Markov Models

Simon Clematide
Institute of Computational Linguistics
University of Zurich

http://www.cl.unizh.ch/siclemat/talks/markov/

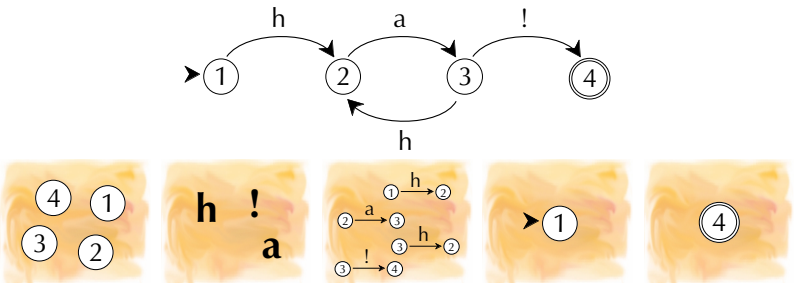
Synopsis

Program

- **Markov Chains** vs. Finite State Automatons
  - Acceptability vs. probability of symbol sequences
  - Probability of state sequences in Markov Chains
  - Formal definitions
  - N-Grams and nth order Markov Chains
- **Markov Models** = Markov Chain + symbol emission probability
  - Definition of state emitting, discrete output, first order Markov Model
  - Task I: Joint probability of a state/symbol sequence
  - Task II: Total probability of a symbol sequence
    - Forward Algorithm
  - Task III: Most probable state sequence for a given symbol sequence
    - Viterbi Algorithm

(Deterministic) Finite State Automatons

Remember Finite State Automatons…

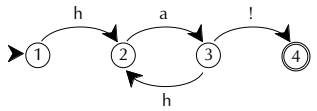


Tasks for FSA

Classification of symbol sequences (aka. strings)

- Which strings $s \in \{h,a,!\}^*$ are accepted by automaton A?

```
?- accept([h,a,!]).
?- accept([h,a]).
```




```
accept(String) :-
  start(StartState),
  accept(String, StartState).

accept([], State) :-
  final(State).
accept([Char|Chars], State) :-
  trans(State, Char, NextState),
  accept(Chars, NextState).
```

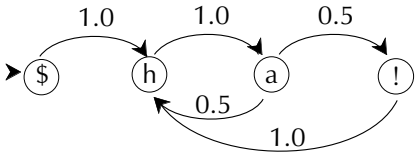
Accept Function

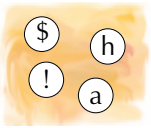
```
start(1).
final(4).

trans(1, h, 2).
trans(2, a, 3).
trans(3, h, 2).
trans(3, !, 4).
```

Automaton

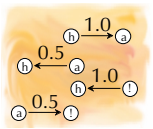
(Discrete First Order) Markov Chains

Some similarity…





States



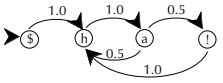
Transitions with Probabilities

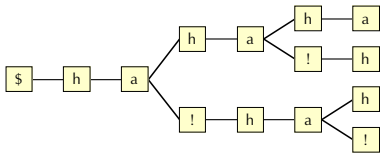
Start State

Task for Markov Chains

Probability of state sequences

- Which probability does the Markov Chain *M* assign to a state sequence $S \in \{h,a,!\}^*$?
  - Give the probability of sequences
    (h,a,!)
    (h,a,!,h,a,!,h,a)
    (a,h,a)





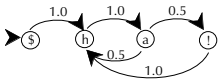
Probability tree for sequences up to length 6

What do equal length paths of the tree have in common?

