Type-Based MCMC

NAACL 2010 – Los Angeles

Percy Liang Michael I. Jordan Dan Klein

University of California
Berkeley

Type-Based MCMC

NAACL 2010 – Los Angeles

Percy Liang Michael I. Jordan Dan Klein

Learning latent-variable models

$\cdots$ *some stocks rose* $\cdots$

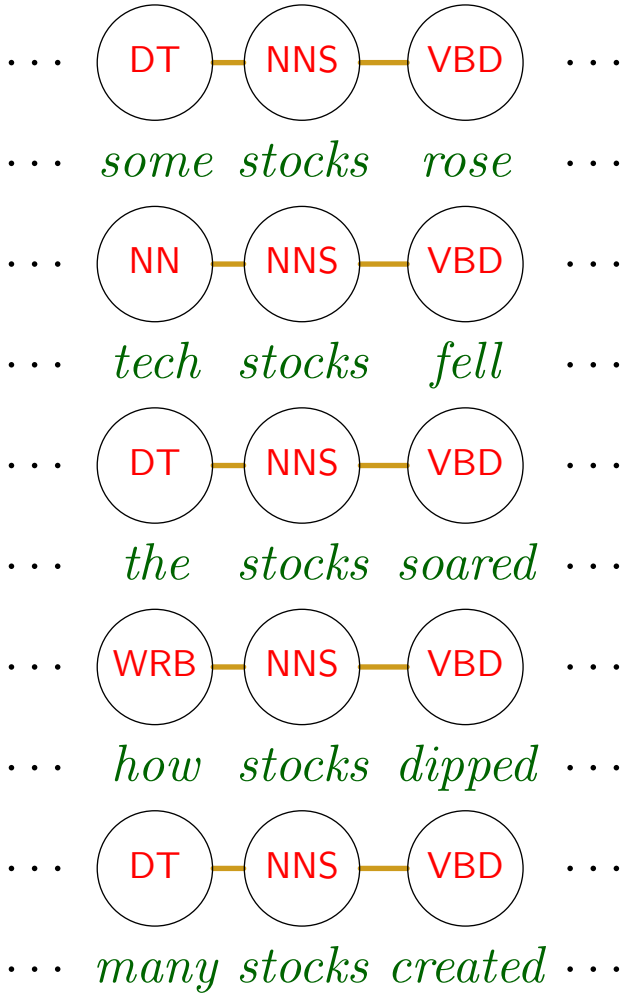
$\cdots$ *tech stocks fell* $\cdots$

$\cdots$ *the stocks soared* $\cdots$

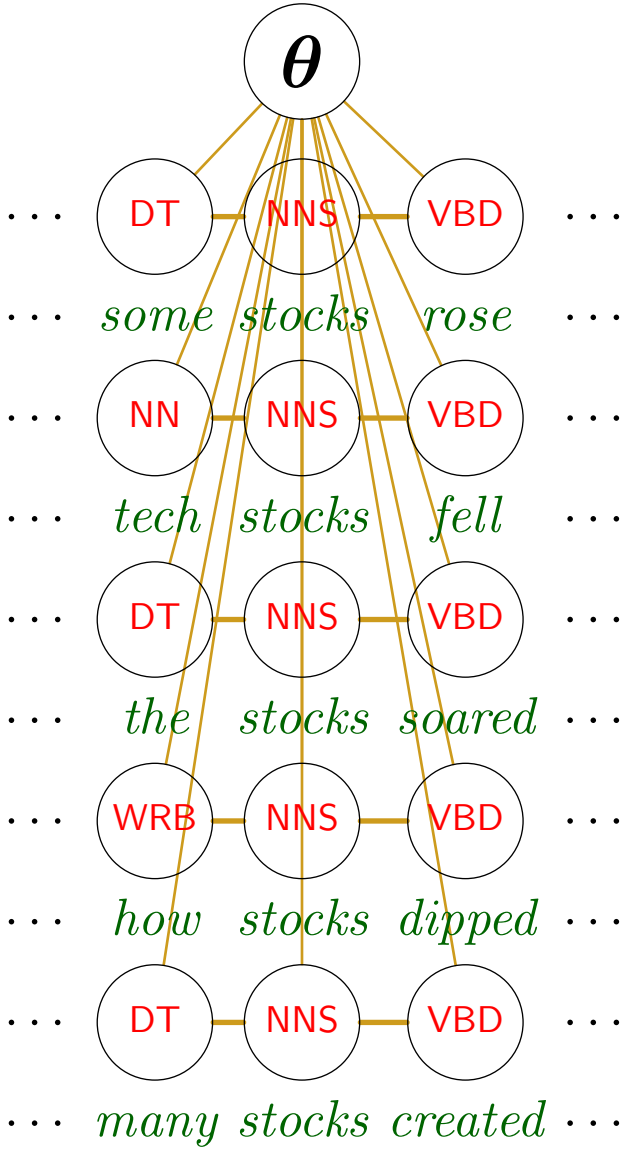
$\cdots$ *how stocks dipped* $\cdots$

$\cdots$ *many stocks created* $\cdots$

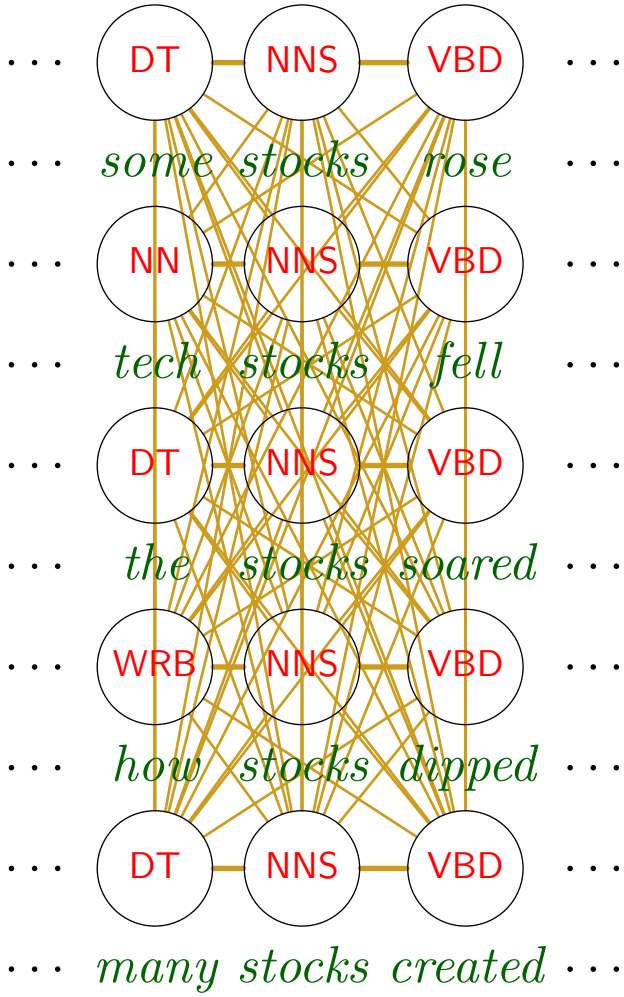
Learning latent-variable models



Learning latent-variable models

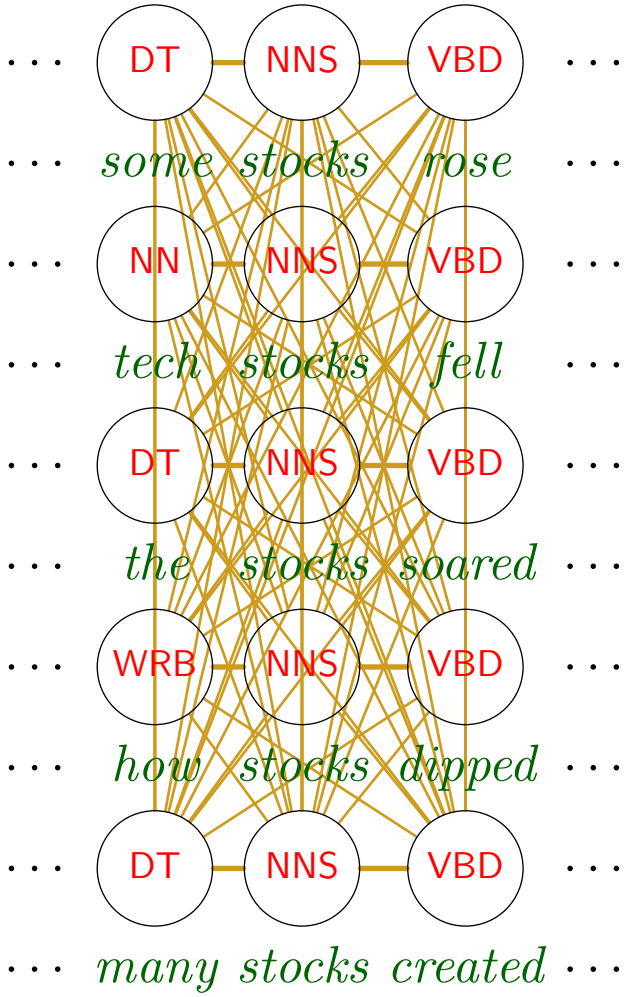


Learning latent-variable models



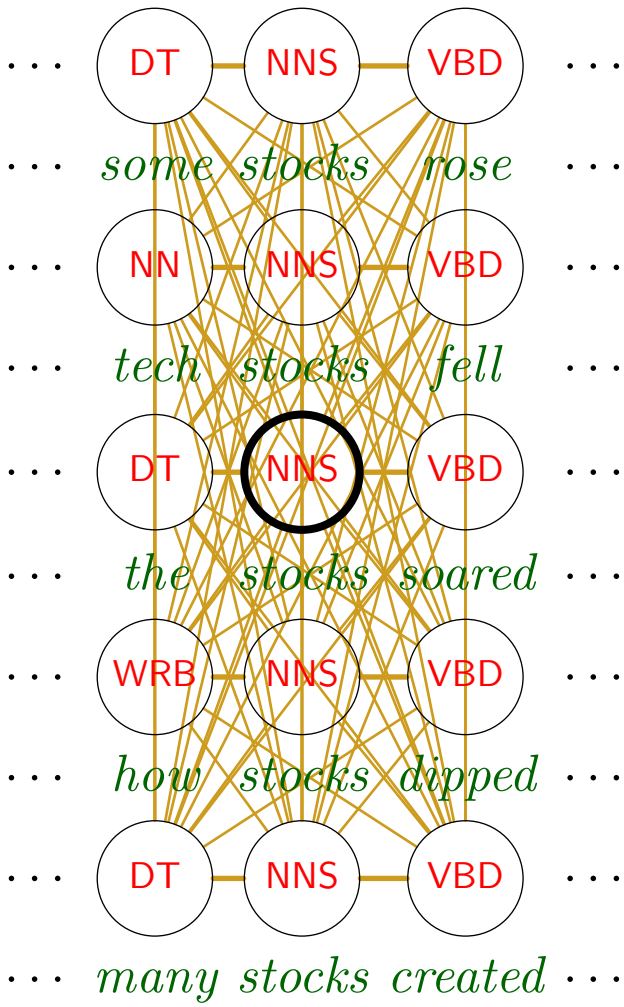
Learning latent-variable models

Variables are highly dependent



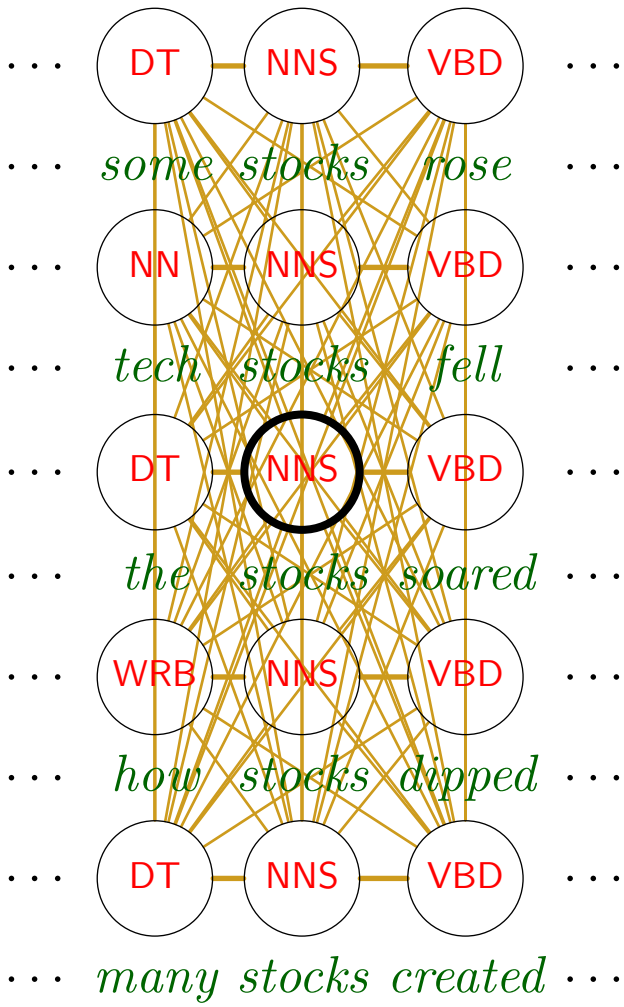
Learning latent-variable models

Variables are highly dependent



Simple: token-based (e.g., Gibbs sampler)

