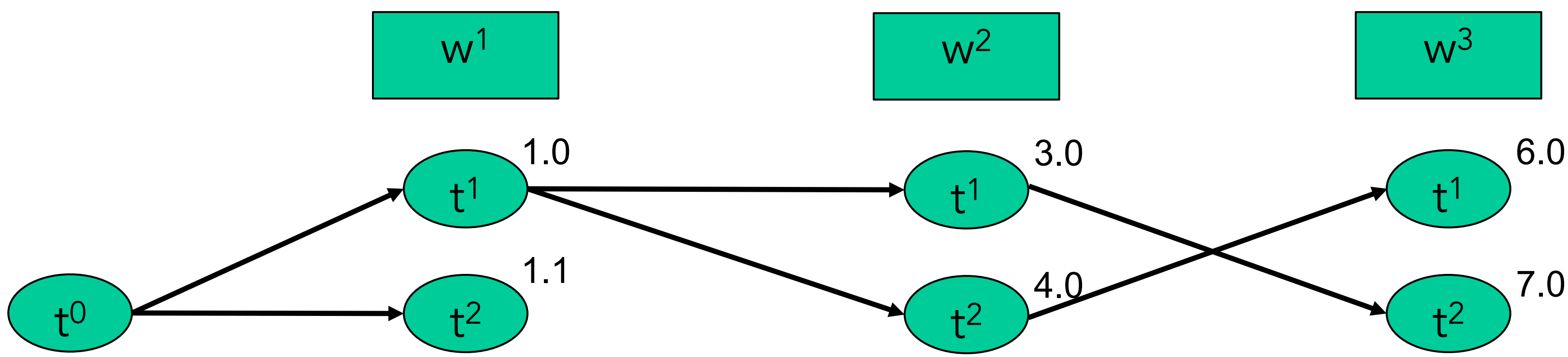
Viterbi algorithm



$$-\log P(t_i | t_{i-1}, w_i, w_{i-1})$$

t_{i-1}	t^0		t^1									t^2								
w_i	w^1	w^2	w^1			w^2			w^3			w^1			w^2			w^3		
w_{i-1}			w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3
$t_i=t^1$	1.0	1.1	2	4	2	2	2	3	2	4	5	2	2	2	3	2	4	5	2	4
$t_i=t^2$	1.1	2.1	3	5	3	3	2	4	2	4	3	3	3	2	4	2	4	3	4	5

Viterbi algorithm



$$-\log P(t_i | t_{i-1}, w_i, w_{i-1})$$

t_{i-1}	t^0		t^1									t^2								
w_i	w^1	w^2	w^1			w^2			w^3			w^1			w^2			w^3		
w_{i-1}			w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3
$t_i=t^1$	1.0	1.1	2	4	2	2	2	3	2	4	5	2	2	2	3	2	4	5	2	4
$t_i=t^2$	1.1	2.1	3	5	3	3	2	4	2	4	3	3	3	2	4	2	4	3	4	5

Viterbi Algorithm

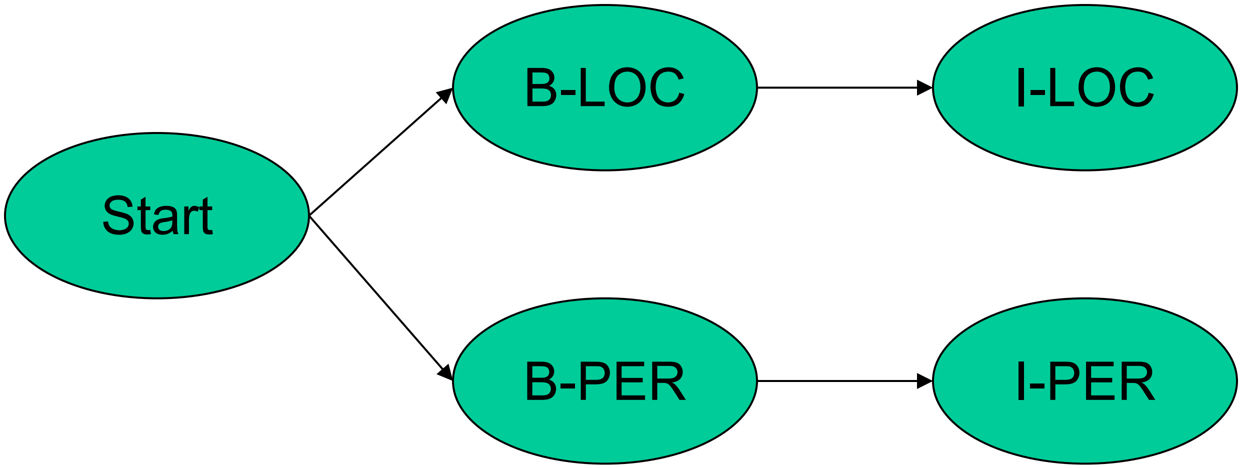
```
D(0, START) = 0
for each tag t != START do:
  D(0, t) = -∞
for i ← 1 to N do:
  for each tag t^j do:
     D(i, t^j) ← max_k (D(i-1,t^k)
                          + log P(t_i|t_{i-1}=t^k,W))
log P(T|W) = max_j D(N, t^j)
```

