COS 484

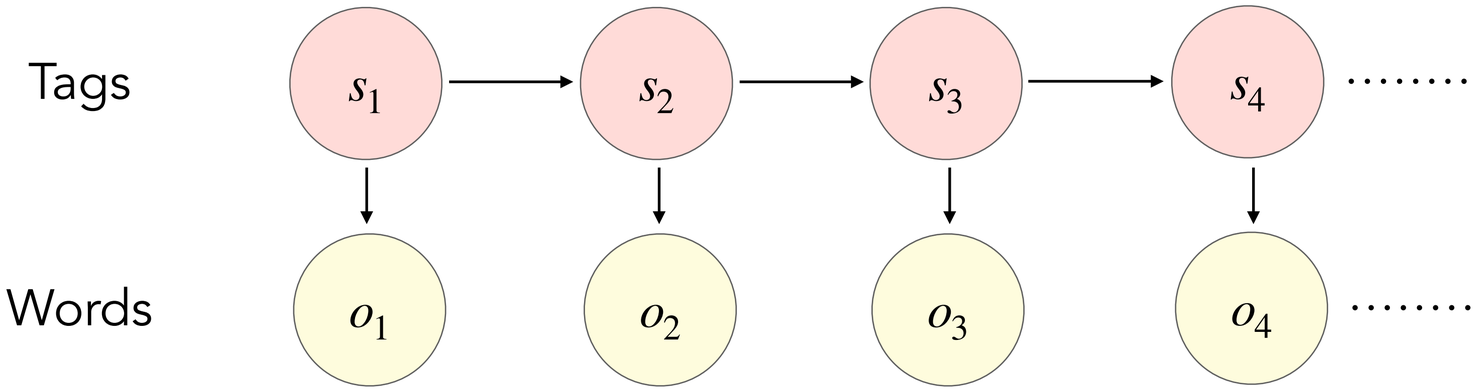
Sequence Models - 2

Spring 2022

Logistics

- Lecture slides:
  - uploaded previous evening/night
  - if you want a head start, you can view slides from Spring 2021 (link available on class website)
  - poll slides may be skipped, updated slides available after class (just re-download from website)
- **Office hours:** if you're unable to make any of the office hours, send us a message

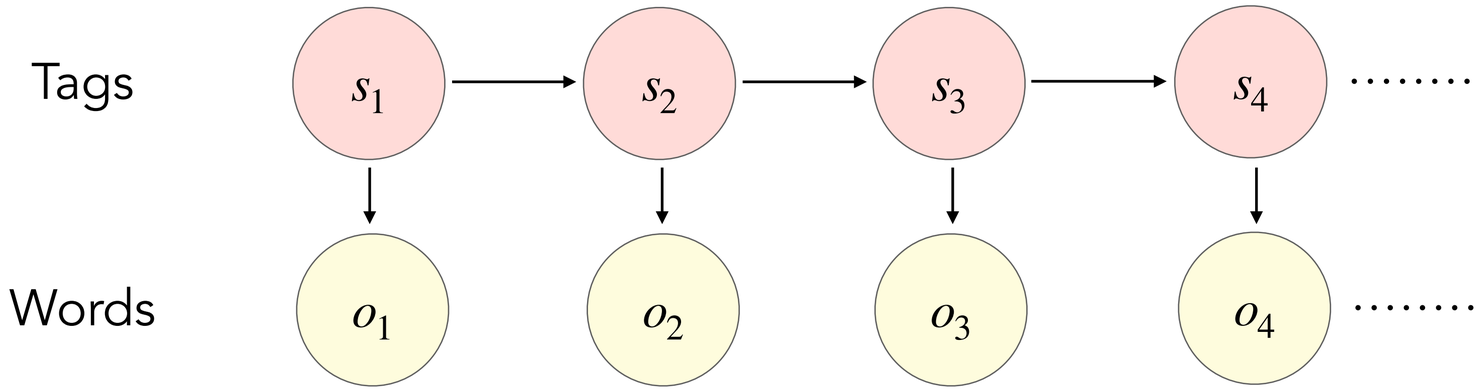
Recap: Hidden Markov Models



1. Set of states S = {1, 2, ..., K} and set of observations O
2. Initial state probability distribution $\pi(s_1)$
3. Transition probabilities $P(s_{t+1} \mid s_t)$ (OR $\theta_{s_t \rightarrow s_{t+1}}$)
4. Emission probabilities $P(o_t \mid s_t)$ (OR $\phi_{s_t \rightarrow o_t}$)