Learning latent-variable models

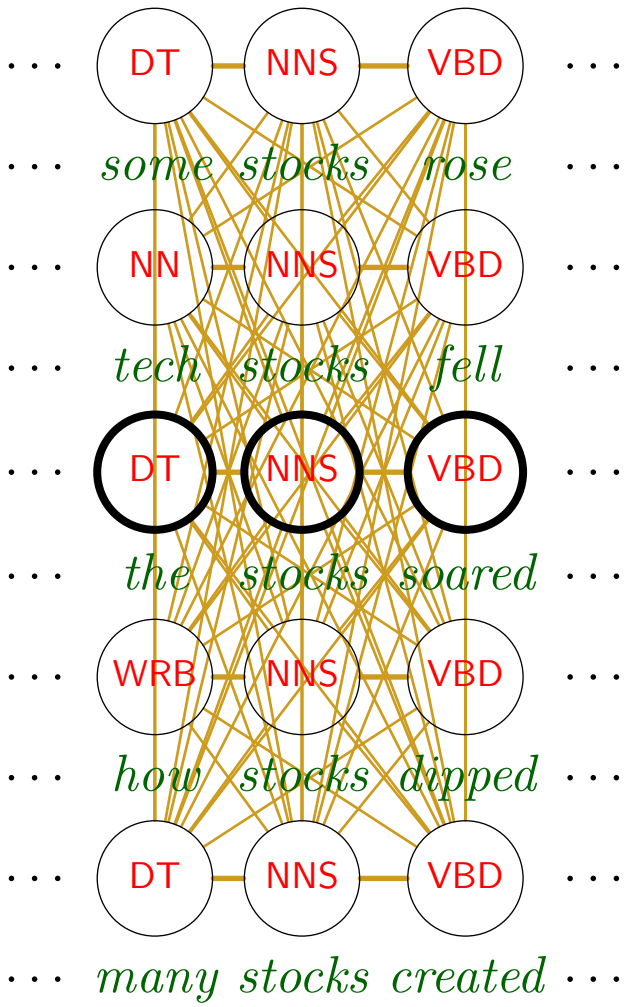


Variables are highly dependent

Simple: token-based (e.g., Gibbs sampler)

$\Rightarrow$ local optima, slow mixing

Learning latent-variable models

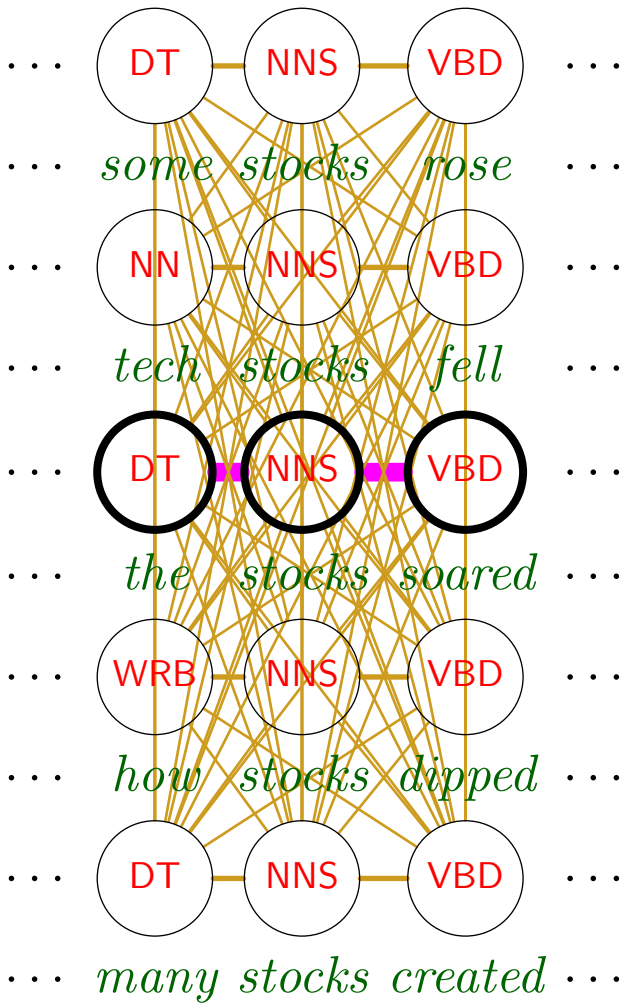


Variables are highly dependent

Simple: token-based (e.g., Gibbs sampler)
$\Rightarrow$ local optima, slow mixing

Classic: sentence-based (e.g., EM)

Learning latent-variable models



Variables are highly dependent

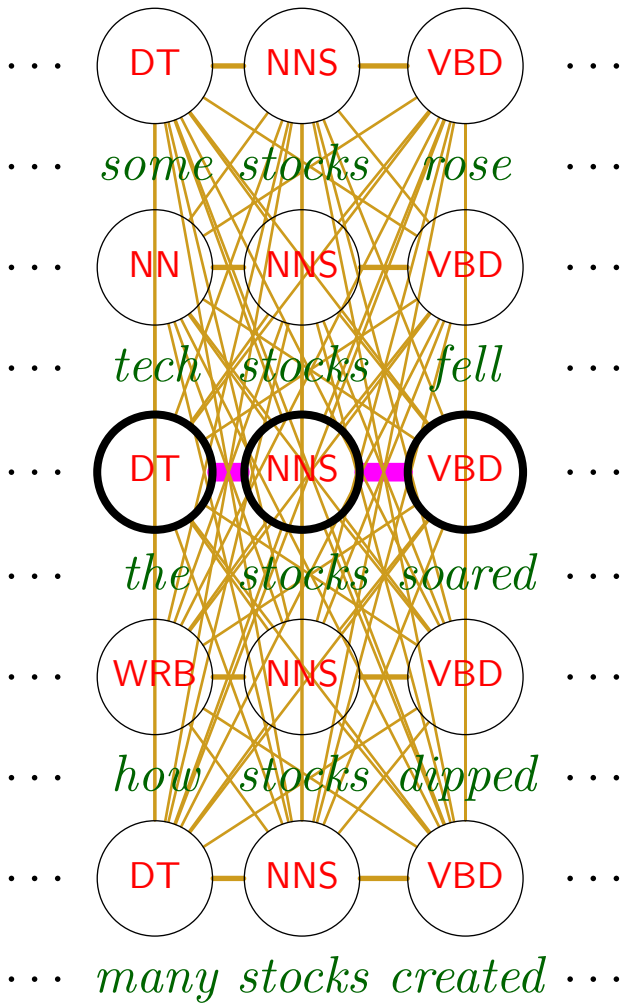
Simple: token-based (e.g., Gibbs sampler)

$\Rightarrow$ local optima, slow mixing

Classic: sentence-based (e.g., EM)

Structural dependencies

Learning latent-variable models



Variables are highly dependent

Simple: token-based (e.g., Gibbs sampler)

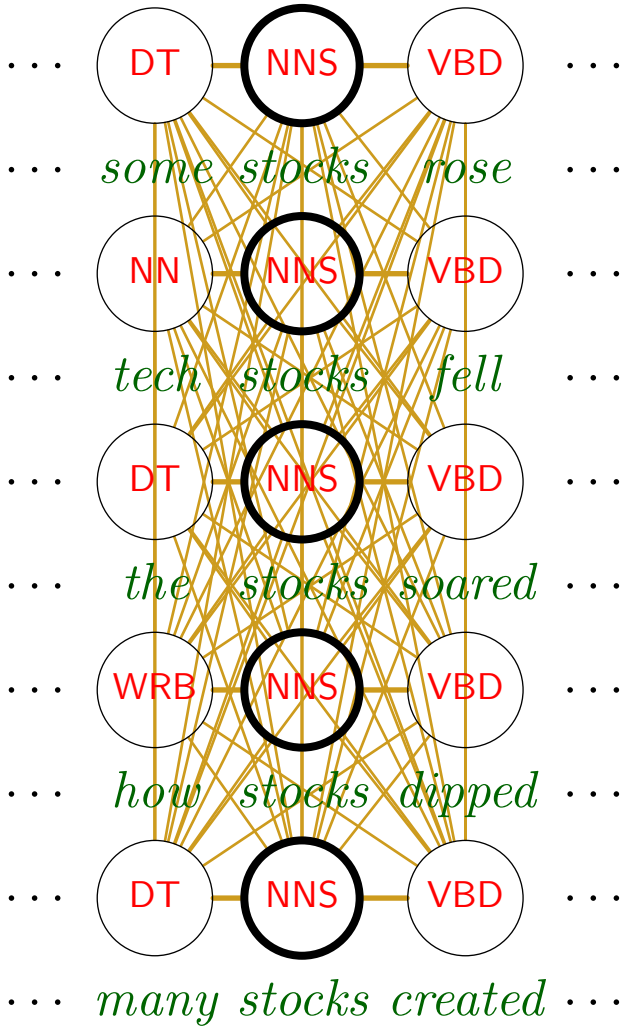
$\Rightarrow$ local optima, slow mixing

Classic: sentence-based (e.g., EM)

 Structural dependencies

 dynamic programming

Learning latent-variable models



Variables are highly dependent

Simple: token-based (e.g., Gibbs sampler)

$\Rightarrow$ local optima, slow mixing

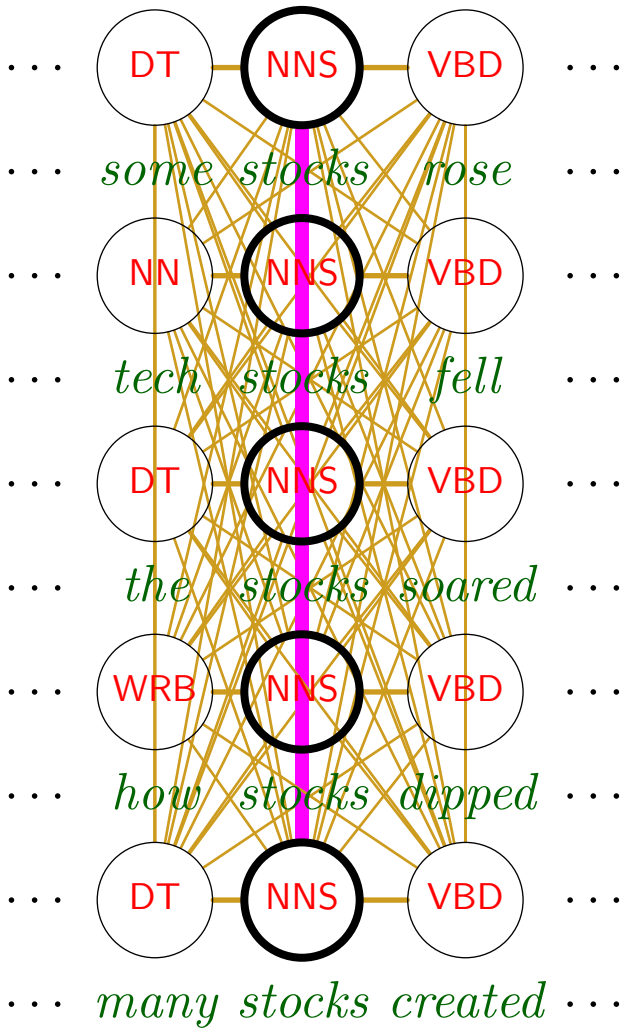
Classic: sentence-based (e.g., EM)

 Structural dependencies

 dynamic programming

New: type-based

Learning latent-variable models



Variables are highly dependent

Simple: token-based (e.g., Gibbs sampler)

$\Rightarrow$ local optima, slow mixing

Classic: sentence-based (e.g., EM)

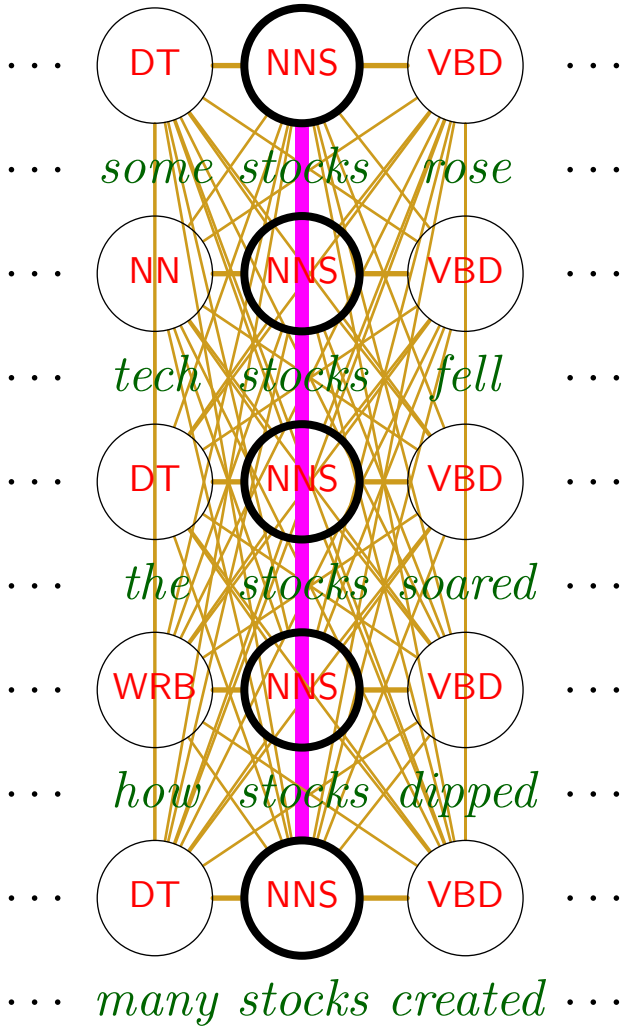
 Structural dependencies

 dynamic programming

New: type-based

 Parameter dependencies

Learning latent-variable models



Variables are highly dependent

Simple: token-based (e.g., Gibbs sampler)

$\Rightarrow$ local optima, slow mixing

Classic: sentence-based (e.g., EM)