MEMM Limitations

- Locally normalized
  - MEMM assumes that the probability distribution over a tagging T can be factored into the product of conditional probabilities with limited history for each tag location in a sentence:
  - But this limits the flexibility of the model. Only paths from the same prior state can "compete" against one another for probability mass

MEMM Limitations: Label Bias

	PERSON count	LOCATION count
Harvey Ford	9	1
Harvey Park	1	9
Myrtle Ford	9	1
Myrtle Park	1	9



Example from Ralph Grishman

MEMM Limitations: Label Bias

	PERSON count	LOCATION count
Harvey Ford	9	1
Harvey Park	1	9
Myrtle Ford	9	1
Myrtle Park	1	9

Conditional probabilities:

$$P(t_i = \text{B}-\text{PER} | t_{i-1} = \text{Start}, w_i = \text{Harvey}) = 0.5$$
$$P(t_i = \text{B}-\text{LOC} | t_{i-1} = \text{Start}, w_i = \text{Harvey}) = 0.5$$
$$P(t_i = \text{B}-\text{PER} | t_{i-1} = \text{Start}, w_i = \text{Myrtle}) = 0.5$$
$$P(t_i = \text{B}-\text{LOC} | t_{i-1} = \text{Start}, w_i = \text{Myrtle}) = 0.5$$
$$P(t_i = \text{I}-\text{PER} | t_{i-1} = \text{B}-\text{PER}, w_i = \text{Ford}) = 1.0$$
$$P(t_i = \text{I}-\text{LOC} | t_{i-1} = \text{B}-\text{LOC}, w_i = \text{Ford}) = 1.0$$
$$P(t_i = \text{I}-\text{PER} | t_{i-1} = \text{B}-\text{PER}, w_i = \text{Park}) = 1.0$$
$$P(t_i = \text{I}-\text{LOC} | t_{i-1} = \text{B}-\text{LOC}, w_i = \text{Park}) = 1.0$$

Example from Ralph Grishman

MEMM Limitations: Label Bias

0.5	1.0	0.5	1.0
`B-LOC`	`E-LOC`	`B-PER`	`E-PER`
Harvey	Park	Harvey	Park

Conditional probabilities:

$$P(t_i = \text{B-PER} | t_{i-1} = \text{Start}, w_i = \text{Harvey}) = 0.5$$
$$P(t_i = \text{B-LOC} | t_{i-1} = \text{Start}, w_i = \text{Harvey}) = 0.5$$
$$P(t_i = \text{B-PER} | t_{i-1} = \text{Start}, w_i = \text{Myrtle}) = 0.5$$
$$P(t_i = \text{B-LOC} | t_{i-1} = \text{Start}, w_i = \text{Myrtle}) = 0.5$$
$$P(t_i = \text{I-PER} | t_{i-1} = \text{B-PER}, w_i = \text{Ford}) = 1.0$$
$$P(t_i = \text{I-LOC} | t_{i-1} = \text{B-LOC}, w_i = \text{Ford}) = 1.0$$
$$P(t_i = \text{I-PER} | t_{i-1} = \text{B-PER}, w_i = \text{Park}) = 1.0$$
$$P(t_i = \text{I-LOC} | t_{i-1} = \text{B-LOC}, w_i = \text{Park}) = 1.0$$

Really? But this was 9 times more common in the training data!