Task for Markov Chains

Probability of state sequences

- Which probability does the Markov Chain *M* assign to the state sequence $S \in \{h,a,!\}^*$?




```
?- prob([h,a,!], P).
?- prob([h,a], P).
```

```
prob(Sequence, P) :-
    start(Start),
    prob([Start|Sequence], 1.0, P).

prob([Last], P, P).
prob([S1,S2|States], P0, P) :-
  trans(S1, S2, P1),
  P2 is P0*P1,
  prob([S2|States], P2, P).
```

Sequence Probability Function

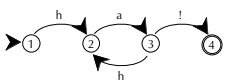
```
start($).
```

```
trans($, h, 1.0).
trans(h, a, 1.0).
trans(a, h, 0.5).
trans(a, !, 0.5).
trans(!, h, 1.0).
```

Markov Chain

Matrix Notations

Finite State Automaton vs. Markov Chain



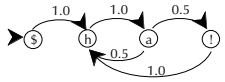
	h	a	!
1	2	∅	∅
2	∅	3	∅
3	2	∅	4
4:	∅	∅	∅

Symbols

To states

sink state

From states



	h	a	!
$	1	0	0
h	0	1	0
a	0.5	0	0.5
!	1	0	0

To states

Proba-bilities

From states

Formal Definition with Start State

Definition

- A Markov Chain $MC=(\mathcal{S}, s_0, A)$ consists of
  - $\mathcal{S}$ a **finite set** of states
  - $s_0 \notin \mathcal{S}$ a special **start state**
  - A: $(\mathcal{S} \cup \{s_0\} \times S) \rightarrow [0,1]$ a **conditional state transition probability** function
- Remarks
  - $A(i,j)$ is normally notated as a_{ij} or $P(j \mid i)$
  - A is often called "stochastic transition matrix"
  - for all i,

$$\sum_j a_{ij} = 1$$

Formal Definition with Initial Vector

Definition

- A Markov Chain $MC=(\mathcal{S}, A, \Pi)$ consists of
  - $\mathcal{S}$: a **finite set** of states
  - A: $(\mathcal{S} \times \mathcal{S}) \rightarrow [0,1]$ a **conditional state transition probability** function
  - Π: $\mathcal{S} \rightarrow [0,1]$ an **initial state probability** function
- Remarks
  - $\Pi(i)$ is often notated as π_i or $P(i)$
  - Both versions are equivalent; for all i

$$\pi_i = a_{s_0 i}$$

and therefore, $\sum_i \pi_i =$

	h	a	!
h	0	1	0
a	0.5	0	0.5
!	1	0	0

Transition Matrix

h	a	!
1	0	0

Initialization Vector

	h	a	!
$	1	0	0
h	0	1	0
a	0.5	0	0.5
!	1	0	0

Start State Matrix

Probability of State Sequence

Laws for Markov Chains

$$P(X_{t+1} = s_u \mid X_1, \ldots, X_t) = P(X_{t+1} \mid X_t)$$

Limited Horizon

$$P(X_{i+1} = s_u \mid X_i = s_v) = P(s_u \mid s_v)$$

Time Invariant (distribution)

Derivation of $P(X_1 = s_1, \ldots, X_T = s_T)$

$$= P(X_1) \cdot P(X_2 \mid X_1) \cdot P(X_3 \mid X_1, X_2) \cdot \cdots \cdot P(X_T \mid X_1, \ldots, X_{T-1}) \quad \text{[Chain Rule]}$$

$$= P(X_1) \cdot P(X_2 \mid X_1) \cdot P(X_3 \mid X_2) \cdot \cdots \cdot P(X_T \mid X_{T-1}) \quad \text{[Limited Horizon]}$$

$$= P(s_1) \cdot P(s_2 \mid s_1) \cdot P(s_3 \mid s_2) \cdot \cdots \cdot P(s_T \mid s_{T-1}) \quad \text{[Time Invariant]}$$

$$= \pi_{s_1} \cdot a_{s_1 s_2} \cdot a_{s_2 s_3} \cdot \cdots \cdot a_{s_{T-1} s_T} \quad \text{[insert } \pi \text{ and } a \text{]}$$

$$= \pi_{s_1} \cdot \prod_{i=1}^{i=T-1} a_{s_i s_{i+1}} \quad \text{[apply product notation]}$$

Computing in Log Space

How to avoid multiplication of many small numbers

- Power Law of Multiplication

$$x \cdot y = 2^{\log_2 x + \log_2 y}$$

- Some numbers

$$1 = 2^0 = \log_2 0 \qquad 0.5 = 2^{-2} = \log_2 -2 \qquad 0 = 2^{-\infty} = \log_2 -\infty$$