Strong assumptions

Recap: Hidden Markov Models



1. Markov assumption:

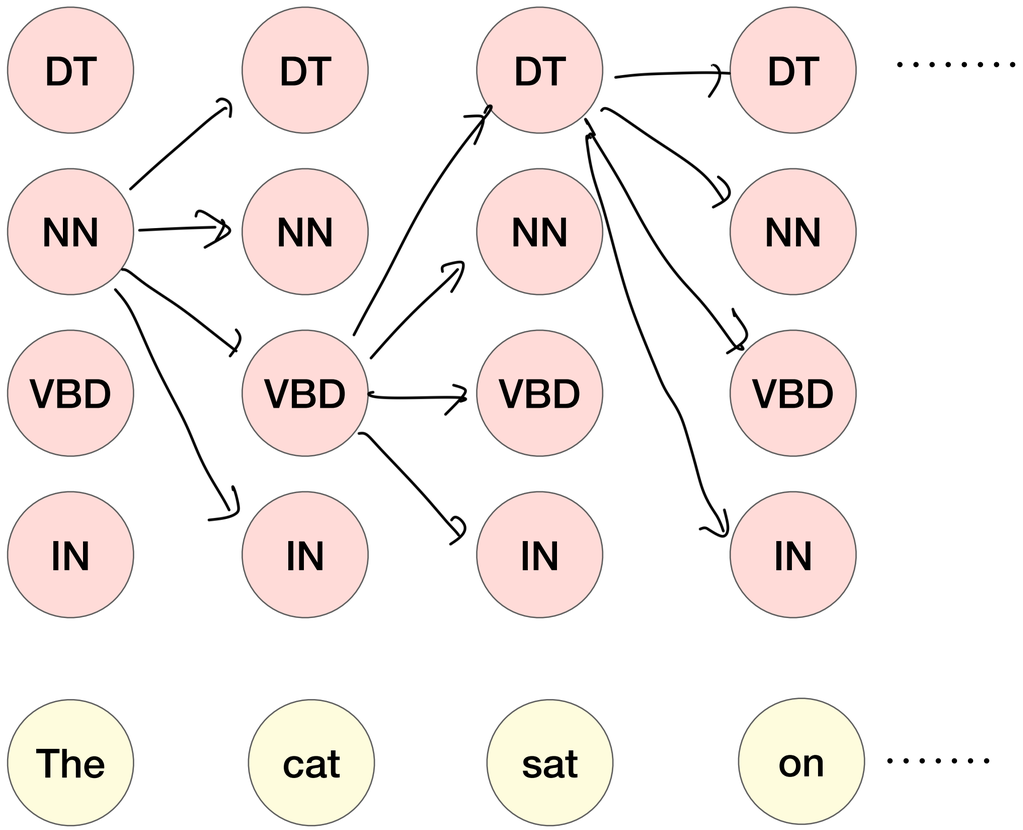
$$P(s_{t+1} \mid s_1, \ldots, s_t) \approx P(s_{t+1} \mid s_t)$$

2. Output independence:

$$P(o_t \mid s_1, \ldots, s_t) \approx P(o_t \mid s_t)$$

1) assumes **s**tate sequences do not have very strong priors/long-range dependencies
2) assumes neighboring **s**tates don't affect current **o**bservation

Recap:Viterbi decoding



$M[i,j]$ stores joint probability of most probable sequence of states ending with state **j** at time **i**

$$M[i,j] = \max_k M[i-1,k]\ P(s_j \mid s_k)\ P(o_i \mid s_j) \quad 1 \le k \le K \quad 1 \le i \le n$$

Backward: Pick $\max_k M[n,k]$ and backtrack using B

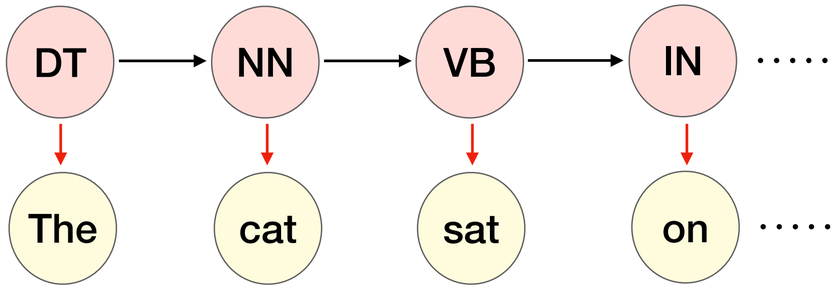
Maximum Entropy Markov Models

Generative vs Discriminative

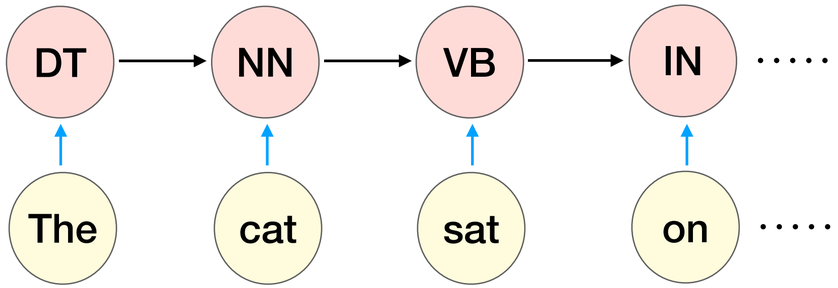
- HMM is a *generative* model
- Can we model $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$ directly?

	Generative	**Discriminative**
Text classification	Naive Bayes: $P(c)P(d \mid c)$	Logistic Regression: $P(c \mid d)$
Sequence prediction	HMM: $P(s_1, \ldots, s_n)P(o_1, \ldots, o_n \mid s_1, \ldots, s_n)$	MEMM: $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$ (No factorization)

Maximum Entropy Markov Model



HMM



MEMM

$$O = \langle o_1, o_2, \ldots, o_n \rangle$$

- Compute the posterior directly:
  - $P(S \mid O) \approx \arg\max_S \prod_i P(s_i \mid o_i, s_{i-1})$ ← No factorization into transition, emission
- Use features and weights: $P(s_i = s \mid o_i, s_{i-1}) \propto \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1}))$

Features

weights

(Bigram MEMM)

- Use features and weights: $P(s_i = s \mid o_i, s_{i-1}) \propto \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1}))$

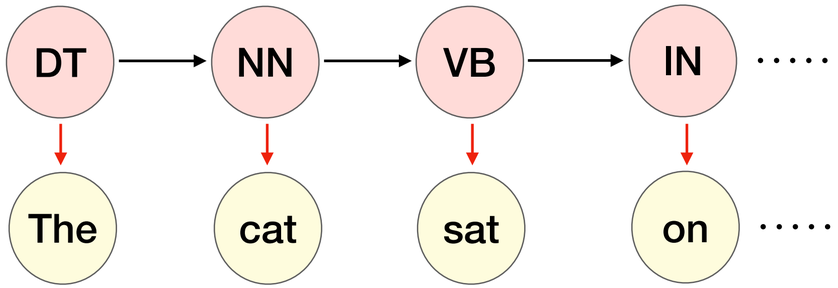
- *Which of the following is the correct way to calculate this probability?*