Example from Ralph Grishman

MEMM Limitations: Label Bias

- *Label bias* problem: Low-entropy states (from which following states are highly predictable) play an unreasonably strong role in determining label sequence
  - In an extreme case, causing the words to be irrelevant in determining the labels
- Due to local normalization of probabilities for each tag, instead of global normalization for entire tag sequence

CONDITIONAL RANDOM FIELDS

Structured prediction

- CRFs fall into a predictive modeling framework called *structured prediction*
  - When we have some complex structured object over which we want to make predictions…
    - pixels within an image, tag sequences or syntactic trees for a text
  - … we try to estimate a probability distribution over the whole output space, rather than distributions over subparts

Structured prediction

Structured prediction for sequence labeling: predict optimal tagging T^* using a scoring function over the output space:

$$\hat{T} = \operatorname*{argmax}_{T} \Psi(T, W)$$

In a probabilistic framework:

$$\hat{T} = \operatorname*{argmax}_{T} P(T|W)$$

Specifically using logistic regression:

$$\hat{T} = \operatorname*{argmax}_{T} \frac{\prod_i e^{\lambda_i f_i\,(T,W)}}{\sum_{T'} \prod_i e^{\lambda_i f_i\,(T',W)}}$$

Feature functions for CRFs

- Essentially the same as for MEMMs: can incorporate left and right context for observations (words)
- For tags, the probabilistic framework theoretically could encompass features built on arbitrary tag dependencies
  - E.g., first and last tag in sentence
  - But we need dynamic programming to make inference tractable. Therefore, in practice, tag dependencies are limited to adjacent tags
- "Linear chain CRF"

The partition function

- Remember our expression for the probability of a tag sequence under a CRF:

$$P(T|W) = \frac{\prod_i e^{\lambda_i f_i(T,W)}}{\sum_{T'} \prod_i e^{\lambda_i f_i(T',W)}}$$

The denominator is called the partition function. It involves a sum over all possible tag sequences for the sentence and is expensive to compute

But note that we don't need to compute it in order to predict the best tagging; the numerator is sufficient

Inference in CRFs: the Viterbi algorithm

$$\operatorname*{argmax}_T P(T|W) = \operatorname*{argmax}_T \frac{\prod_i e^{\lambda_i f_i\,(T,W)}}{\sum_{T'} \prod_i e^{\lambda_i f_i\,(T',W)}}$$

$$= \operatorname*{argmax}_T \prod_i e^{\lambda_i f_i\,(T,W)}$$

$$= \operatorname*{argmax}_T \sum_i \lambda_i f_i\,(T,W)$$

Inference in CRFs: the Viterbi algorithm

$$\underset{T}{\operatorname{argmax}}\, P(T|W) = \underset{T}{\operatorname{argmax}} \textstyle\sum_i \lambda_i f_i\,(T, W)$$

$$= \underset{T}{\operatorname{argmax}} \sum_{j=1}^{N} \sum_k \lambda_k f_k\,(T_{j-1}, T_j, W)$$

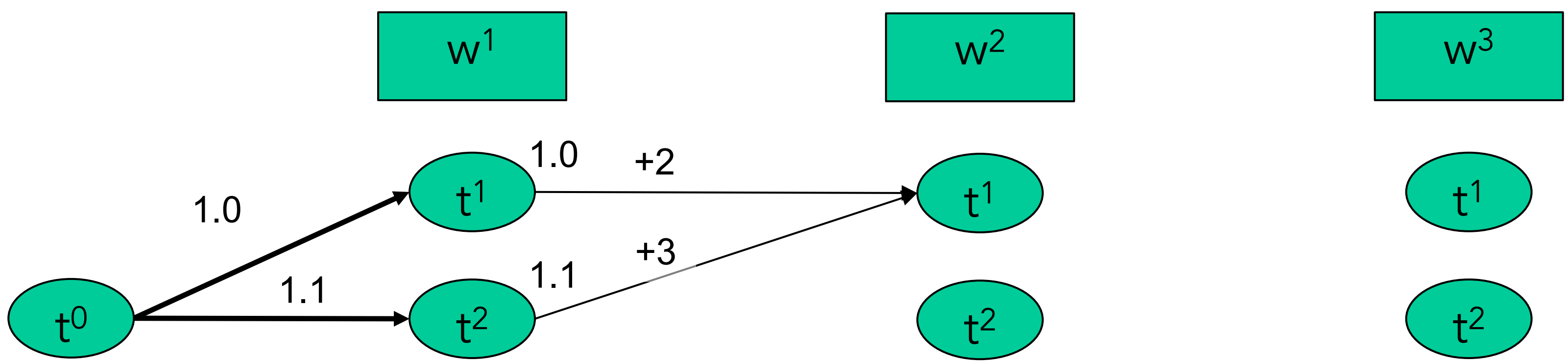
Because feature functions only express dependencies between adjacent tag pairs, we can group them into sets based on the final tag in the dependency chain