- Transition probabilities as powers of $\log_2$
- Computing log probability by summation

$$\log_2 P(s_1 \ldots s_T) = \pi_{s_1} + \sum_{i=1}^{i=T-1} a_{s_i s_{i+1}}$$

- How to add to $-\infty$? $\qquad -\infty + x = -\infty$

	h	a	!
h	0	1	0
a	0.5	0	0.5
!	1	0	0

Transition Matrix

	h	a	!
h	-inf	0	-inf
a	-2	-inf	-2
!	0	-inf	-inf

Transitions in $\log_2$

N-grams and Markov Chains

Constructing Markov Chains from empirical linguistic data

- Count bigrams from any sequential data as
  - words, part-of-speech tags, chunk tags, phones, …

$$freq(s_i, s_{i+1})$$

- Count unigrams as well

$$freq(s_i)$$

- Compute conditional transition probability

$$a_{s_i s_{i+1}} = P(s_{i+1} \mid s_i) = \frac{freq(s_i, s_{i+1})}{freq(s_i)}$$

Word	PoS	Chunk
Shares	NNS	B-NP
of	IN	B-PP
the	DT	B-NP
new	JJ	I-NP
partnership	NN	I-NP
would	MD	B-VP
trade	VB	I-VP
on	IN	B-PP
an	DT	B-NP
exchange	NN	I-NP
like	IN	B-PP
a	DT	B-NP
stock	NN	I-NP
.	.	O

Bigrams Annotated Corpus Data

from the CONL 2000 shared task

Where to Start?

What about initial probability?

- Treating a corpus as one long sequence gives few hints on initial probability.
- But (textual) linguistic data has natural segments: sentences!
  - Treat corpus as sequence of sequences!
- Count unigrams of **initial** elements

$$freq_{init}(s_i)$$

- Compute initial probability

$$\pi_{s_i} = P(s_i) = \frac{freq_{init}(s_i)}{\sum_j freq_{init}(s_j)}$$

Word	PoS	Chunk
and	CC	I-NP
manager	NN	I-NP
.	.	O
Shares	**NNS**	**B-NP**
of	IN	B-PP
the	DT	B-NP
⋮	⋮	⋮
like	IN	B-PP
a	DT	B-NP
stock	NN	I-NP
.	.	O
Hallwood	**NNP**	**B-NP**
is	VBZ	B-VP
a	DT	B-NP

Segmented Corpus

Where is the Start State?

What about the start state?

- Assume a special element in front of all segments! Say $ or just an empty line…
- Count the bigrams of $ and initial elements

$$freq(\$, s_i)$$

- Compute start state transition probability

$$a_{s_0 s_i} = P(s_i \mid s_0) = \frac{freq(\$, s_i)}{freq(s_i)}$$

- Special element can also be viewed as an end state – introducing transitions to them in Markov Chains as in FSA!

Word	PoS	Chunk
and	CC	I-NP
manager	NN	I-NP
.	.	O
$	$	$
Shares	NNS	B-NP
of	IN	B-PP
⋮	⋮	⋮
a	DT	B-NP
stock	NN	I-NP
.	.	O
$	$	$
Hallwood	NNP	B-NP
is	VBZ	B-VP
a	DT	B-NP

Annotated Corpus Data

Empirical linguistic Markov Chains

Querying frequency information

- How many words and word types?

```
?- findall(W, l(_,_,W,_,_), Words),
   length(Words, NumWords),
   sort(Words, Lexicon),
   length(Lexicon, NumTypes).
```

```
...
l(8699,30,'partner','NN','I-NP').
l(8699,31,'and','CC','I-NP').
l(8699,32,'manager','NN','I-NP').
l(8699,33,'.','.','O').
l(8700,1,'Shares','NNS','B-NP').
l(8700,2,'of','IN','B-PP').
l(8700,3,'the','DT','B-NP').
...
```

Corpus in PROLOG

- How many word bigrams and bigram types?

```
?- findall(W1-W2,(l(S,N1,W1,_,_),N2 is N1+1,l(S,N2,W2,_,_)),Bigrams),
   length(Bigrams, NumBigrams),
   sort(Bigrams, BigramTypes),
   length(BigramsTypes, NumTypes).
```

- Why is there only a (small) fraction of the possible bigrams in the corpus?