 Structural dependencies

 dynamic programming

New: type-based

 Parameter dependencies

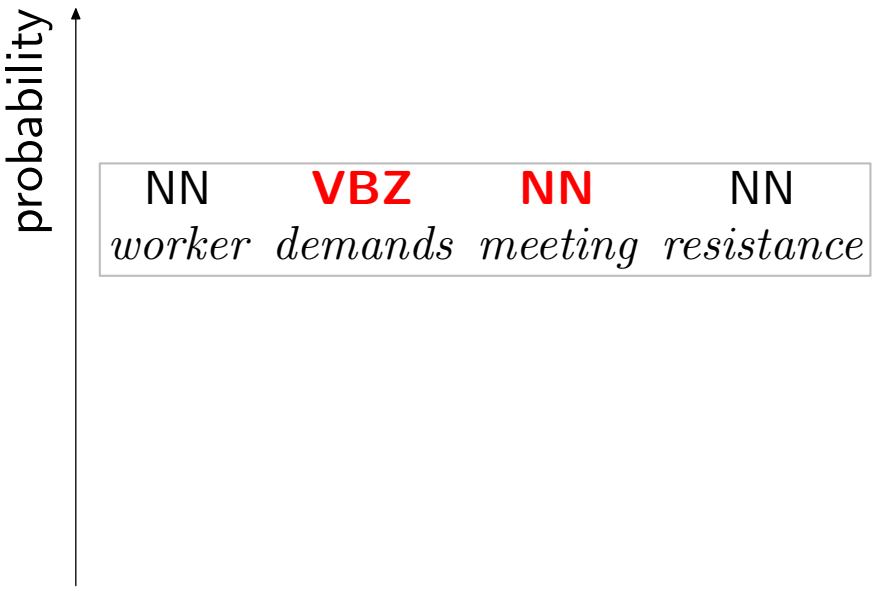
 **exchangeability**

Structural dependencies

Dependencies between adjacent variables

Structural dependencies

Dependencies between adjacent variables



Structural dependencies

Dependencies between adjacent variables

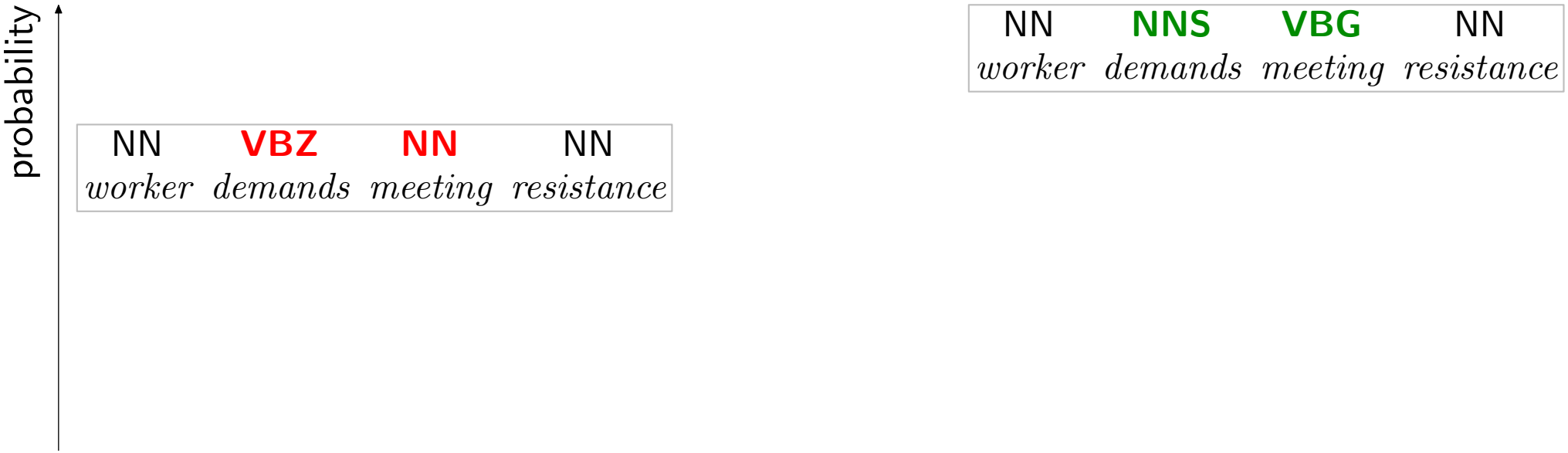
probability

NN	**VBZ**	**NN**	NN
worker	*demands*	*meeting*	*resistance*

NN	**NNS**	**VBG**	NN
worker	*demands*	*meeting*	*resistance*

Structural dependencies

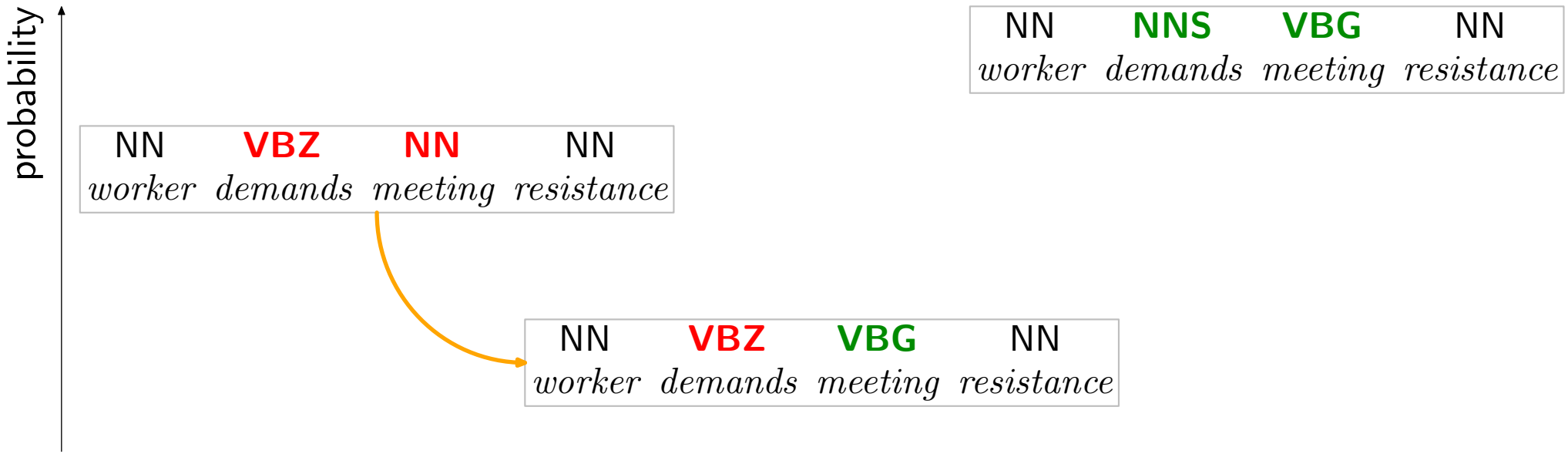
Dependencies between adjacent variables



Token-based: update only one variable at a time

Structural dependencies

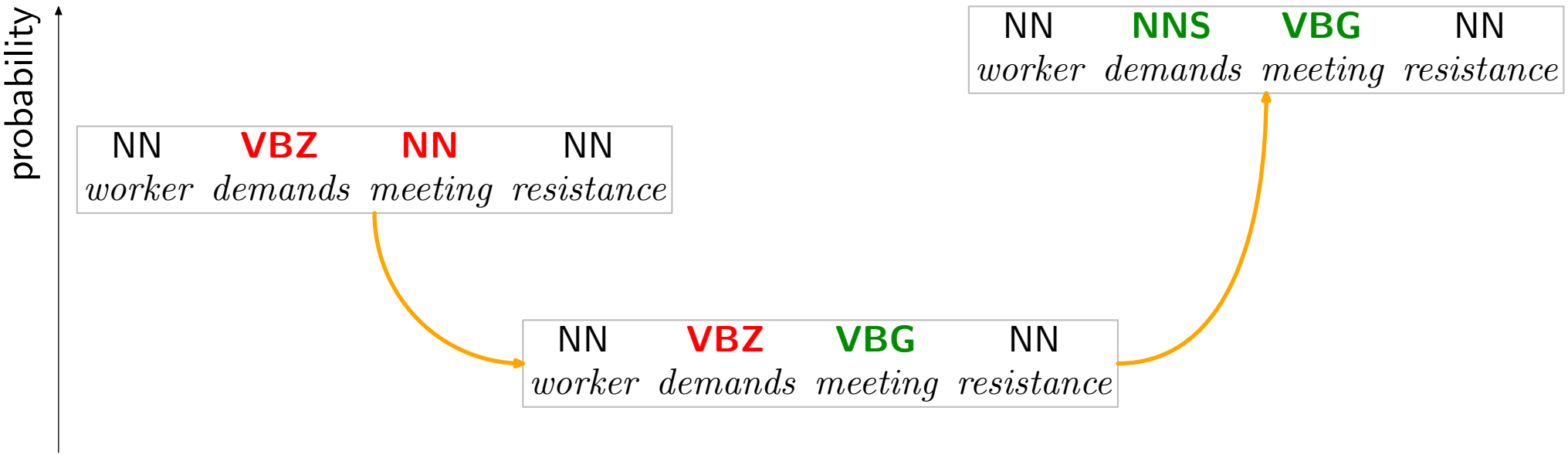
Dependencies between adjacent variables



Token-based: update only one variable at a time

Structural dependencies

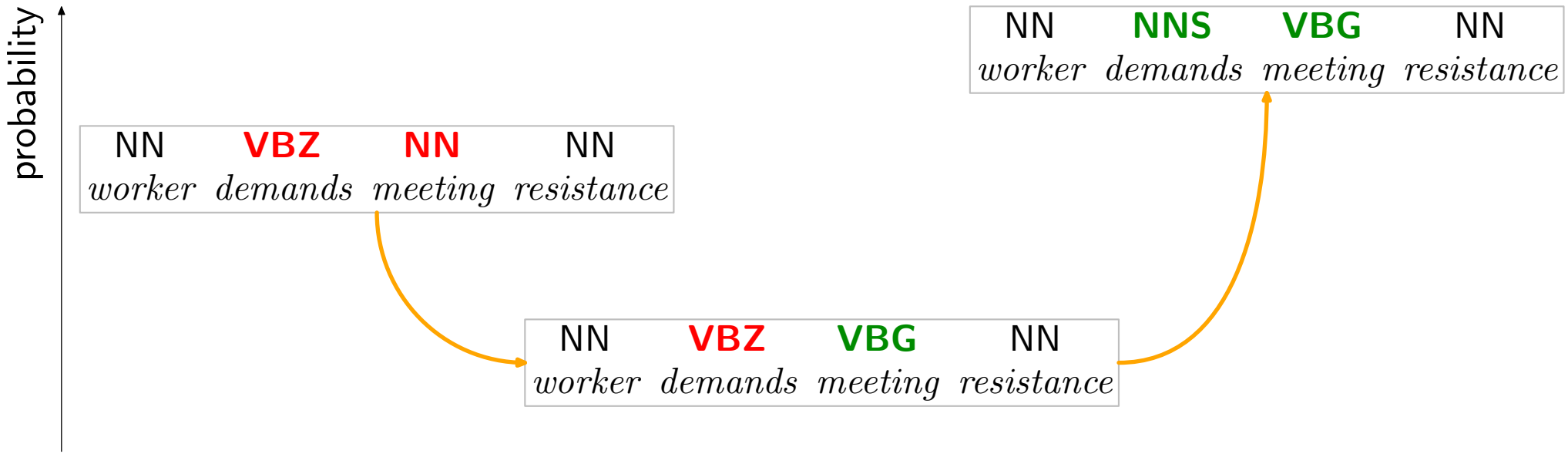
Dependencies between adjacent variables



Token-based: update only one variable at a time

Structural dependencies

Dependencies between adjacent variables



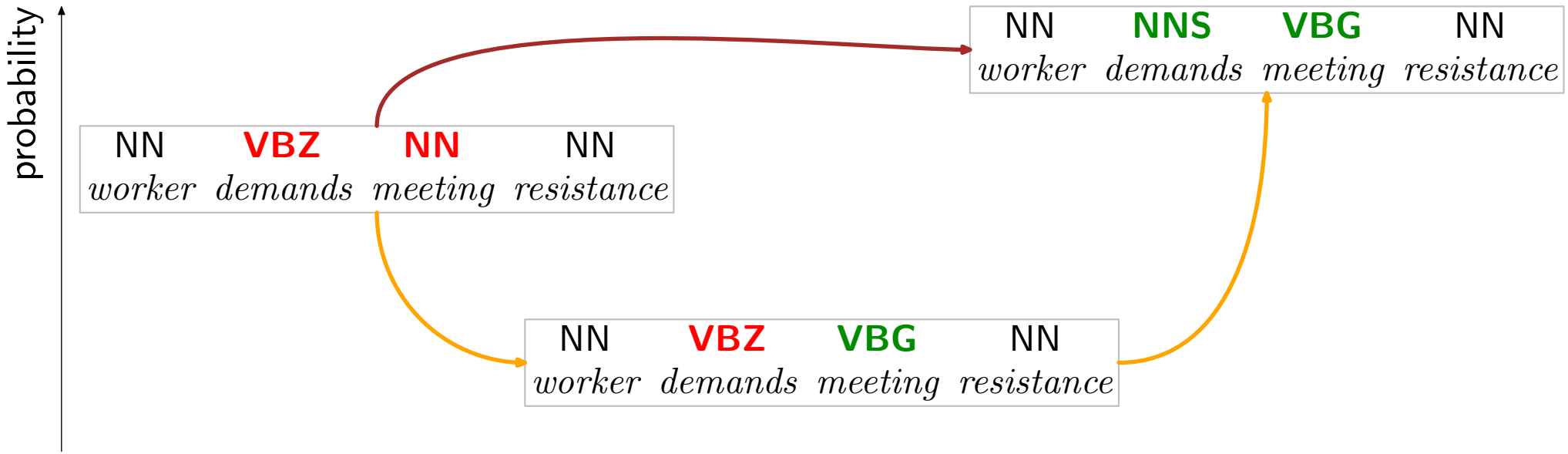
Token-based: update only one variable at a time

Problem: need to go downhill before uphill

Structural dependencies

Dependencies between adjacent variables

Sentence-based: update all variables in a sentence



Token-based: update only one variable at a time

Problem: need to go downhill before uphill

Parameter dependencies

Dependencies between variables with shared parameters

Parameter dependencies

Dependencies between variables with shared parameters

probability

VBZ	VBD	...
stocks	*rose*	...
VBZ	VBD	...
stocks	*fell*	...
VBZ	VBD	...
stocks	*took*	...
VBZ	VBD	...
stocks	*shot*	...