Inference in CRFs: the Viterbi algorithm

Word	the	stories	about	well-heeled	communities	and	developers
Tag	DT	NNS	IN	JJ	NNS	CC	NNS
Position	1	2	3	4	5	6	7

$$f_{1..K}(T_1, T_2, W)\ f_{1..K}(T_2, T_3, W)\ f_{1..K}(T_3, T_4, W)\ f_{1..K}(T_4, T_5, W)\ f_{1..K}(T_5, T_6, W)\ f_{1..K}(T_6, T_7, W)$$

Viterbi algorithm

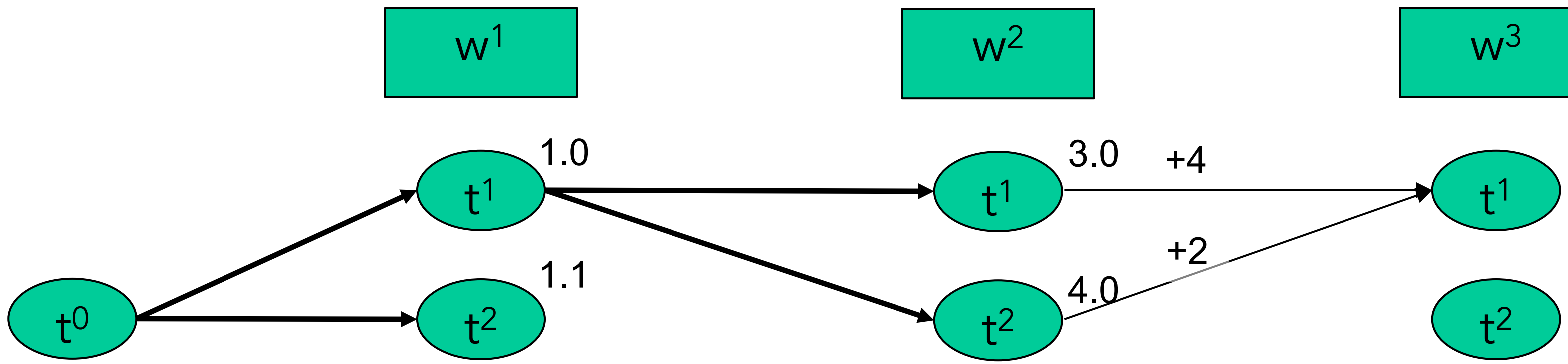


$$\phi(t_i, t_{i-1}, w_i, w_{i-1})$$

t_{i-1}	t^0		t^1									t^2								
w_i	w^1	w^2	w^1			w^2			w^3			w^1			w^2			w^3		
w_{i-1}			w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3
$t_i=t^1$	1.0	1.1	2	4	2	2	2	3	2	4	5	2	2	2	3	2	4	5	2	4
$t_i=t^2$	1.1	2.1	3	5	3	3	2	4	2	4	3	3	3	2	4	2	4	3	4	5