A) $P(s_i = s \mid o_i, s_{i-1}) = \dfrac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1}))}{\sum_{s' \in S} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1} = s'))}$

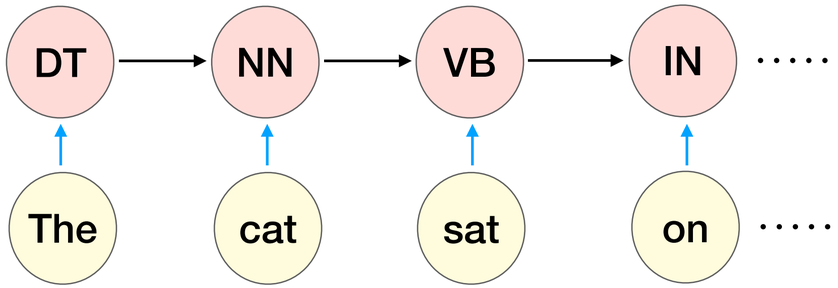
B) $P(s_i = s \mid o_i, s_{i-1}) = \dfrac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1}))}{\sum_{s' \in S} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s', o_i, s_{i-1}))}$

C) $P(s_i = s \mid o_i, s_{i-1}) = \dfrac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1}))}{\sum_{o' \in O} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i = o', s_{i-1}))}$

Maximum Entropy Markov Model



HMM



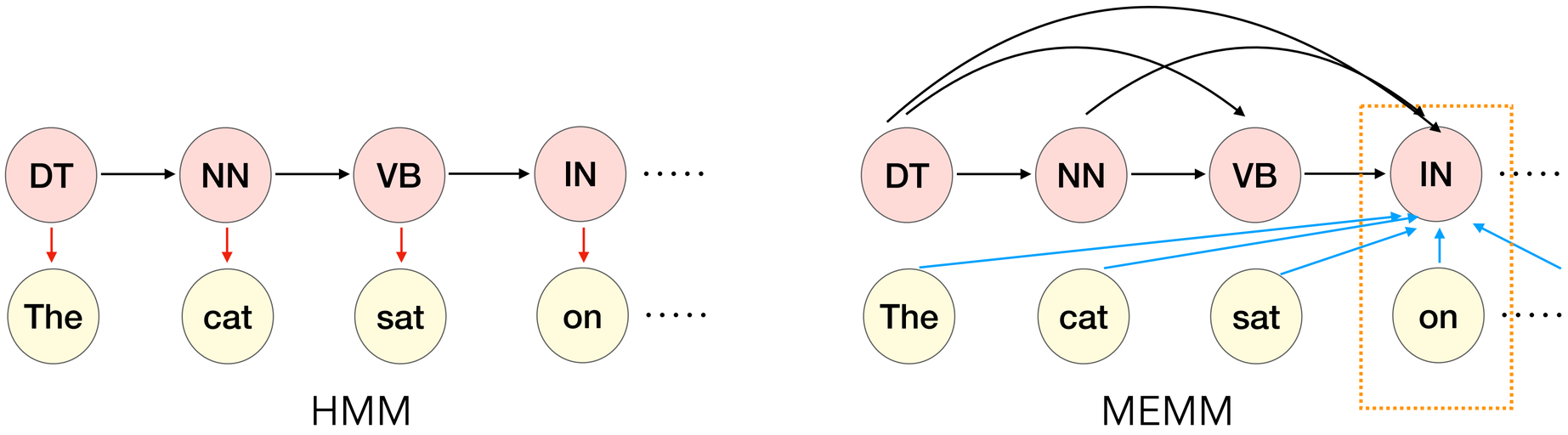
MEMM

$$O = \langle o_1, o_2, \ldots, o_n \rangle$$

- Compute the posterior directly:
  - $P(S \mid O) \approx \arg\max_S \prod_i P(s_i \mid o_i, s_{i-1})$
- Use features and weights: $P(s_i = s \mid o_i, s_{i-1}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(s_i = s, o_i, s_{i-1}))}{\sum_{s' \in S} \exp(\mathbf{w} \cdot \mathbf{f}(s_i = s', o_i, s_{i-1}))}$

(Bigram MEMM)

MEMM



- In general, we can use all observations and all previous states:

$$P(S \mid O) = \arg\max_{S} \prod_{i} P(s_i \mid o_n, o_{i-1}, \ldots, o_1, s_{i-1}, \ldots, s_1)$$

$$P(s_i \mid s_{i-1}, \ldots, s_1, O) \propto \exp(\mathbf{w} \cdot \mathbf{f}(s_i, s_{i-1}, \ldots, s_1, O)$$

Q: Why couldn't we do this with HMMs?