Parameter dependencies

Dependencies between variables with shared parameters

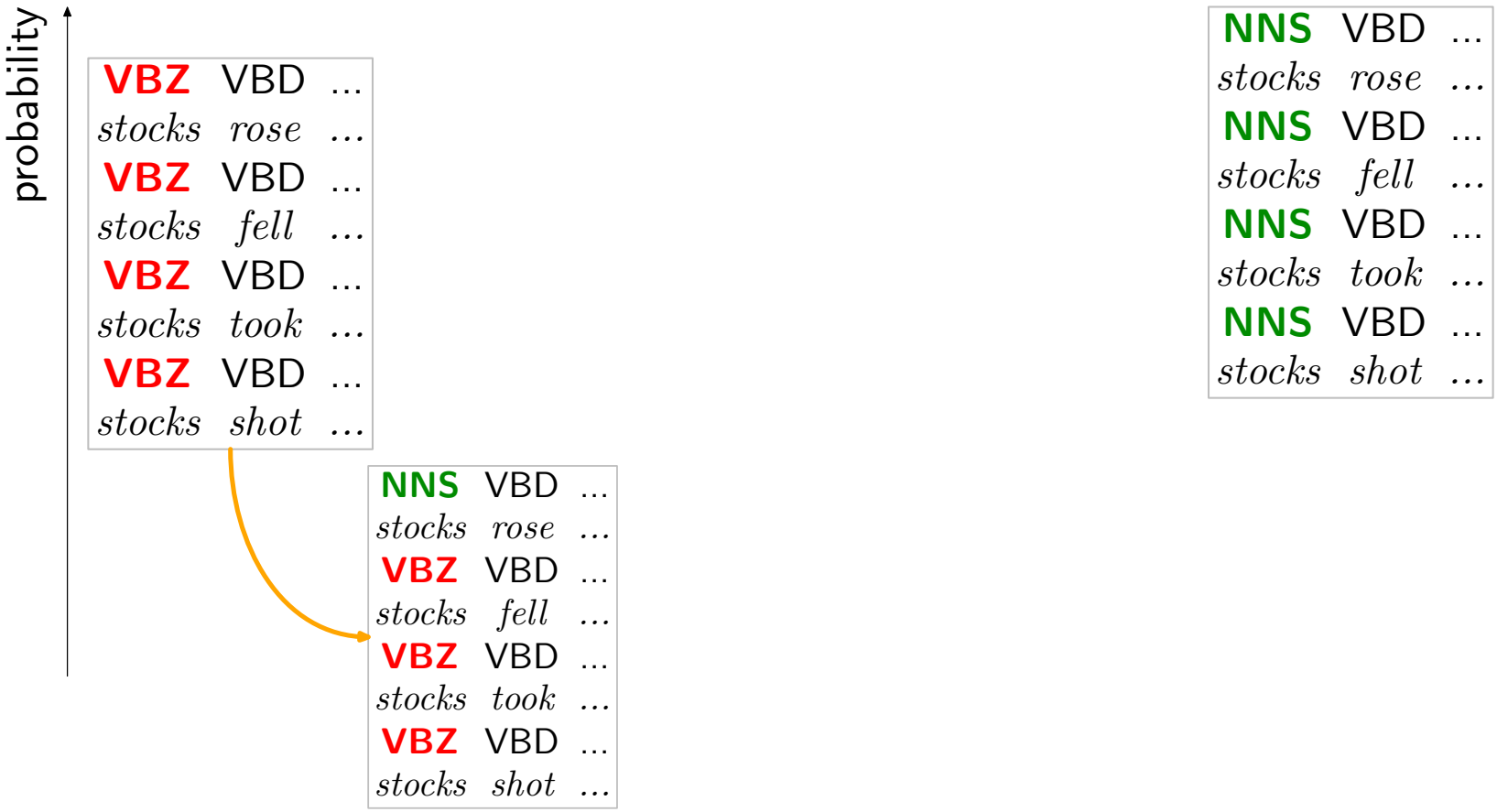
probability

VBZ	VBD	...
stocks	*rose*	...
VBZ	VBD	...
stocks	*fell*	...
VBZ	VBD	...
stocks	*took*	...
VBZ	VBD	...
stocks	*shot*	...

NNS	VBD	...
stocks	*rose*	...
NNS	VBD	...
stocks	*fell*	...
NNS	VBD	...
stocks	*took*	...
NNS	VBD	...
stocks	*shot*	...

Parameter dependencies

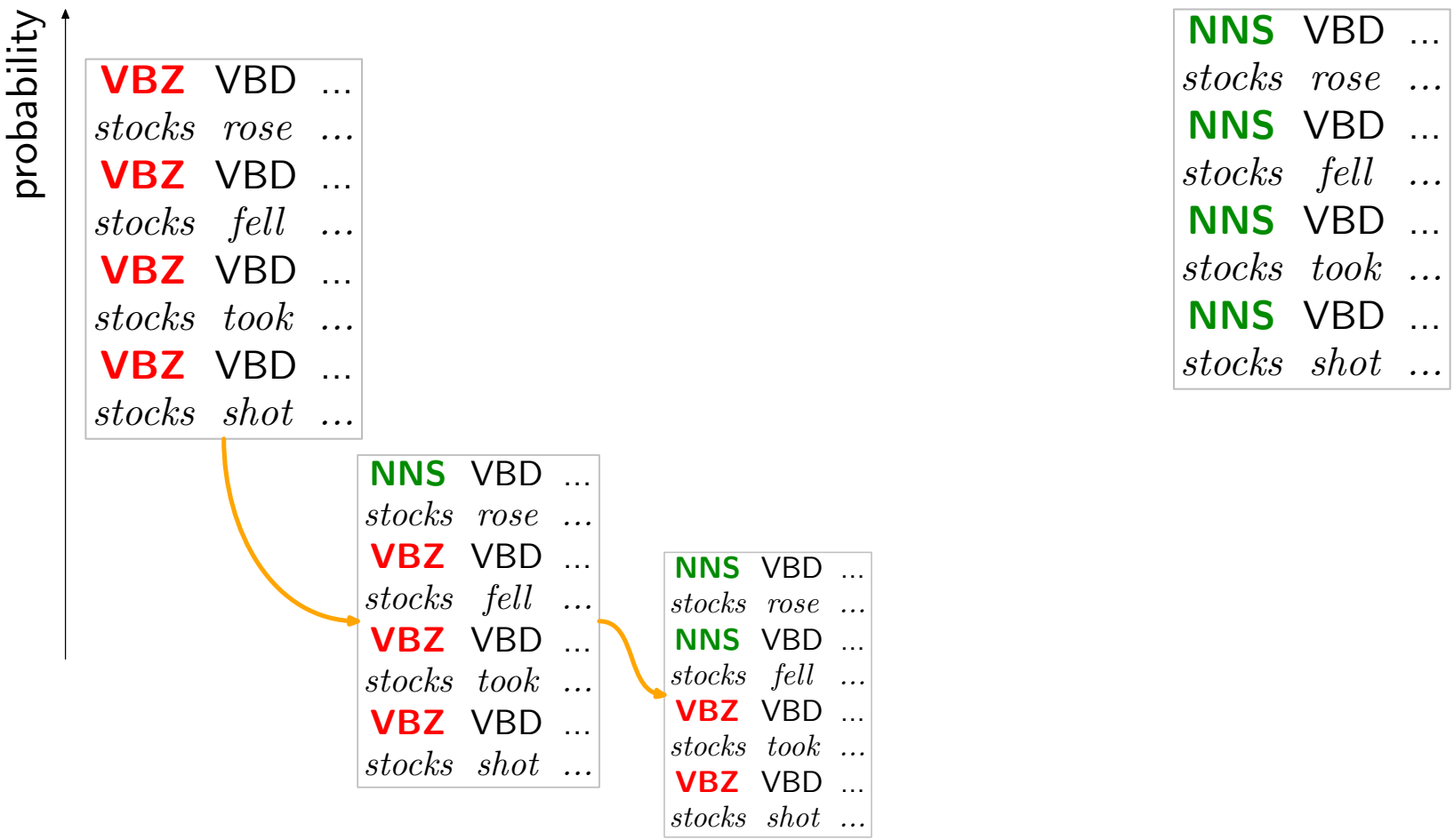
Dependencies between variables with shared parameters



Token-based:

Parameter dependencies

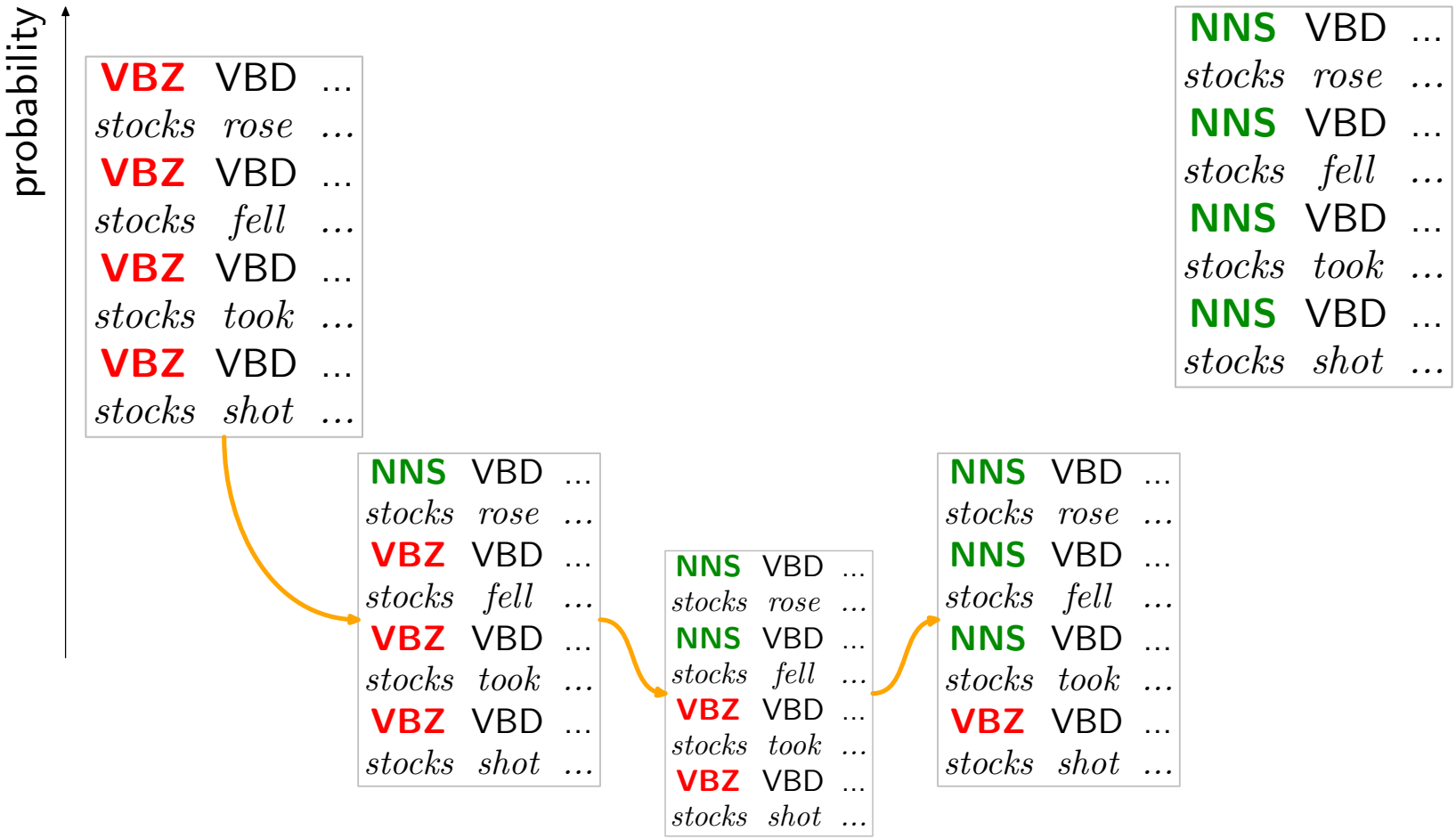
Dependencies between variables with shared parameters



Token-based:

Parameter dependencies

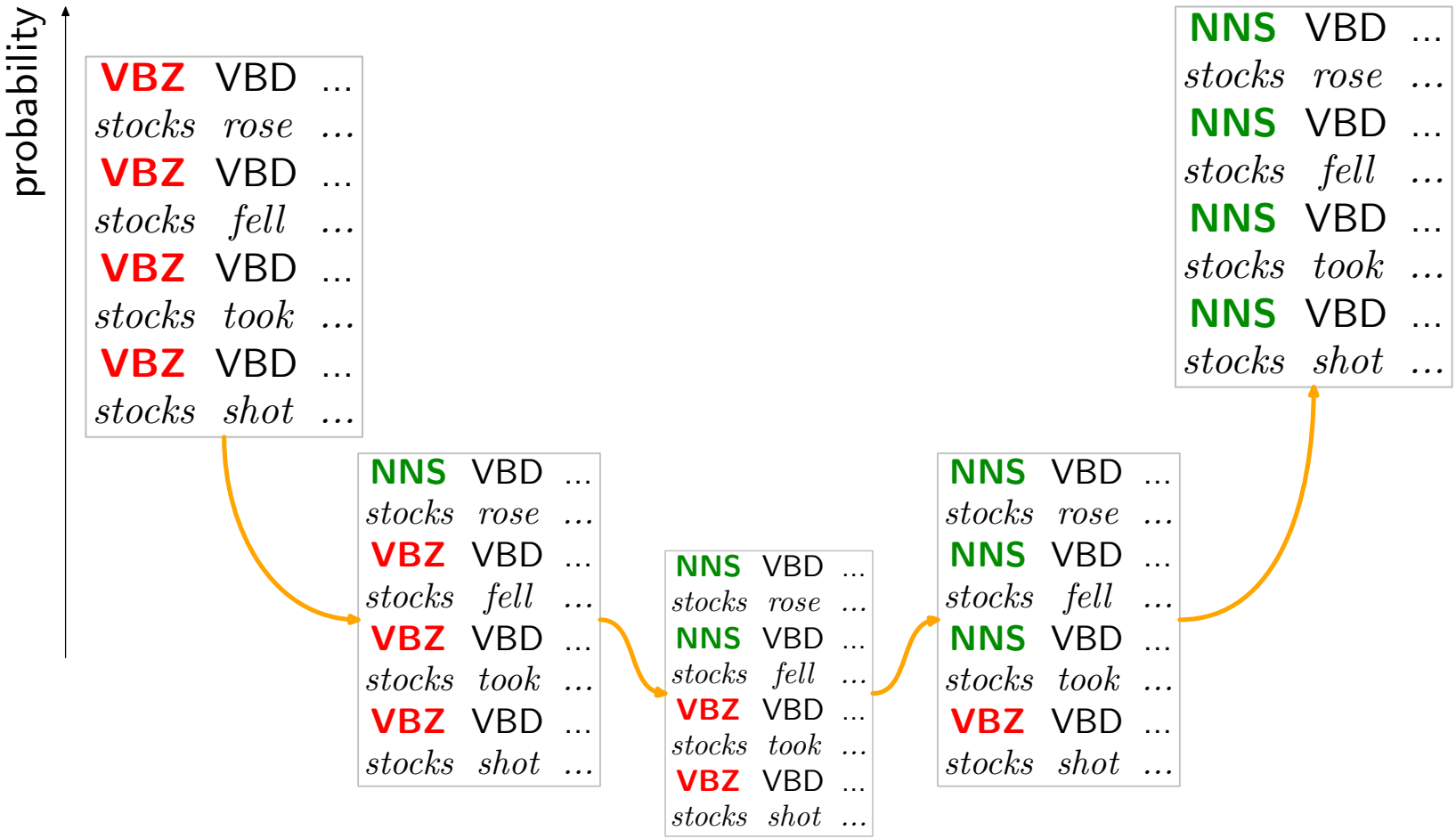
Dependencies between variables with shared parameters



Token-based:

Parameter dependencies

Dependencies between variables with shared parameters

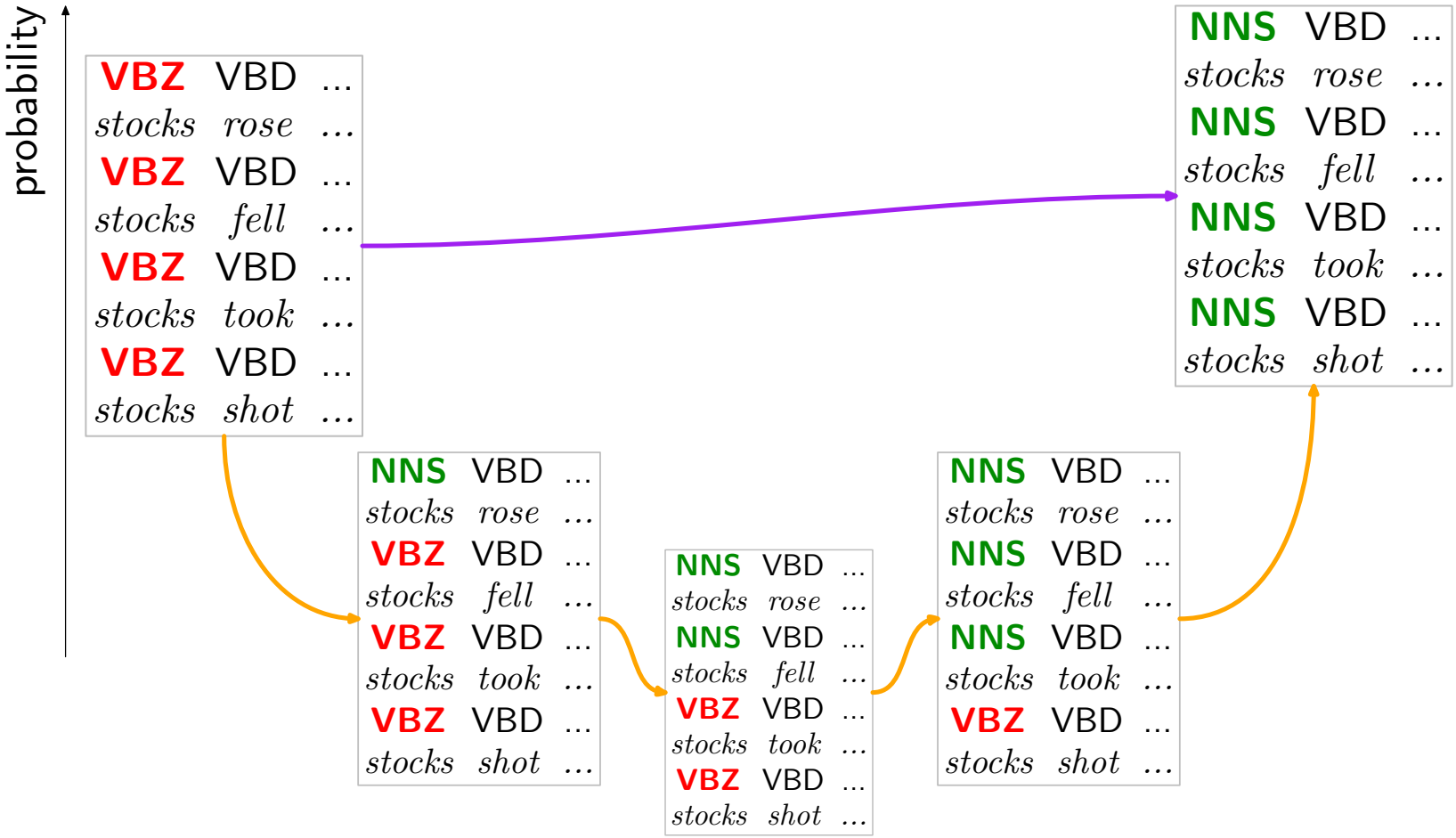


Token-based: need to go downhill **a lot** before going uphill

Parameter dependencies

Dependencies between variables with shared parameters

Type-based: update all variables of one **type**

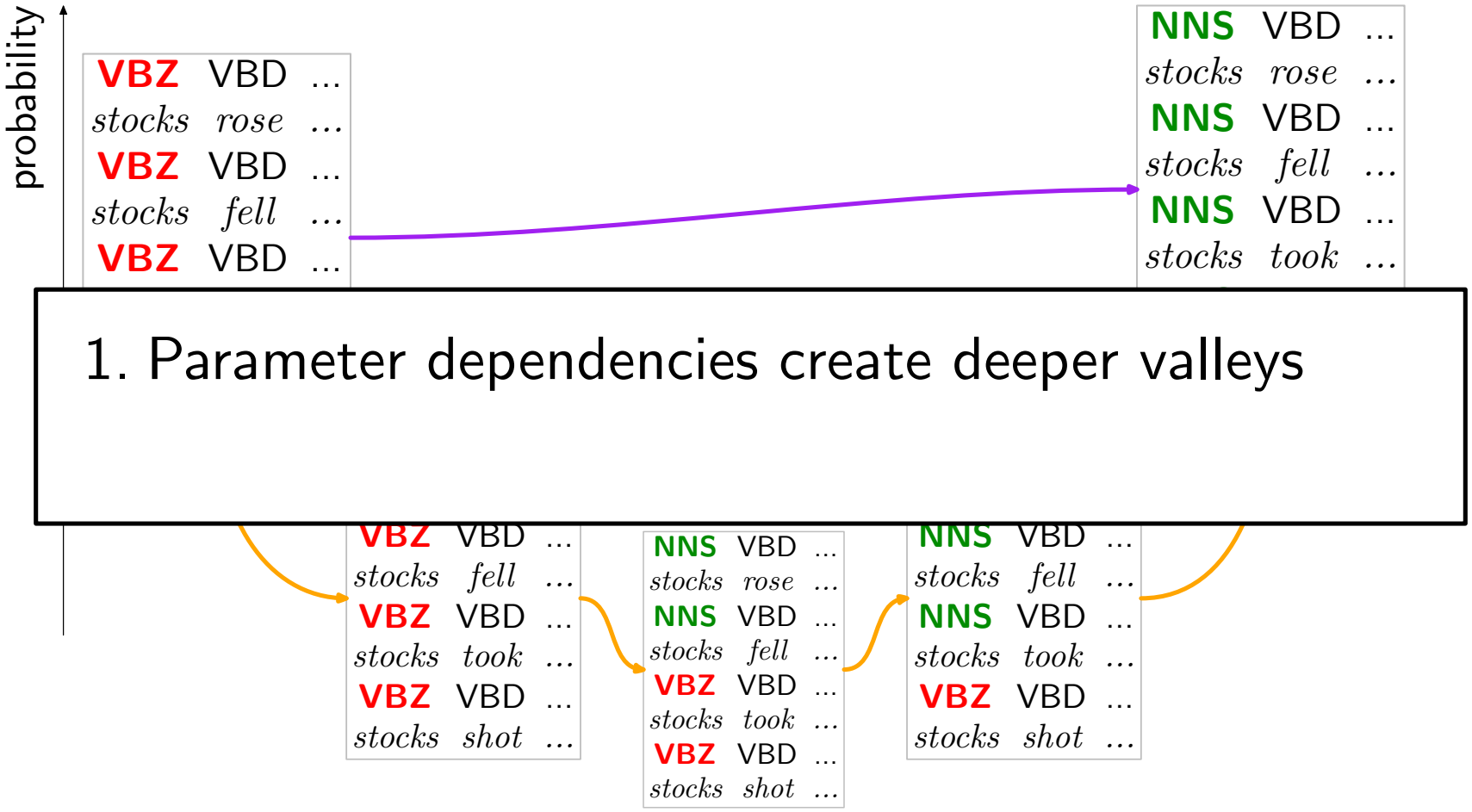


Token-based: need to go downhill **a lot** before going uphill

Parameter dependencies

Dependencies between variables with shared parameters

Type-based: update all variables of one **type**



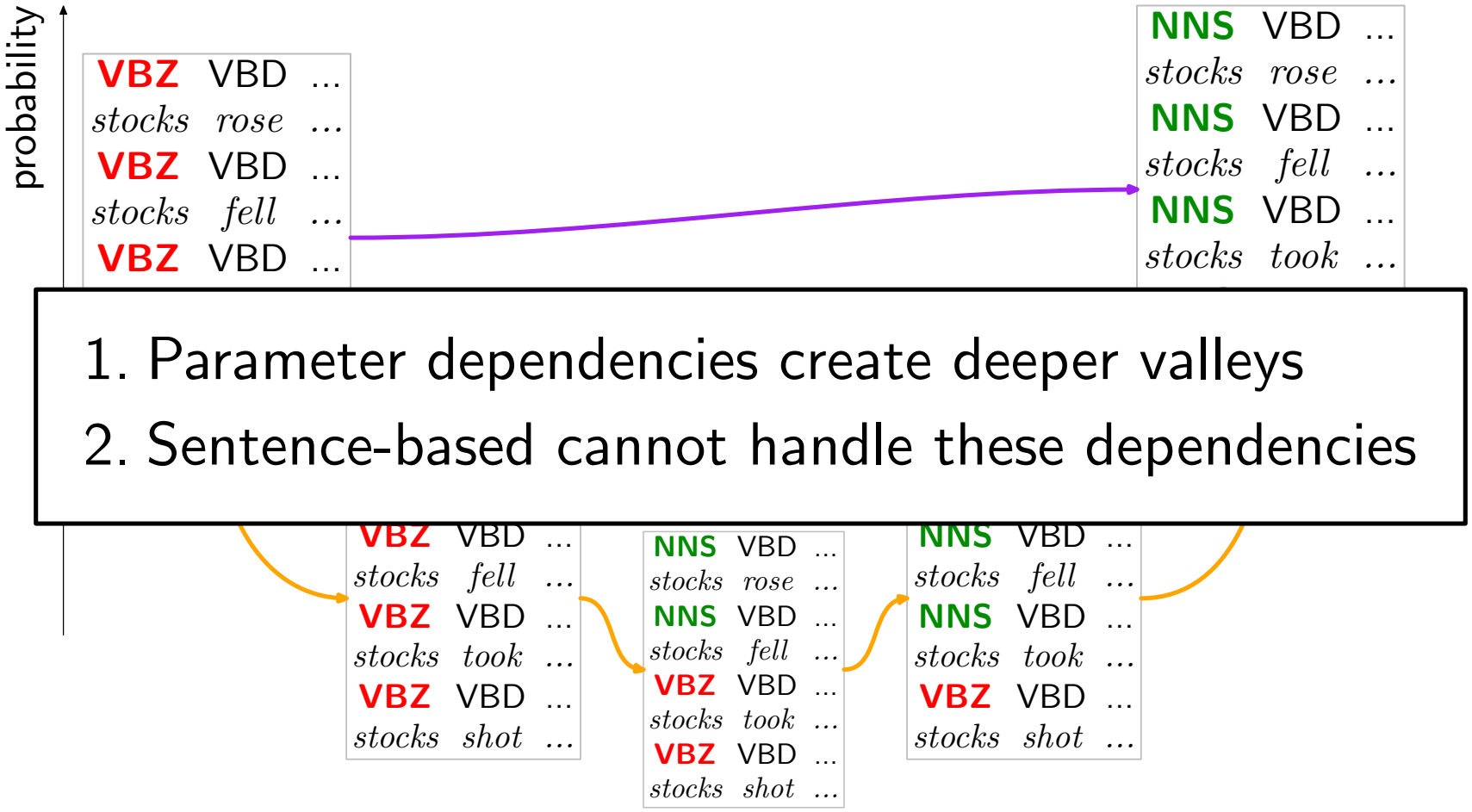
1. Parameter dependencies create deeper valleys

Token-based: need to go downhill **a lot** before going uphill

Parameter dependencies

Dependencies between variables with shared parameters

Type-based: update all variables of one **type**



1. Parameter dependencies create deeper valleys
2. Sentence-based cannot handle these dependencies

Token-based: need to go downhill **a lot** before going uphill

What exactly is a **type**?