	Token	Types	Bigrams found	in %
Words	211272	19122	104896	0.03%
PoS	211272	44	1094	56.51%
Chunk tags	211272	22	145	29.96%

Markov Chains as Linguistic Models

How adequate are Markov Chain Models?

- Linguistic first order Markov Chain Models are essentially bigram models.
- Therefore, they share the same **strength**
  - as ...

  and **weakness**
  - as ...

N-Grams and *N*th Order MC

In second order Markov Chains

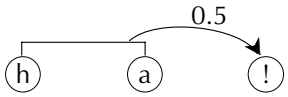
transition probability depends on 2 preceding states.

- Compute probabilities using trigram frequency

$$P(s_3 \mid s_1, s_2) = \frac{freq(s_1, s_2, s_3)}{freq(s_1, s_2)}$$

- Build n-th order MC using $n+1$-grams!

$P(s_3 \mid s_1, s_2) = P(!\mid \mathrm{h}, \mathrm{a}) = 0.5$



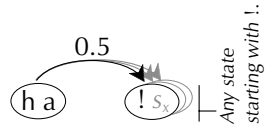
Second Order Transition

Every *n*th order Markov Chain

can be represented by a first order MC with history of length n encoded in the states.

- Represent every n-th order MC with k states using a first order MC with k^n states!

$P(!\, s_x \mid \mathrm{h\,a}) = 0.5$



Equivalent First Order Transitions

Definition of (Hidden) Markov Models

Definition (following Brants)

- A discrete output, first order Hidden Markov Model $HMM=(\mathcal{S}, s_0, s_e, A, \mathcal{V}, B)$ consists of
  - $\mathcal{S}$ a **finite set** of states $\{s_1,\ldots,s_n\}$
  - $s_0 \notin \mathcal{S}$ a special **start state**
  - $s_e \notin \mathcal{S}$ a special **end state**
  - A: $(\mathcal{S}\cup\{s_0\} \times \mathcal{S}\cup\{s_e\}) \rightarrow [0,1]$ a **conditional state transition probability** function
  - $\mathcal{V}$ a finite set of emission symbols $\{w_1,\ldots,w_m\}$
  - B: $(\mathcal{S} \times \mathcal{V}) \rightarrow [0,1]$ a **conditional symbol emission probability** function

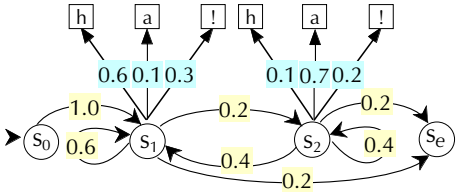
Remarks

HMM are

- Markov Chains equipped with probabilistic symbol emission.
- either **state emitting** or **state transition emitting** (as in M&S).

Remarks

- $B(s_i, w_j)$ is notated as b_{ij} or just $P(w \mid s)$.
- For all i,
  $\sum_j b_{ij} = 1$
- Start state and end state do **not** emit symbols!



Laughing Machine as HMM

Constructing MM from empirical data

What are the states? E.G. PoS-tags

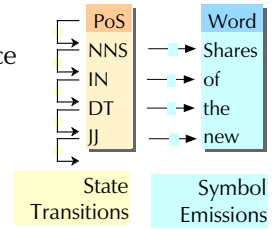
- Count bigrams and unigrams of tag sequence

$freq(t_1, t_2)$ $freq(t_1)$

What are the symbols? E.G. words.

- Count word-tag combinations

$freq(w, t)$



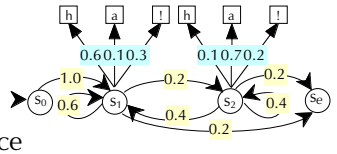
Estimate probabilites

- Compute conditional transition probabilities

$$a_{t_1 t_2} = P(t_2 \mid t_1) = \frac{freq(t_1, t_2)}{freq(t_1)} \qquad b_{tw} = P(w \mid t) = \frac{freq(w, t)}{freq(t)}$$

Task I for Markov Models

Joint probability of given state and symbol sequences



- Which joint probability does Markov Model *M* assign to the symbol sequence (h,a) while running through the state sequence (s_0,s_1,s_2,s_e)?