Features in an MEMM

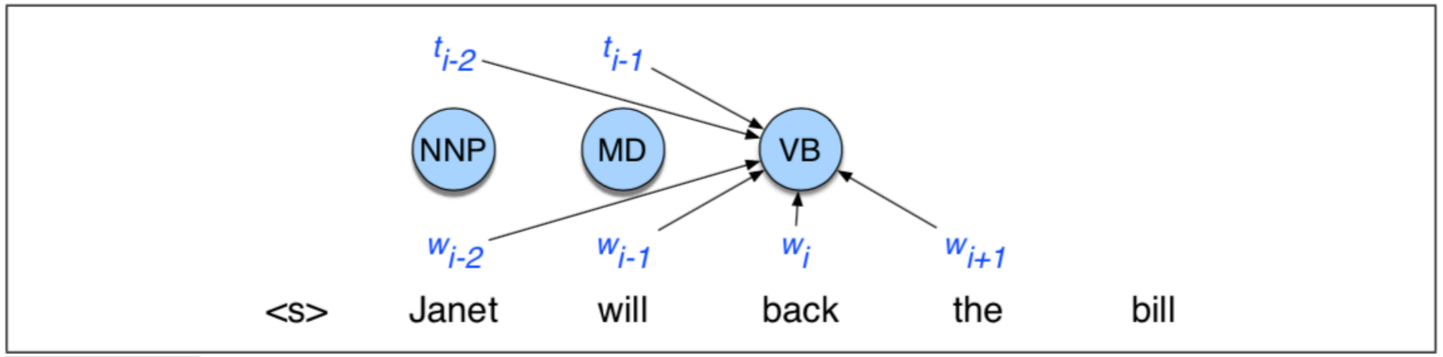


Figure 8.13 An MEMM for part-of-speech tagging showing the ability to condition on more features.

t_i = tags (states)

w_i = words (observations)

$$\langle t_i, w_{i-2}\rangle, \langle t_i, w_{i-1}\rangle, \langle t_i, w_i\rangle, \langle t_i, w_{i+1}\rangle, \langle t_i, w_{i+2}\rangle$$
$$\langle t_i, t_{i-1}\rangle, \langle t_i, t_{i-2}, t_{i-1}\rangle,$$
$$\langle t_i, t_{i-1}, w_i\rangle, \langle t_i, w_{i-1}, w_i\rangle \langle t_i, w_i, w_{i+1}\rangle,$$

Feature templates

$t_i = \text{VB}$ and $w_{i-2} = \text{Janet}$
$t_i = \text{VB}$ and $w_{i-1} = \text{will}$
$t_i = \text{VB}$ and $w_i = \text{back}$
$t_i = \text{VB}$ and $w_{i+1} = \text{the}$
$t_i = \text{VB}$ and $w_{i+2} = \text{bill}$
$t_i = \text{VB}$ and $t_{i-1} = \text{MD}$
$t_i = \text{VB}$ and $t_{i-1} = \text{MD}$ and $t_{i-2} = \text{NNP}$
$t_i = \text{VB}$ and $w_i = \text{back}$ and $w_{i+1} = \text{the}$

Features (binary)

Features in an MEMM

Incorrect	DT	JJ	NN	DT	NN
Correct	DT	NN	VB	DT	NN
	The	old	man	the	boat

Which of these feature templates would help most to tag 'old' correctly?

A) $\langle t_i, w_i, w_{i-2}, w_{i-1}, w_{i+1} \rangle$

B) $\langle t_i, w_i, w_{i-1} \rangle$

C) $\langle t_i, w_i, w_{i-1}, w_{i+1} \rangle$

D) $\langle t_i, w_i, w_{i-1}, w_{i+1}, w_{i+2} \rangle$

t = tags

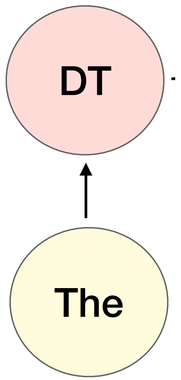
w = words

MEMMs: Decoding

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{S} \Pi_i P(s_i \mid o_i, s_{i-1})$$

(assume bigram MEMM, i.e. features only on previous time step and current obs)

- Greedy decoding:



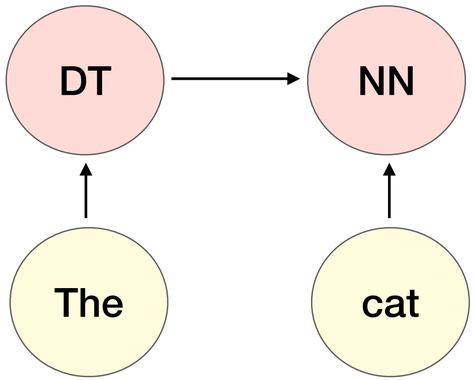
$$\hat{S}_1 = \underset{S}{\text{argmax}}\ P(S \mid \text{The})$$

$$= DT$$

MEMMs: Decoding

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{S} \Pi_i P(s_i \mid o_i, s_{i-1})$$

- Greedy decoding:



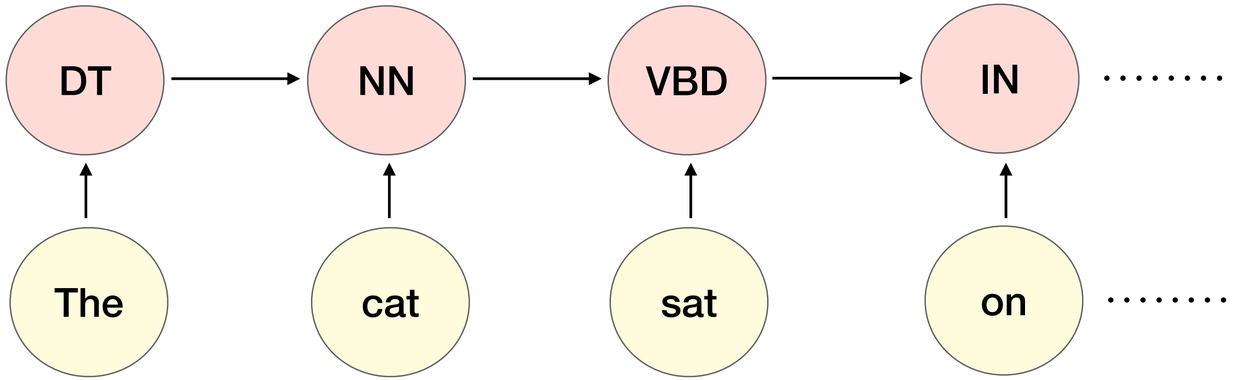
$$\hat{S}_2 = \arg\max_{S} P(S \mid \text{cat}, \text{DT})$$

$$= \text{NN}$$

MEMMs: Decoding

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{S} \Pi_i P(s_i \mid o_i, s_{i-1})$$