How can we update all variables of a type **efficiently**?

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

$\theta_{\mathsf{T}:\mathsf{V},\mathsf{N}}$: from state V, probability of transitioning to state N

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

$\theta_{\mathsf{T}:\mathsf{V},\mathsf{N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

$\theta_{\mathsf{T}:\mathsf{V},\mathsf{N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

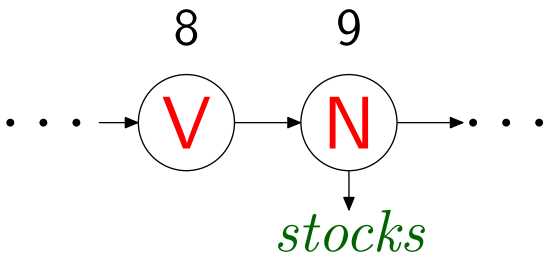
Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

$\theta_{\mathsf{T}:\mathsf{V},\mathsf{N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables



Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

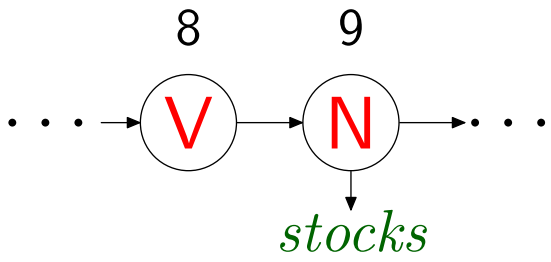
$\theta_{\mathsf{T}:\mathsf{V},\mathsf{N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

$z \in \mathbf{z}$

[T : V, N at 8]



Formal setup

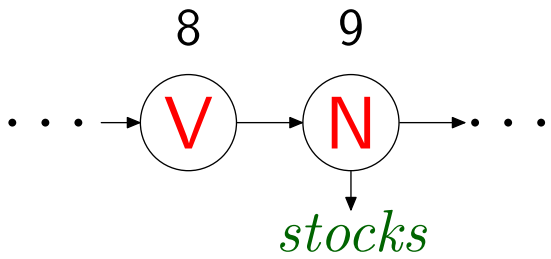
Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

$\theta_{\text{T:V,N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

$z \in \mathbf{z}$	$j(z)$ [parameter used]
[T : V, N at 8]	T:V,N



Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

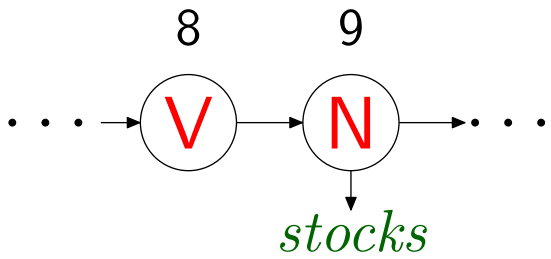
$\theta_{\text{T:V,N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

$z \in \mathbf{z}$ $j(z)$ [parameter used]

[T : V, N at 8] T:V,N



$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_{z \in \mathbf{z}} \theta_{j(z)}$ [likelihood]

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

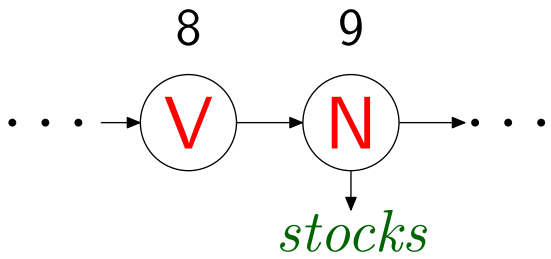
$\theta_{\text{T:V,N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

$z \in \mathbf{z}$ $j(z)$ [parameter used]

[T : V, N at 8] T:V,N



$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_{z \in \mathbf{z}} \theta_{j(z)}$ [likelihood]

$p(\mathbf{z}) = \int p(\mathbf{z} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$ [marginal likelihood]

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

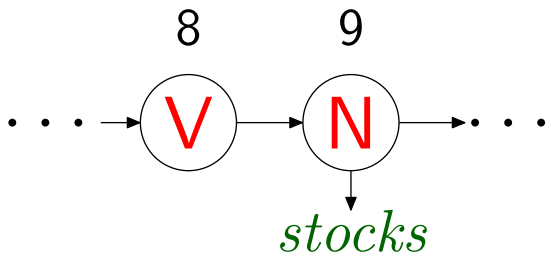
$\theta_{\text{T:V,N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

$z \in \mathbf{z}$ $j(z)$ [parameter used]

[T : V, N at 8] T:V,N



$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_{z \in \mathbf{z}} \theta_{j(z)}$ [likelihood]

$p(\mathbf{z}) = \int p(\mathbf{z} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$ [marginal likelihood]

Observations $\mathbf{x}$: observed part of $\mathbf{z}$ (e.g., the words)

Formal setup

Parameters $\boldsymbol{\theta}$: vector of conditional probabilities

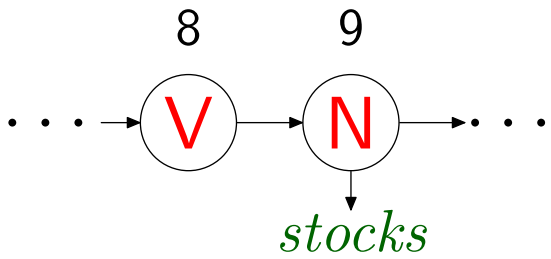
$\theta_{\text{T:V,N}}$: from state V, probability of transitioning to state N

$p(\boldsymbol{\theta})$ is product of Dirichlets [prior]

Choices $\mathbf{z}$: specifies values of latent and observed variables

$z \in \mathbf{z}$ $j(z)$ [parameter used]

[T : V, N at 8] T:V,N



$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_{z \in \mathbf{z}} \theta_{j(z)}$ [likelihood]

$p(\mathbf{z}) = \int p(\mathbf{z} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$ [marginal likelihood]

Observations $\mathbf{x}$: observed part of $\mathbf{z}$ (e.g., the words)

Goal: sample from $p(\mathbf{z} \mid \mathbf{x})$ [not $p(\mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta})$]

Exchangeability

Sufficient statistics $\mathbf{n}$: $\#$ times parameters were used in $\mathbf{z}$

Exchangeability

Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\mathsf{T:V,N}}$: # times that state V transitioned to state N

Exchangeability

Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\mathsf{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j}$$

Exchangeability

Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j} =$ simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

Exchangeability

Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j} =$ simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z}) =$ simple function of $\mathbf{n}$

Exchangeability

Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j} =$ simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z}) =$ simple function of $\mathbf{n}$ (key: **exchangeability**)

Exchangeability

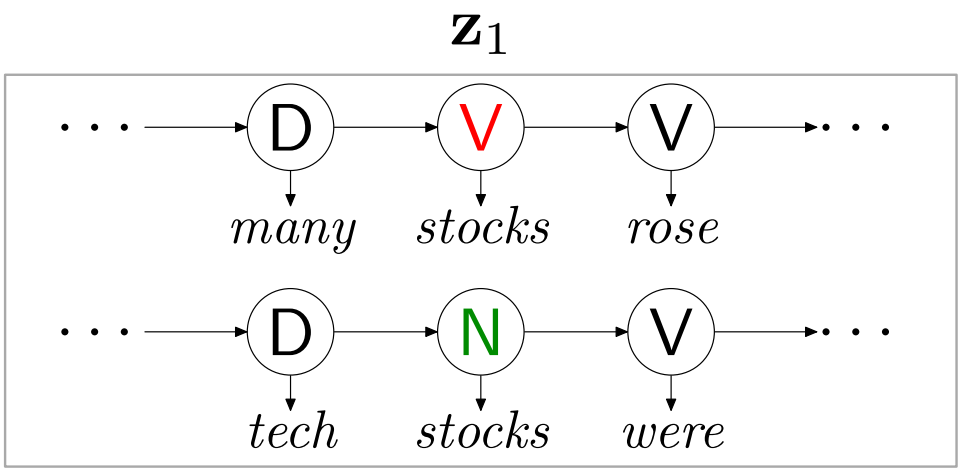
Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j} =$ simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z}) =$ simple function of $\mathbf{n}$ (key: **exchangeability**)



Exchangeability

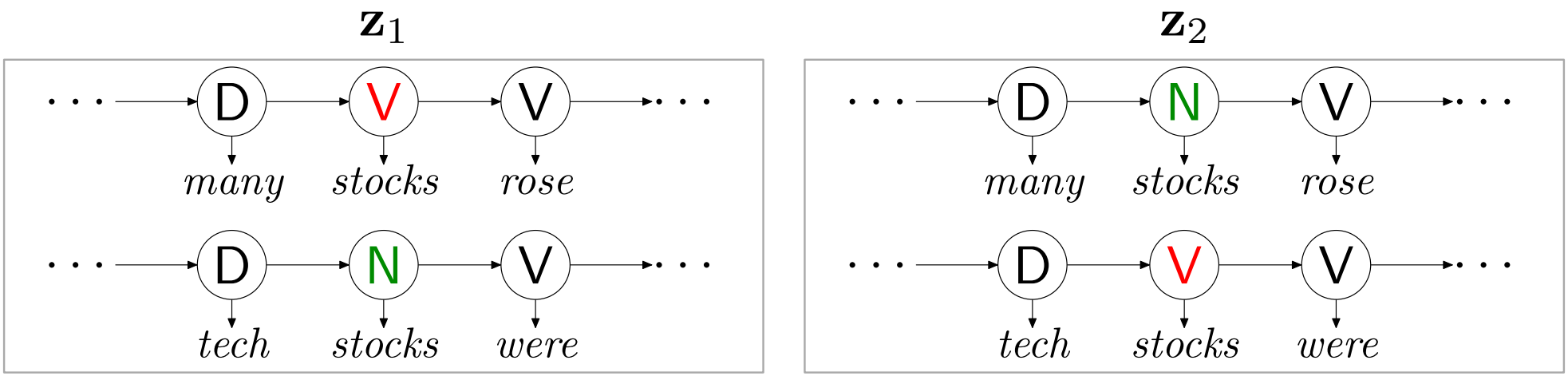
Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j} =$ simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z}) =$ simple function of $\mathbf{n}$ (key: **exchangeability**)



Exchangeability

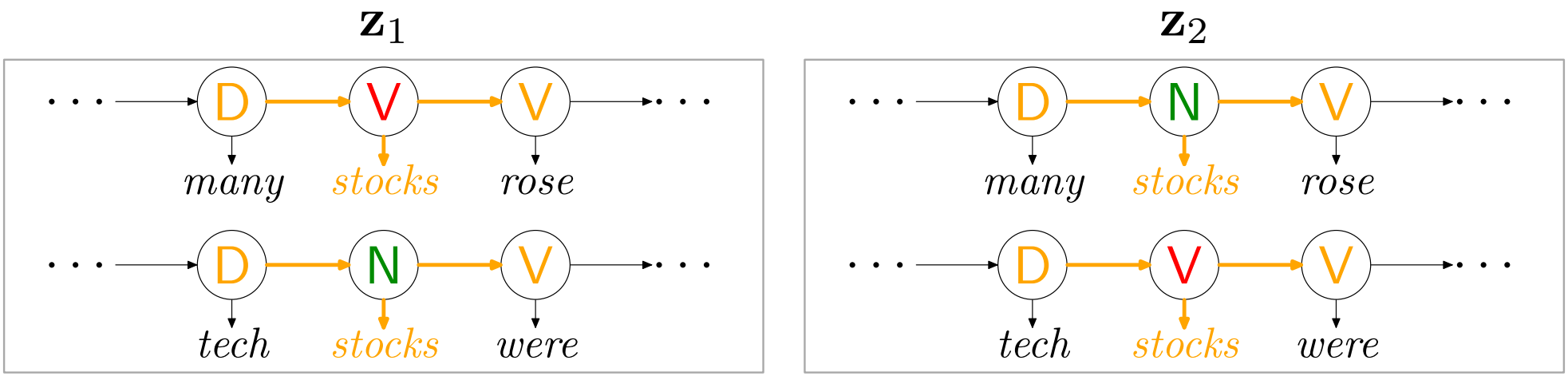
Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j}$ = simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z})$ = simple function of $\mathbf{n}$ (key: **exchangeability**)



Exchangeability

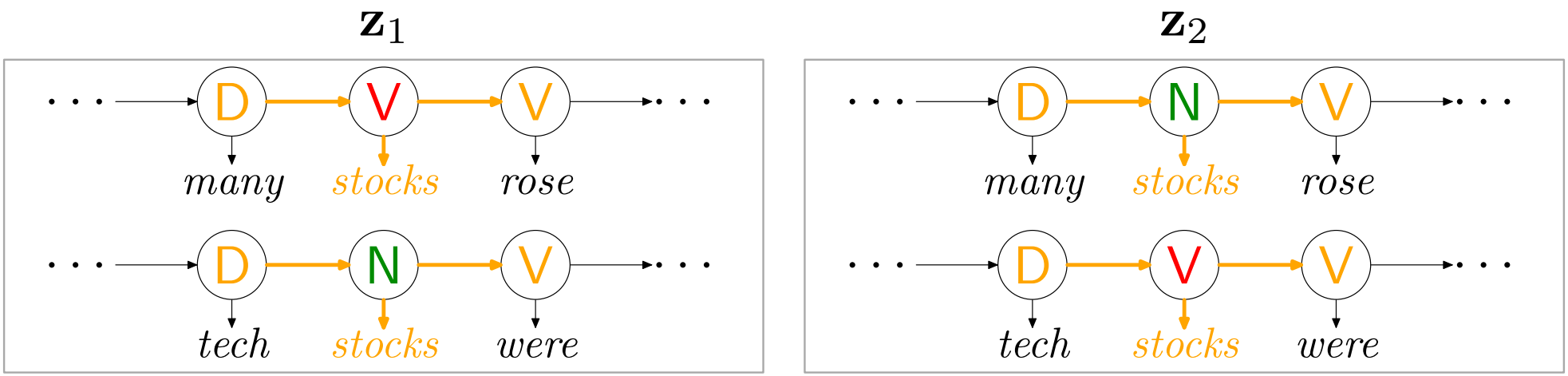
Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j}$ = simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z})$ = simple function of $\mathbf{n}$ (key: **exchangeability**)