$$= -\sum_k \lambda_k f_k\,(t_i, t_{i-1}, w_i, w_{i-i})$$

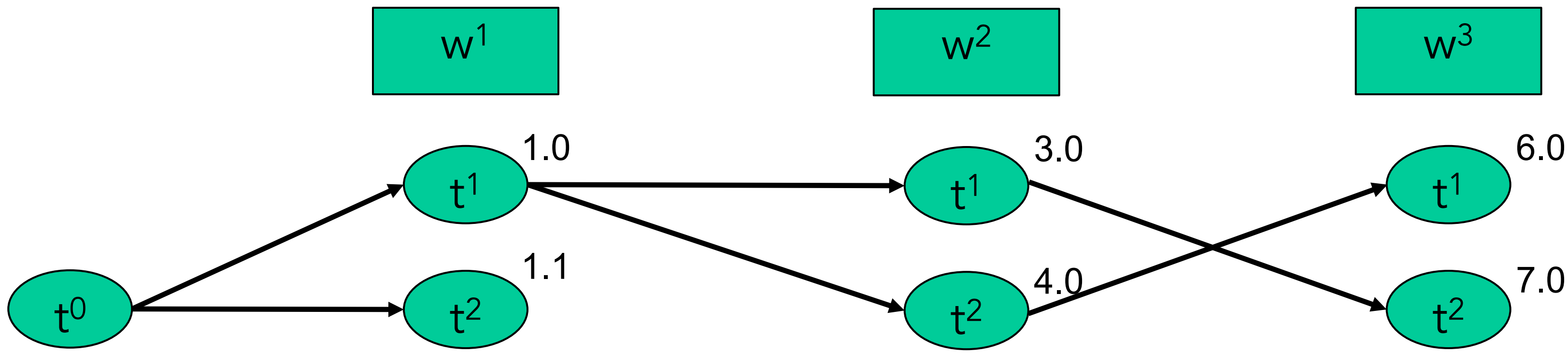
Viterbi algorithm



$\phi(t_i, t_{i-1}, w_i, w_{i-1})$

t_{i-1}	t^0		t^1									t^2								
w_i	w^1	w^2	w^1			w^2			w^3			w^1			w^2			w^3		
w_{i-1}			w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3
$t_i=t^1$	1.0	1.1	2	4	2	2	2	3	2	4	5	2	2	2	3	2	4	5	2	4
$t_i=t^2$	1.1	2.1	3	5	3	3	2	4	2	4	3	3	3	2	4	2	4	3	4	5

Viterbi algorithm



$$\phi(t_i, t_{i-1}, w_i, w_{i-1})$$

t_{i-1}	t^0		t^1									t^2								
w_i	w^1	w^2	w^1			w^2			w^3			w^1			w^2			w^3		
w_{i-1}			w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3	w^1	w^2	w^3
$t_i=t^1$	1.0	1.1	2	4	2	2	2	3	2	4	5	2	2	2	3	2	4	5	2	4
$t_i=t^2$	1.1	2.1	3	5	3	3	2	4	2	4	3	3	3	2	4	2	4	3	4	5

Training of CRFs: the forward recurrence

The conditional log likelihood of the training data under the model is

$$\log P_m(T^*|W) = \sum_{k=1}^{N} \log \frac{\prod_i e^{\lambda_i f_i (T_k^*, W_k)}}{\sum_{T'} \prod_i e^{\lambda_i f_i (T', W_k)}}$$

$$= \sum_{k=1}^{N} \left(\sum_i \lambda_i f_i \, (T_k^*, W_k) - \log \sum_{T'} \prod_i e^{\lambda_i f_i \, (T', W_k)} \right)$$

We want to compute gradients of this so that we can use gradient descent for optimization

Training of CRFs: the forward recurrence

$$\sum_{k=1}^{N}\left(\sum_{i}\lambda_i f_i\left(T_k^*, W_k\right) - \log\sum_{T'}\prod_{i} e^{\lambda_i f_i\left(T', W_k\right)}\right)$$

Sum of feature potentials; easy to compute

Partition function; expensive to compute

Use dynamic programming!

Sum over all training data

Training of CRFs: the forward recurrence

Define forward variable $a_i(t^j)$ as the sum of scores of all paths ending up with tag t^j at word w_i:

$$a_i(t^j) = \sum_{t' \in \mathfrak{T}} \left(e^{\lambda_k f_k(t', t^j, W)} \times a_{i-1}(t') \right)$$

Sum over all possible previous tags

Score for transitioning from previous tag to current tag

Forward variable for previous tag with previous word

MEMMs and CRFs: Key Points

- MEMMs and CRFs incorporate complex features for sequence labeling in a probabilistic framework
- MEMMs and CRFs are based on logistic regression (a.k.a. maximum entropy)
- CRFs improve on MEMMs by using globally, rather than locally normalized probabilities
- CRFs excel at incorporating global constraints on well-formedness of label sequences

[Notebook]

- https://github.com/scrapinghub/python-crfsuite/blob/master/examples/CoNLL%2002002.ipynb