- Greedy decoding:



$$\forall t, \quad \hat{s}_{t+1} = \underset{s}{\text{argmax}}\ P(s \mid o_{t+1}, \hat{s}_t)$$

MEMMs: Decoding

$$\hat{S} = \arg\max_{S} P(S \mid O) = \arg\max_{S} \Pi_i P(s_i \mid o_i, s_{i-1})$$

- Greedy decoding

- Viterbi decoding:

$$M[i,j] = \max_{k} M[i-1,k]\ P(s_j \mid o_i, s_k) \quad 1 \le k \le K \quad 1 \le i \le n$$

(or equivalent log form)

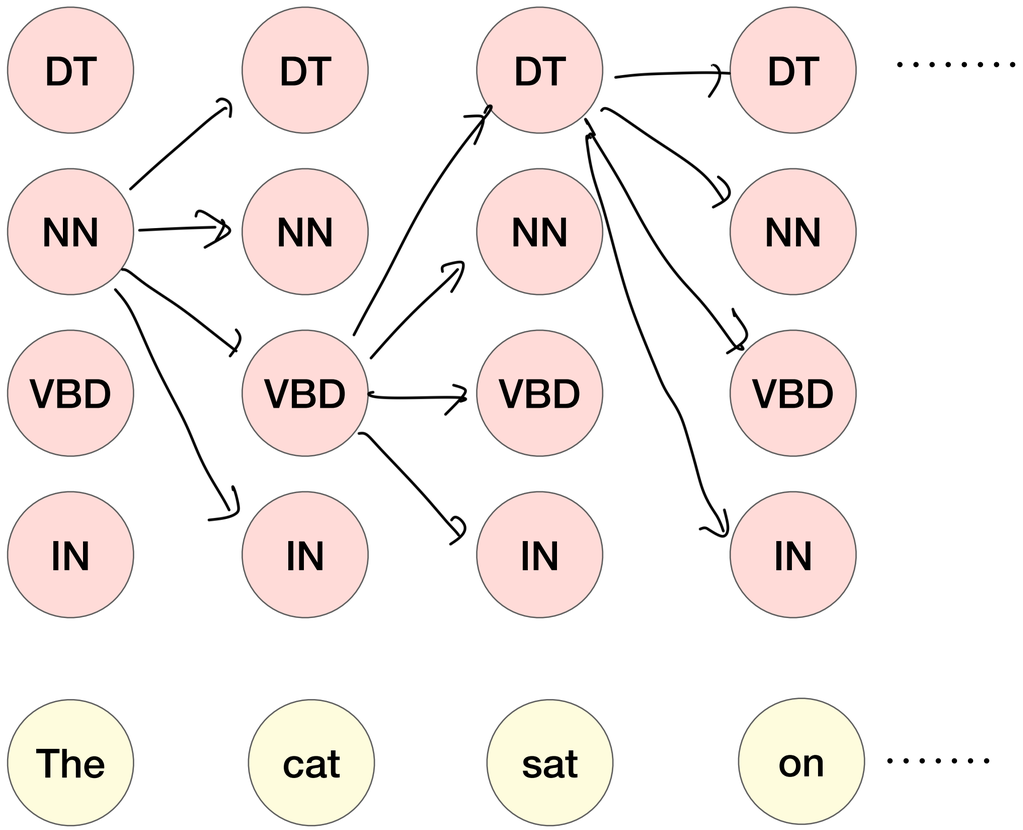
DP Lattice

(Best sequence ending in s_j)

states

timesteps

Viterbi decoding for MEMMs



$M[i,j]$ stores joint probability of most probable sequence of states ending with state **j** at time **i**

$$M[i,j] = \max_k M[i-1,k]\, P(s_j \mid o_i, s_k) \quad 1 \le k \le K \quad 1 \le i \le n$$

Backward: Pick $\max_k M[n,k]$ and backtrack using B

MEMMs: Decoding

How would you compare the computational complexity of Viterbi decoding for bigram MEMMs compared to decoding for bigram HMMs?

A) More operations in MEMM

B) More operations in HMM

C) Equal

D) Depends on number of features in MEMM

$$M[i,j] = \max_k M[i-1,k]\ P(s_j \mid o_i, s_k) \quad 1 \le k \le K \quad 1 \le i \le n$$

DP Lattice

states

timesteps

MEMM: Learning

- **Gradient descent:** similar to logistic regression!

$$P(s_i = s \mid s_1, \ldots, s_{i-1}, O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(s_1, \ldots, s_{i-1}, s_i = s, O))}{\sum_{s'} \exp(\mathbf{w} \cdot \mathbf{f}(s_1, \ldots, s_{i-1}, s_i = s', O))}$$

- **Given:** annotated pairs of (S, O) where each $S = \langle s_1, s_2, \ldots, s_n \rangle$