$\mathbf{z}_1$ and $\mathbf{z}_2$ have same sufficient statistics

Exchangeability

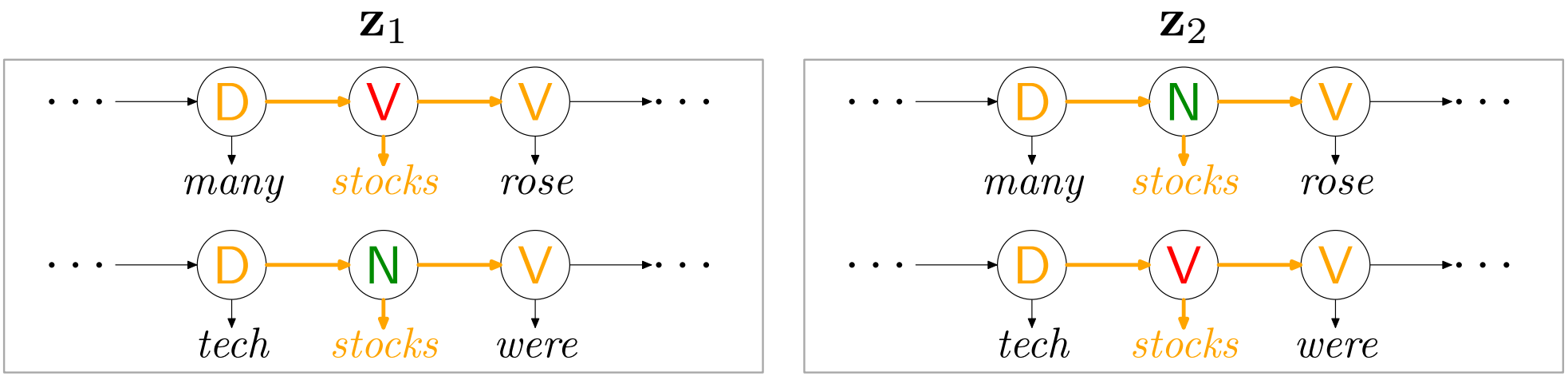
Sufficient statistics $\mathbf{n}$: # times parameters were used in $\mathbf{z}$

$n_{\text{T:V,N}}$: # times that state V transitioned to state N

Rewrite likelihood:

$p(\mathbf{z} \mid \boldsymbol{\theta}) = \prod_j \theta_j^{n_j}$ = simple function of $\mathbf{n}$ and $\boldsymbol{\theta}$

$p(\mathbf{z})$ = simple function of $\mathbf{n}$ (key: **exchangeability**)



$\mathbf{z}_1$ and $\mathbf{z}_2$ have same sufficient statistics $\Rightarrow$ $p(\mathbf{z}_1) = p(\mathbf{z}_2)$

Types

Type of a variable: its dependent parameter components

Types

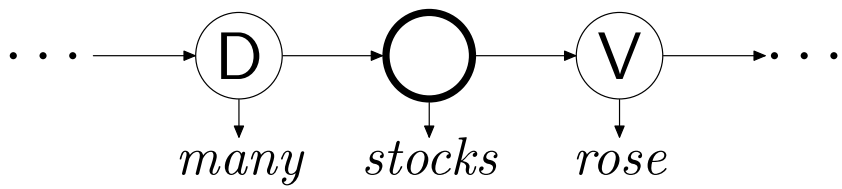
Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

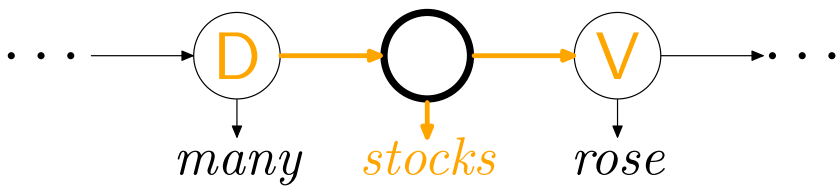


Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)

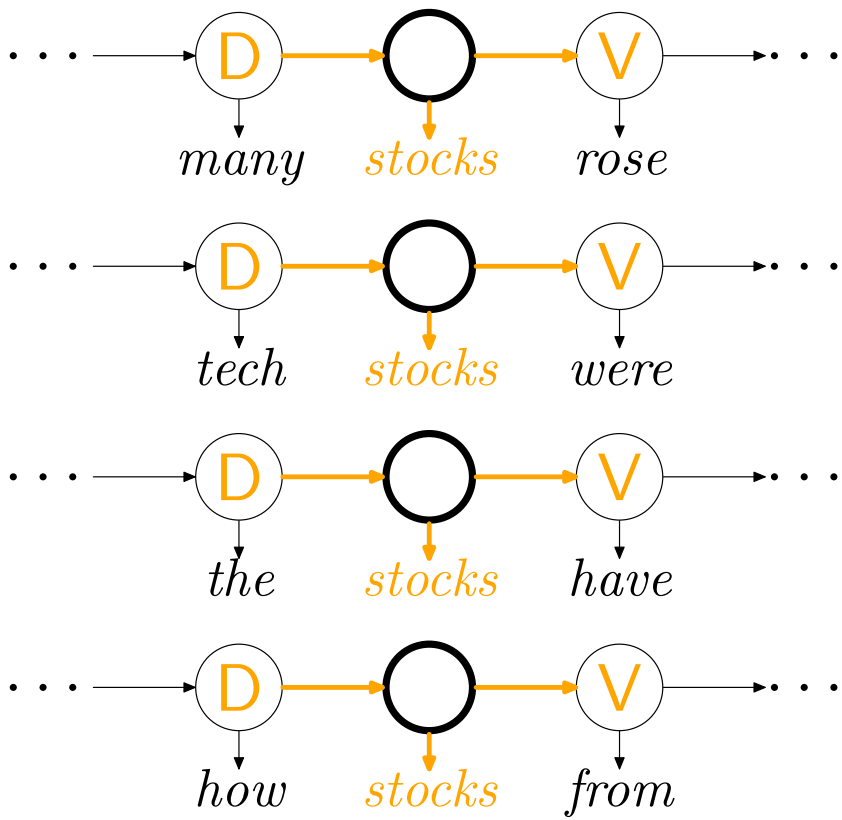


Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)

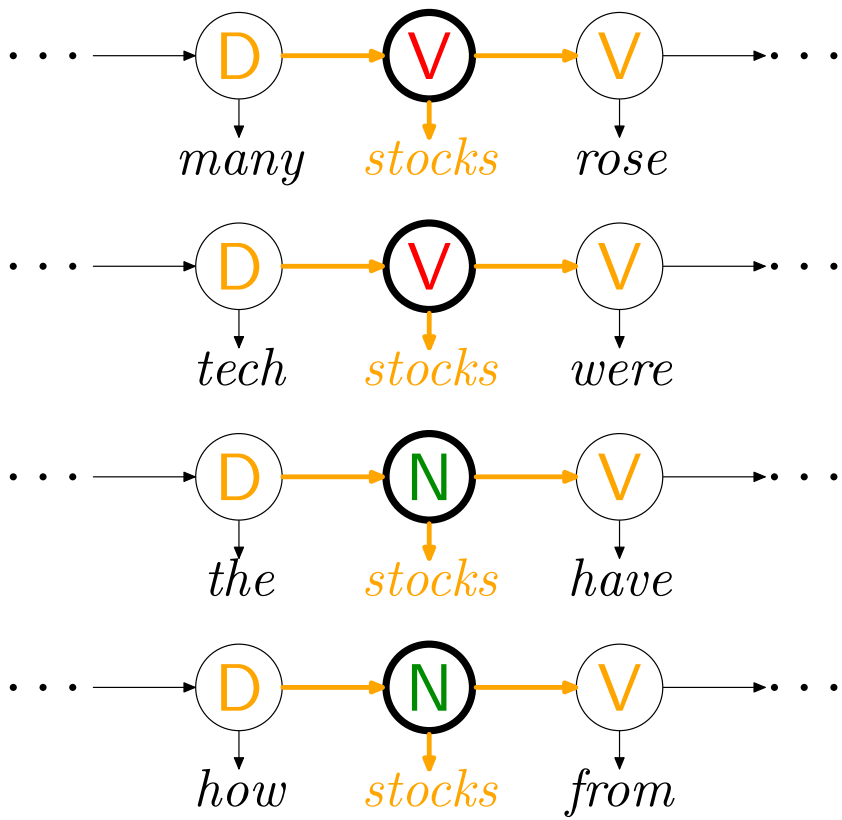


Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

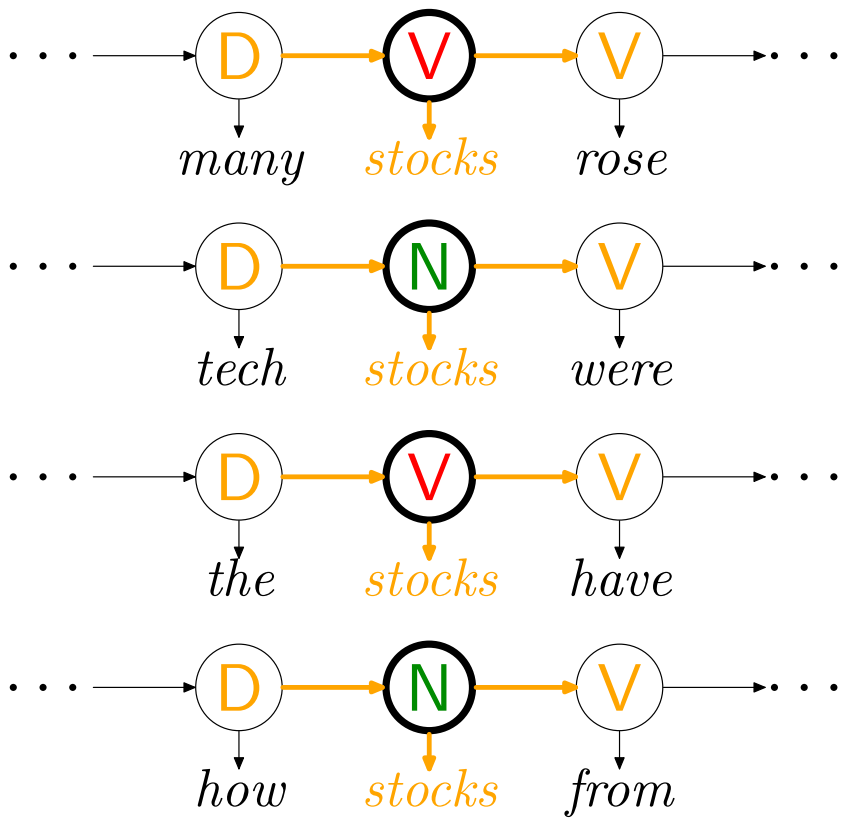
[VVNN]

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

[VVNN]

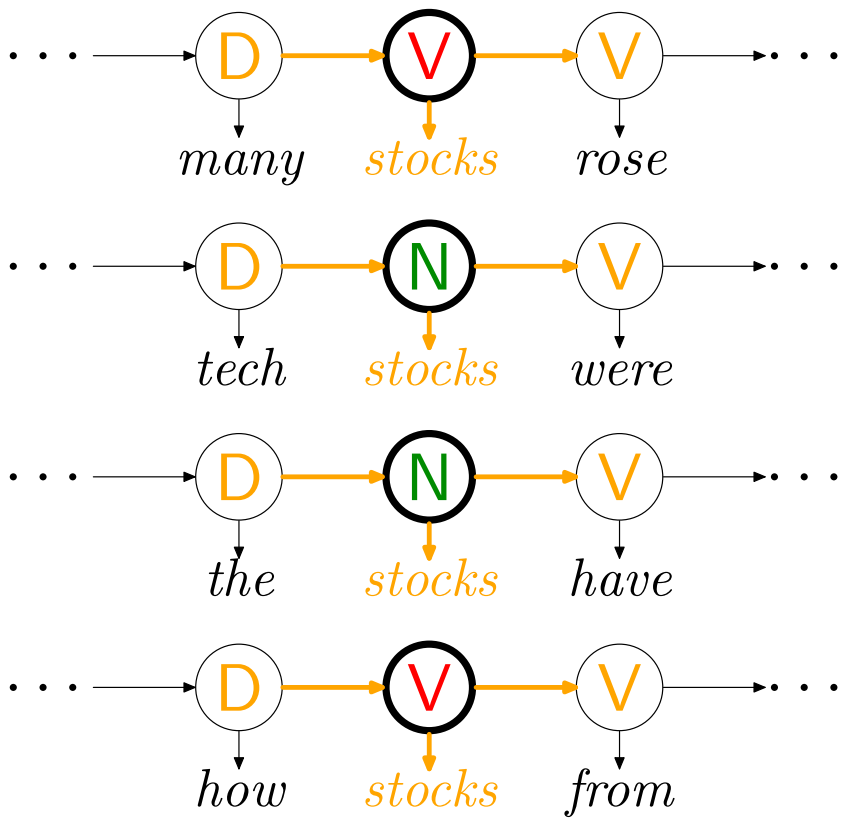
[VNVN]

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

[VVNN]

[VNVN]