$P(s_1 \mid s_0)P(\text{h} \mid s_1)P(s_2 \mid s_1)P(\text{a} \mid s_1)P(s_e \mid s_2)$

$= 1.0 \cdot 0.6 \cdot 0.2 \cdot 0.7 \cdot 0.2 = 168 / 10000$

S	1	2	e
0	1	0	0
1	0.6	0.2	0.2
2	0.4	0.4	0.2

A as Matrix

- Or state sequence(s_0,s_1,s_1,s_e)?

$P(s_1 \mid s_0)P(\text{h} \mid s_1)P(s_1 \mid s_1)P(\text{a} \mid s_1)P(s_e \mid s_1)$

$= 1.0 \cdot 0.6 \cdot 0.6 \cdot 0.1 \cdot 0.2 = 72 / 10000$

S\V	h	a	!
1	0.6	0.1	0.3
2	0.1	0.7	0.2

B as Matrix

Task I: The Formula

In general

$P(W, S)$

$= P(w_1, w_2, \ldots, w_t, s_0, s_1, s_2, \ldots, s_t, s_e)$

[Applying some Markov assumption…]

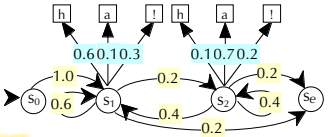
$= \ldots$

$= P(s_1 \mid s_0)P(w_1 \mid s_1) \cdot \ldots \cdot P(s_{t-1} \mid s_t)P(w_t \mid s_t) \cdot P(s_e \mid s_t)$

$= \left(\prod_{i=1}^{i=t} P(s_i \mid s_{i-1})P(w_i \mid s_i) \right) \cdot P(s_e \mid s_t)$

Task I: Implementation

The structure




```
a(s0, s1, 1.0).
a(s1, s1, 0.6).
a(s1, s2, 0.2).
a(s1, se, 0.2).
a(s2, s1, 0.4).
a(s2, s2, 0.4).
a(s2, se, 0.2).
```

A in Prolog

```
b(s1, h, 0.6).
b(s1, a, 0.1).
b(s1, !, 0.3).
b(s2, h, 0.1).
b(s2, a, 0.7).
b(s2, !, 0.2).
```

B in Prolog

```
s(s1).
s(s2).
```

S in Prolog

Computation of joint probability

```
?- prob([h,a], [s1,s2], P).
```

```
prob(W, S, P) :-
  prob(W, [s0|S], 1.0, P).
```

Initialisation, Accumulator and Recursion…

```
prob([], [St], P0, P):-
  a(St, se, A),
  P is P0*A.
prob([W1|W], [S1,S2|S], P0, P) :-
  a(S1, S2, A),
  b(S2, W1, B),
  P1 is P0*A*B,
  prob(W, [S2|S], P1, P).
```

Task II for Markov Models

Total probability of a given emission sequence

- Which probability does Markov Model *M* assign to a emission sequence *W* using all possible state sequences?
- Straight-forward solution
  - Enumerate all possible state sequences and sum up!

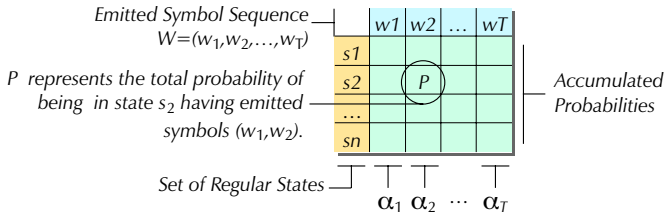
$$P(W) = \sum_{S \in S^T} P(W, S)$$

```
?- findall(P, prob([h,a,h,a,h,a,!],S,P), Ps),
   sum_list(Ps, TotalProb).
```

- This is naive! Why…
  - Take a symbol sequence of length *T*. Since there are $|S|$ possible states for each emitted symbol, there are $|S|^T$ possible state sequences!
  - Computation time grows exponentially!
- Solution: Use a caching algorithm!