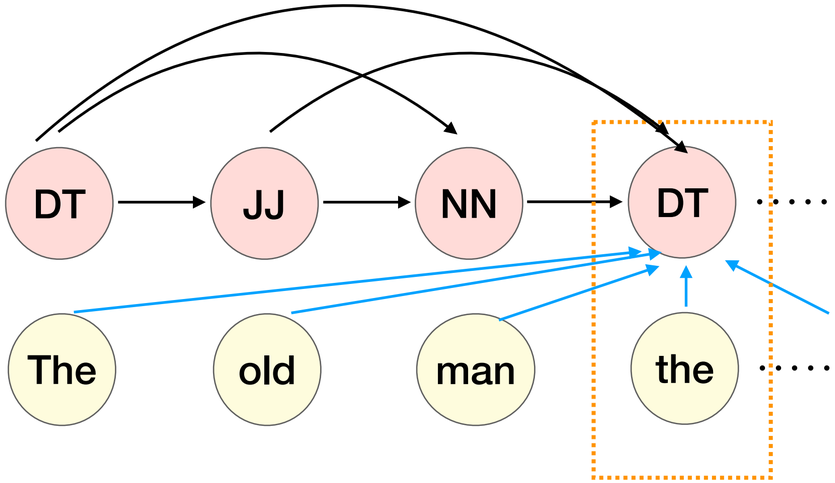
$$\text{Loss for one sequence, } L = -\sum_{i=1}^{n} \log P(s_i \mid s_1, \ldots, s_{i-1}, O)$$

- Compute gradients with respect to weights $\mathbf{w}$ and update

MEMM vs HMM

- HMM models the joint $P(S, O)$ while MEMM models the required prediction $P(S \mid O)$
- MEMM has more expressivity
  - accounts for dependencies between neighboring states and entire observation sequence
  - allows for more flexible features
- HMM may hold an advantage if the dataset is small

Label bias



The/? old/? man/? the/? boat/?

$$P(JJ \mid DT)\ P(\text{old} \mid JJ)\ P(NN \mid JJ)\ P(\text{man} \mid NN)\ P(DT \mid NN)$$

$$P(NN \mid DT)\ P(\text{old} \mid NN)\ P(VB \mid NN)\ P(\text{man} \mid VB)\ P(DT \mid VB)$$

Low entropy transitions (e.g. JJ -> NN) between labels may override the effect of observations

Stanford Parser

Please enter a sentence to be parsed:

The old man the boat

Language: English Sample Sentence Parse

Your query

The old man the boat

Tagging

```
The/DT  old/JJ  man/NN  the/DT  boat/NN
```

Solution?

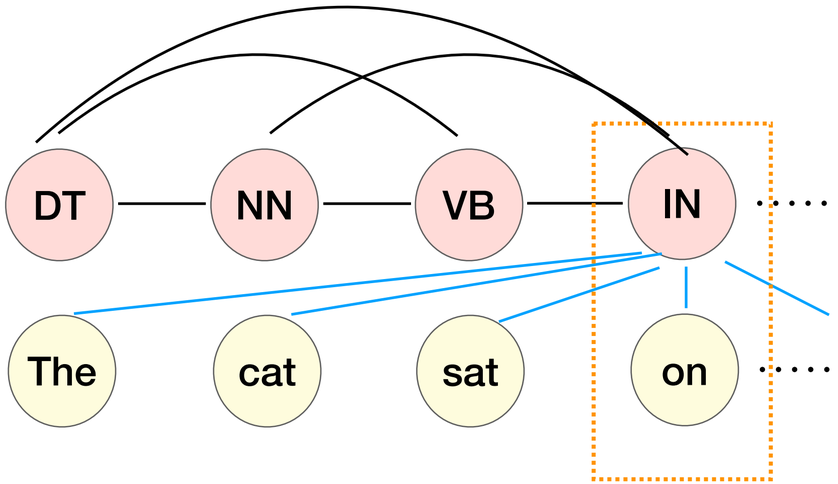
Conditional Random Fields

Conditional Random Fields

Conditional Random Field

- Model $P(s_1, \ldots, s_n \mid o_1, \ldots, o_n)$ directly
- No Markov assumption
  - Map entire sequence of states S and observations O to a **global** feature vector
- Normalize over entire sequences

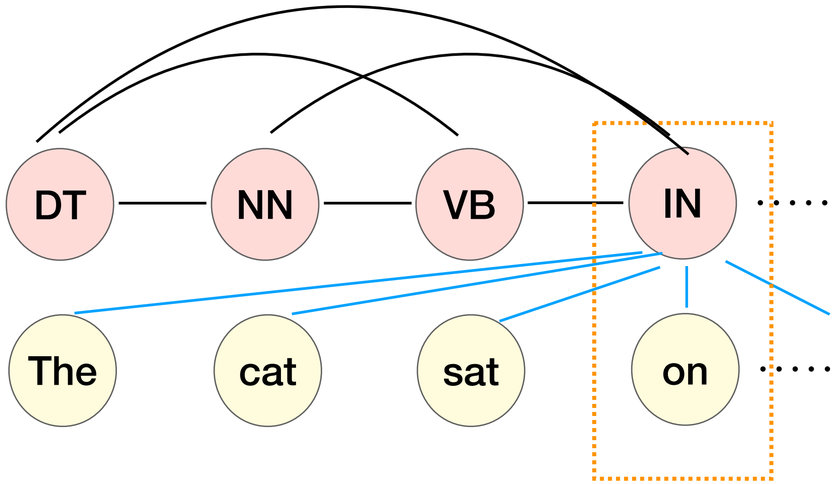
Conditional Random Field



$$P(S \mid O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{\sum_{S'} \exp(\mathbf{w} \cdot \mathbf{f}(S', O))}$$

$$= \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{Z(O)}$$

Features



$$P(S \mid O) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{\sum_{S'} \exp(\mathbf{w} \cdot \mathbf{f}(S', O))}$$

$$= \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{Z(O)}$$

- Each F_k in $\mathbf{f}$ is a **global** feature function
- Can be computed as a combination of local features (linear chain CRF)
  - $F_k = \sum_{i=1}^{n} f_k(s_{i-1}, s_i, O, i)$
  - each local feature only depends on previous and current states (outputs)