Forward Algorithm & Alpha

Alpha, the caching data structure



The Forward Algorithm

- initializes alpha in column α_1.
- fills the alpha matrix column by column from left to right.
- computes the final total probability from the last column α_T.

Induction of Alpha

F1: Initialization

$$\alpha_1(s) = P(s \mid s_0)P(w_1 \mid s)$$

		$w1$
s_0	$s1$	$\alpha_1(s_1)$
	$s2$	$\alpha_1(s_2)$
	…	…
	sn	$\alpha_1(s_n)$

F2: Induction for $0 < t < T$

$$\alpha_{t+1}(s) = \left(\sum_{s' \in S} \alpha_t(s')P(s \mid s')\right)P(w_{t+1} \mid s)$$

	…	wt	$wt+1$
$s1$	…	$\alpha_t(s_1)$	$\alpha_{t+1}(s_1)$
$s2$	…	$\alpha_t(s_2)$	$\alpha_{t+1}(s_2)$
…	…	…	…
sn	…	$\alpha_t(s_n)$	$\alpha_{t+1}(s_n)$

F3: Final computation

$$P(W) = \sum_{s \in S} \alpha_T(s)P(s_e \mid s)$$

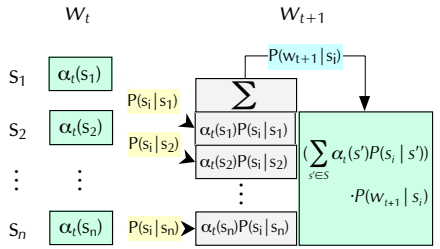
	wT
$s1$	$\alpha_T(s_1)$
$s2$	$\alpha_T(s_2)$
…	…
sn	$\alpha_T(s_n)$

s_e

Closeup

Computing cell $\alpha_{t+1}(s_i)$

- Multiply all α values of preceding column with their state transition probability!
- Sum them up!
- Multiply the sum with the emission probability of the actual state!



Pseudo Code for Forward Algorithm

```
# Given
# Set of states: Array S
# Start state: s0
# End state: se
# Symbol sequence: Array w
# State transition probabilities: Matrix a
# Symbol emission probabilities: Matrix b
# alpha: Matrix alpha

# Returns
# Total probability: p



# Initialisation F1
foreach s in S do
  alpha [1][s] := a[s0][s]*b[s][w[1]]
```

```
# Initialisation F1
foreach s in S do
  alpha [1][s] := a[s0][s]*b[s][w[1]]
done

# Induction F2
for i := 1 to length(w)-1 do
  foreach s in S do
    foreach s' in S do
      alpha[i+1][s] += alpha[i][s']*a[s'][s]
    done
    alpha[i+1][s] *= b[s][w[i+1]]
  done
done

# Termination F3
foreach s in S do
  p += alpha[length(w)]*a[s][se]
done

return p
```


Task III for Markov Models

Maximum probability of a emission sequence

- Which state sequence S has the maximal probability for a given emission sequence?

$$\arg\max_{S \in \mathcal{S}} P(W,S)$$

- Straight-forward solution
  - Enumerate all possible state sequences and take the maximum!

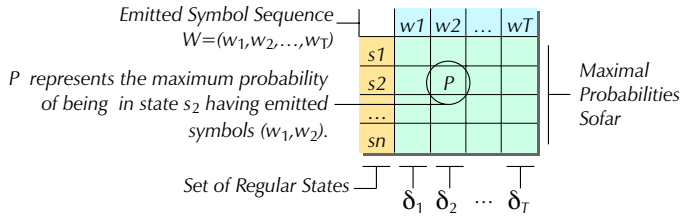
```
?- findall(P-S, prob([h,a,h,!],S,P), Pairs),
   max_key(Pairs,MaxP-MaxS).
```

- This again is naive! For the very same reason…
  - As there are $|S|^T$ possible state sequences for a emission sequence of length T

Length T	Sequences
1	$5.0*10^{1}$
3	$1.3*10^{5}$
5	$3.1*10^{8}$
10	$9.8*10^{16}$
30	$9.3*10^{50}$
50	$8.9*10^{84}$

Viterbi Algorithm & Delta

Delta, the caching data structure



The Viterbi Algorithm

- initializes delta in column δ_1.
- fills the delta matrix column by column from left to right.
- computes the final maximal probability from the last column δ_T.

Induction of Delta

V1: Initialization

$$\delta_1(s) = P(s \mid s_0)P(w_1 \mid s)$$

s_0

	w1
s1	$\delta_1(s_1)$
s2	$\delta_1(s_2)$
…	…
sn	$\delta_1(s_n)$

V2: Induction for $0 < t < T$

$$\delta_{t+1}(s) = \left(\max_{s' \in S} \delta_t(s')P(s \mid s')\right)P(w_{t+1} \mid s)$$

	…	wt	wt+1
s1	…	$\delta_t(s_1)$	$\delta_{t+1}(s_1)$
s2	…	$\delta_t(s_2)$	$\delta_{t+1}(s_2)$
…	…	…	…
sn	…	$\delta_t(s_n)$	$\delta_{t+1}(s_n)$

V3: Final computation

$$\max_{S \in S^T} P(S,W) = \max_{s \in S} \delta_T(s)P(s_e \mid s)$$

	wT
s1	$\delta_T(s_1)$
s2	$\delta_T(s_2)$
…	…
sn	$\delta_T(s_n)$

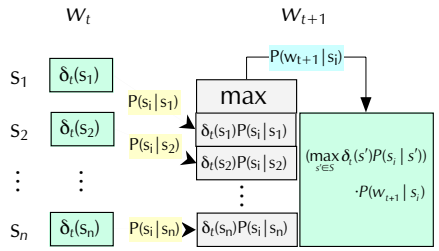
s_e

Closeup

Computing cell $\delta_{t+1}(s_i)$

- Multiply all δ values of preceding column with their state transition probability!
- Take the maximum!
- Multiply the sum with the emission probability of the actual state!

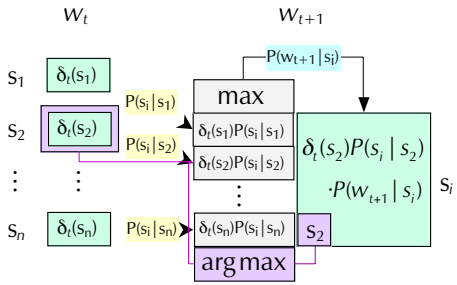
  - We do have the maximum probability. But what about the sequence...



Closeup with best incoming state

Remembering the state where the maximum probability came in.

- After the computation of the maximum, we know the state where the best path leading to s_i ended.
- If s_i will be part of the best sequence then s_2 will be its predecessor!



Reading Off the Best Sequence

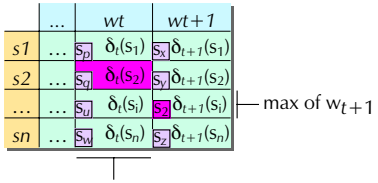
Work back from the end through delta

- Last state

$$s_T = \arg\max_{s \in S} \delta_T(s)P(s_e \mid s)$$

- The states before, for $T>t>0$

$$s_t = \arg\max_{s \in S} \delta_t(s)P(s_{t+1} \mid s)$$



therefore, s_2 is best previous state

Storing incoming states

- To avoid recomputation, we can store arg max in an own matrix psi while constructing delta.

$$\psi_{t+1}(s) = \arg\max_{s' \in S} \delta_t(s')P(s \mid s')$$

Applications of Markov Models

Timeline

- Historically
  - A. A. Markov 1913 for letter sequence models in Russian literature
- Since the 40ies
  - General models (stochastic processes) in statistics, physics, etc.
  - Theory of HMM late 70ies
- Since the 70ies
  - HMMs for speech recognition
- Since the 80ies
  - PoS-Tagging
- Since the 90ies
  - Chunking, Partial Parsing, Named Entity Recognition, ...
  - Bioinformatics, Economics,...