Features

- Each F_k in $\mathbf{f}$ is a **global** feature function

- Can be computed as a combination of local features (linear chain CRF)

$$\mathbb{1}\{x_i = \textit{the},\ y_i = \text{DET}\}$$
$$\mathbb{1}\{y_i = \text{PROPN}, x_{i+1} = \textit{Street}, y_{i-1} = \text{NUM}\}$$
$$\mathbb{1}\{y_i = \text{VERB}, y_{i-1} = \text{AUX}\}$$

- $$F_k = \sum_{i=1}^{n} f_k(s_{i-1}, s_i, O, i)$$

- **Training:** Optimize weights using supervised data similar to logistic regression

Inference

- $$\hat{S} = \arg\max_S P(S \mid O) = \arg\max_S \frac{\exp(\mathbf{w} \cdot \mathbf{f}(S, O))}{Z(O)}$$

$$= \arg\max_S \exp(\mathbf{w} \cdot \mathbf{f}(S, O))$$

$$= \arg\max_S \exp\left(\sum_{k=1}^{K} \sum_{i=1}^{n} w_k f_k(s_{i-1}, s_i, O, i)\right)$$

(for linear chain CRF)

- Use Viterbi similar to HMM and MEMM

CRF vs MEMM

- MEMM models the required prediction $P(S|O)$ using the Markov assumption, while the CRF does not
- CRF uses global features while MEMM features are localized
- Feature design is flexible in both models
- CRF is computationally more complex

History of CRFs

Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data

John Lafferty[†*] LAFFERTY@CS.CMU.EDU
Andrew McCallum[*†] MCCALLUM@WHIZBANG.COM
Fernando Pereira[*‡] FPEREIRA@WHIZBANG.COM
[*]WhizBang! Labs–Research, 4616 Henry Street, Pittsburgh, PA 15213 USA
[†]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA
[‡]Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104 USA

- Lafferty, McCallum, Pereria (2001): introduced CRFs for sequence modeling
  - Mitigates the label bias problem (in HMMs/MEMMs)
  - Better empirical performance compared to HMMs/MEMMs
  - Parameter estimation not straightforward

History of CRFs

- Very popular in the 2000s
- Wide variety of applications:
  - Information extraction
  - Summarization
  - Image labeling/segmentation

Information extraction from research papers using conditional random fields ☆

Fuchun Peng [a], Andrew McCallum [b]

Multiscale conditional random fields for image labeling

Publisher: IEEE Cite This PDF

Xuming He ; R.S. Zemel ; M.A. Carreira-Perpinan All Authors

Document Summarization using Conditional Random Fields

Dou Shen[1], Jian-Tao Sun[2], Hua Li[2], Qiang Yang[1], Zheng Chen[2]
[1]Department of Computer Science and Engineering
Hong Kong University of Science and Technology, Hong Kong
{dshen, qyang}@cse.ust.hk
[2]Microsoft Research Asia, 49 Zhichun Road, China
{jtsun, huli, zhengc}@microsoft.com

History of CRFs

- Very popular in the 2000s
- Wide variety of applications:
  - Information extraction
  - Summarization
  - Image labeling/segmentation

Software [edit]

This is a partial list of software that implement generic CRF tools.

- RNNSharp CRFs based on recurrent neural networks (C#, .NET)
- CRF-ADF Linear-chain CRFs with fast online ADF training (C#, .NET)
- CRFSharp Linear-chain CRFs (C#, .NET)
- GCO CRFs with submodular energy functions (C++, Matlab)
- DGM General CRFs (C++)
- GRMM General CRFs (Java)
- factorie General CRFs (Scala)
- CRFall General CRFs (Matlab)
- Sarawagi's CRF Linear-chain CRFs (Java)
- HCRF library Hidden-state CRFs (C++, Matlab)
- Accord.NET Linear-chain CRF, HCRF and HMMs (C#, .NET)
- Wapiti Fast linear-chain CRFs (C)[15]
- CRFSuite Fast restricted linear-chain CRFs (C)
- CRF++ Linear-chain CRFs (C++)
- FlexCRFs First-order and second-order Markov CRFs (C++)
- crf-chain1 First-order, linear-chain CRFs (Haskell)
- imageCRF CRF for segmenting images and image volumes (C++)
- MALLET Linear-chain for sequence tagging (Java)

Empirical performance

Model	F score
SVM combination (Kudo and Matsumoto, 2001)	94.39%
CRF	94.38%
Generalized winnow (Zhang et al., 2002)	93.89%
Voted perceptron	94.09%
MEMM	93.70%

Table 2: NP chunking F scores

null hypothesis	p-value
CRF vs. SVM	0.469
CRF vs. MEMM	0.00109
CRF vs. voted perceptron	0.116
MEMM vs. voted perceptron	0.0734

Table 4: McNemar's tests on labeling disagreements

Sha and Pereira (2003)

CRFs in deep learning era

Conditional Random Fields as Recurrent Neural Networks

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, Philip H. S. Torr, Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1529-1537

Neural Architectures for Named Entity Recognition

Guillaume Lample[♠] **Miguel Ballesteros**[♣♠]
Sandeep Subramanian[♠] **Kazuya Kawakami**[♠] **Chris Dyer**[♠]
[♠]Carnegie Mellon University [♣]NLP Group, Pompeu Fabra University
{glample,sandeeps,kkawakam,cdyer}@cs.cmu.edu,
miguel.ballesteros@upf.edu

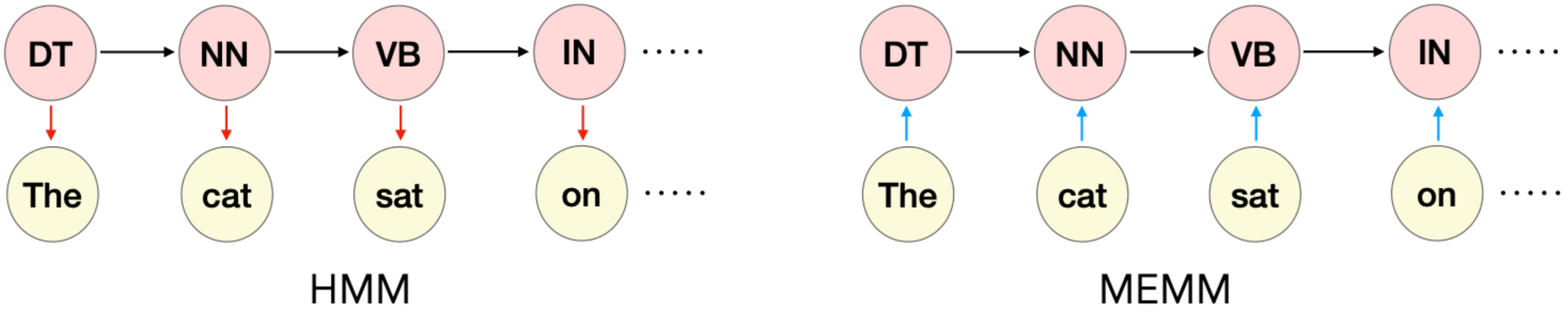
Bidirectional LSTM-CRF Models for Sequence Tagging

Zhiheng Huang Baidu research huangzhiheng@baidu.com
Wei Xu Baidu research xuwei06@baidu.com
Kai Yu Baidu research yukai@baidu.com

- Use CRFs on top of neural representations (instead of features and weights)
- Joint sequence prediction without the need for defining features!
- Recent architectures such as seq2seq w/ attention or Transformer may implicitly do the job

Preview: Recurrent neural networks (RNNs)

How can we model sequences using **neural networks**?

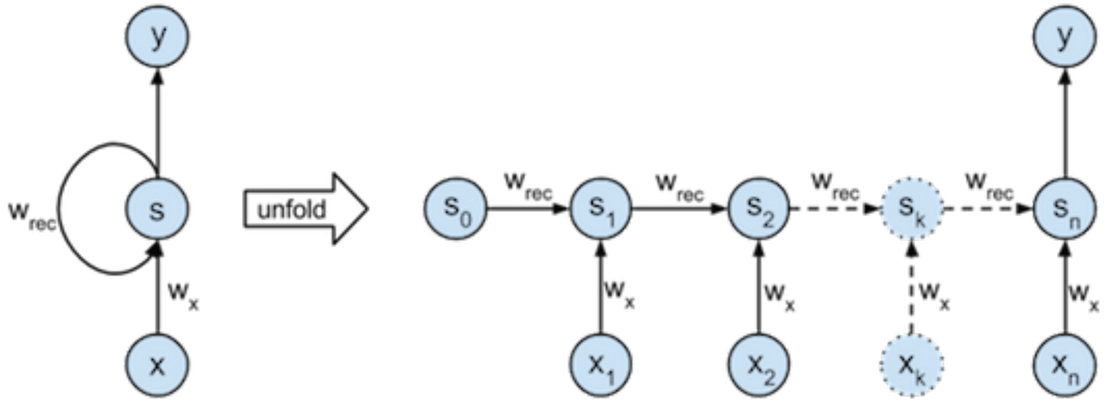


- Recurrent neural networks = A class of neural networks used to model sequences, allowing to handle **variable length inputs**

- Very crucial in NLP problems (different from images) because sentences/paragraphs are variable-length, sequential inputs

Preview: Recurrent neural networks (RNNs)

A family of neural networks allowing to handle **variable length inputs**

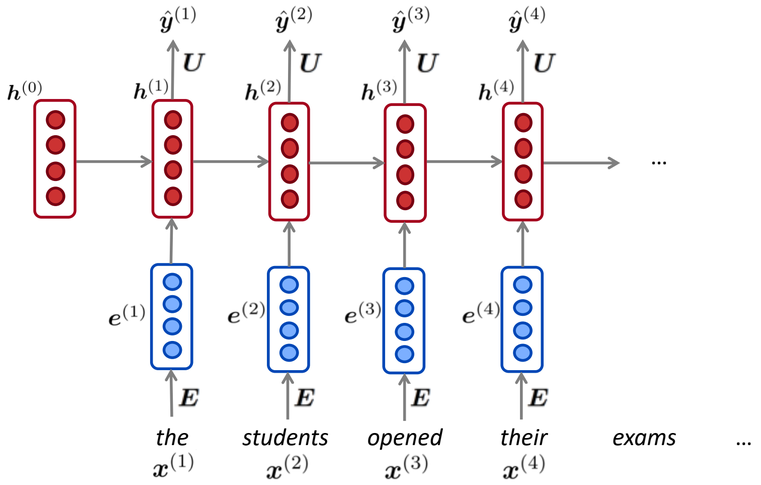


A function: $y = \text{RNN}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{R}^h$ where $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

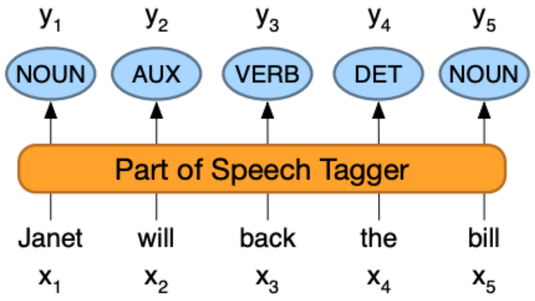
Preview: Recurrent neural networks (RNNs)

Proven to be an highly effective approach to language modeling, sequence tagging as well as text classification tasks:

Language modeling



Sequence tagging



Text classification