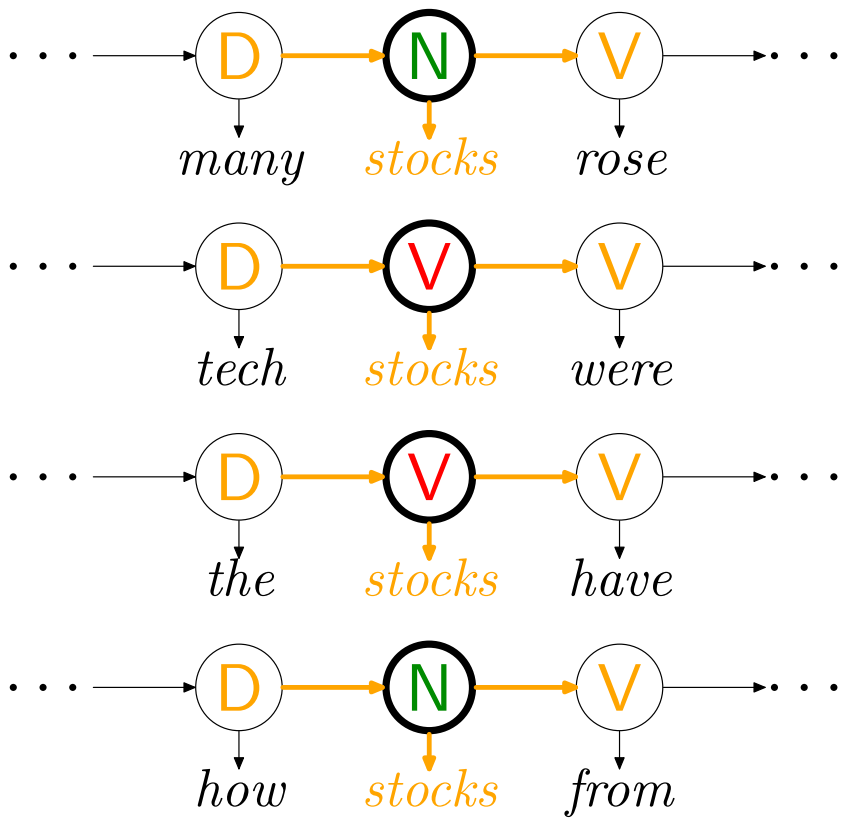
[VNNV]

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

[VVNN]

[VNVN]

[VNNV]

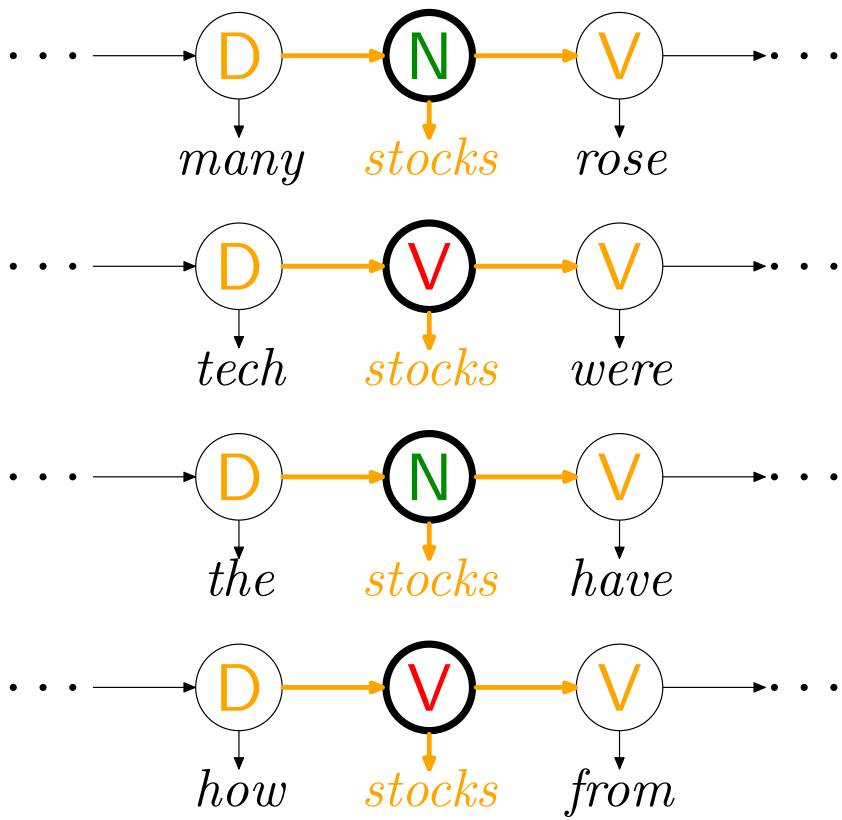
[NVVN]

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

[VVNN]

[VNVN]

[VNNV]

[NVVN]

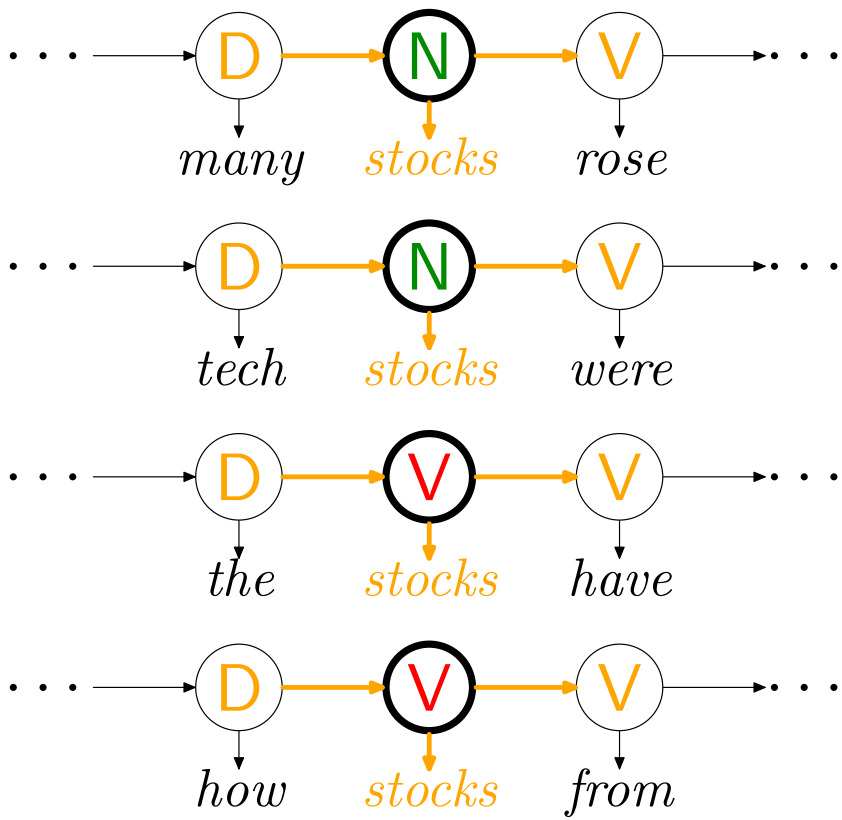
[NVNV]

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

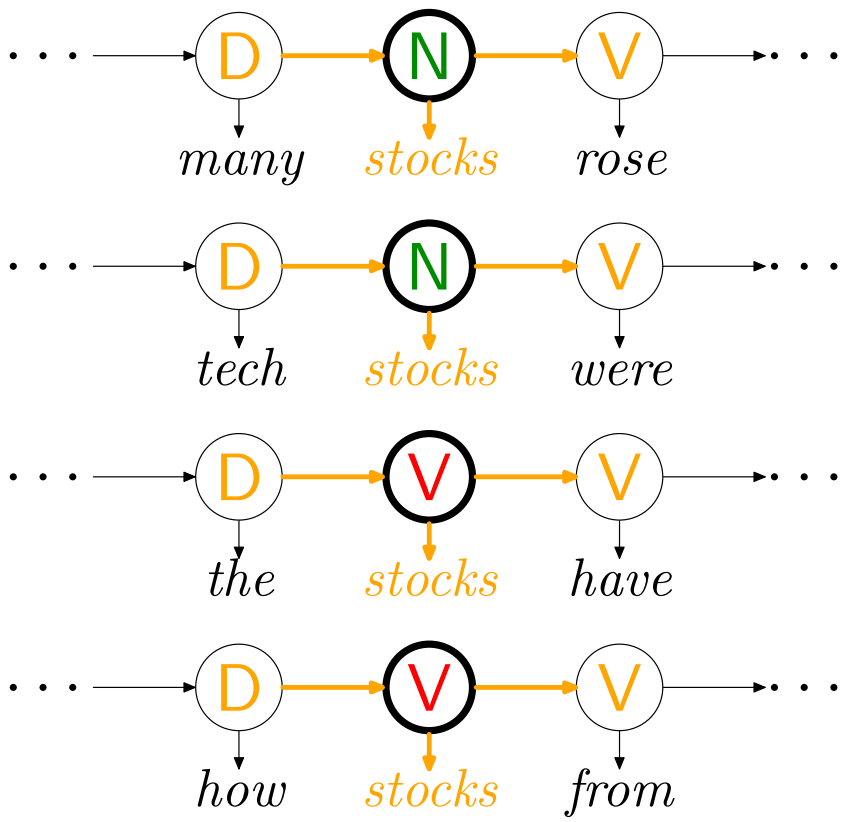
[VVNN]
[VNVN]
[VNNV]
[NVVN]
[NVNV]
[NNVV]

Types

Type of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

type(◯) = (D, *stocks*, V)



Assignments

[VVNN]
[VNVN]
[VNNV]
[NVVN]
[NVNV]
[NNVV]

p(assignment) only depends on number of Vs and Ns

Sampling same-type variables

Goal: sample an assignment of a set of same-type variables

Sampling same-type variables

Goal: sample an assignment of a set of same-type variables

$m = 0$	$m = 1$	$m = 2$	$m = 3$	$m = 4$
p_0 [VVVV]	p_1 [VVVN]	p_2 [VVNN]	p_3 [VNNN]	p_4 [NNNN]
	p_1 [VVNV]	p_2 [VNVN]	p_3 [NVNN]	
	p_1 [VNVV]	p_2 [VNNV]	p_3 [NNVN]	
	p_1 [NVVV]	p_2 [NVVN]	p_3 [NNNV]	
		p_2 [NVNV]		
		p_2 [NNVV]		

Sampling same-type variables

Goal: sample an assignment of a set of same-type variables

Algorithm:

1. Choose $m \in \{0, \ldots, B\}$ with prob. $\propto \binom{B}{m} p_m$

$m = 0$	$m = 1$	$m = 2$	$m = 3$	$m = 4$
p_0 [VVVV]	p_1 [VVVN]	p_2 [VVNN]	p_3 [VNNN]	p_4 [NNNN]
	p_1 [VVNV]	p_2 [VNVN]	p_3 [NVNN]	
	p_1 [VNVV]	p_2 [VNNV]	p_3 [NNVN]	
	p_1 [NVVV]	p_2 [NVVN]	p_3 [NNNV]	
		p_2 [NVNV]		
		p_2 [NNVV]		

Sampling same-type variables

Goal: sample an assignment of a set of same-type variables

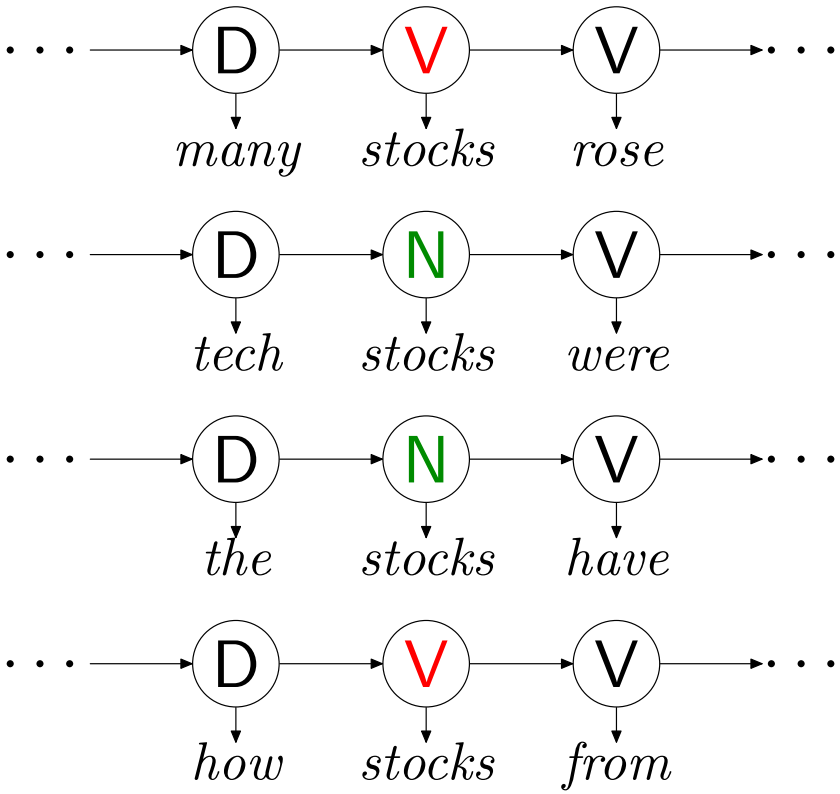
Algorithm:

1. Choose $m \in \{0, \ldots, B\}$ with prob. $\propto \binom{B}{m} p_m$
2. Choose assignment uniformly from column m

$m = 0$	$m = 1$	$m = 2$	$m = 3$	$m = 4$
p_0 [VVVV]	p_1 [VVVN]	p_2 [VVNN]	p_3 [VNNN]	p_4 [NNNN]
	p_1 [VVNV]	p_2 [VNVN]	p_3 [NVNN]	
	p_1 [VNVV]	p_2 [VNNV]	p_3 [NNVN]	
	p_1 [NVVV]	p_2 [NVVN]	p_3 [NNNV]	
		p_2 [NVNV]		
		p_2 [NNVV]		

Full algorithm

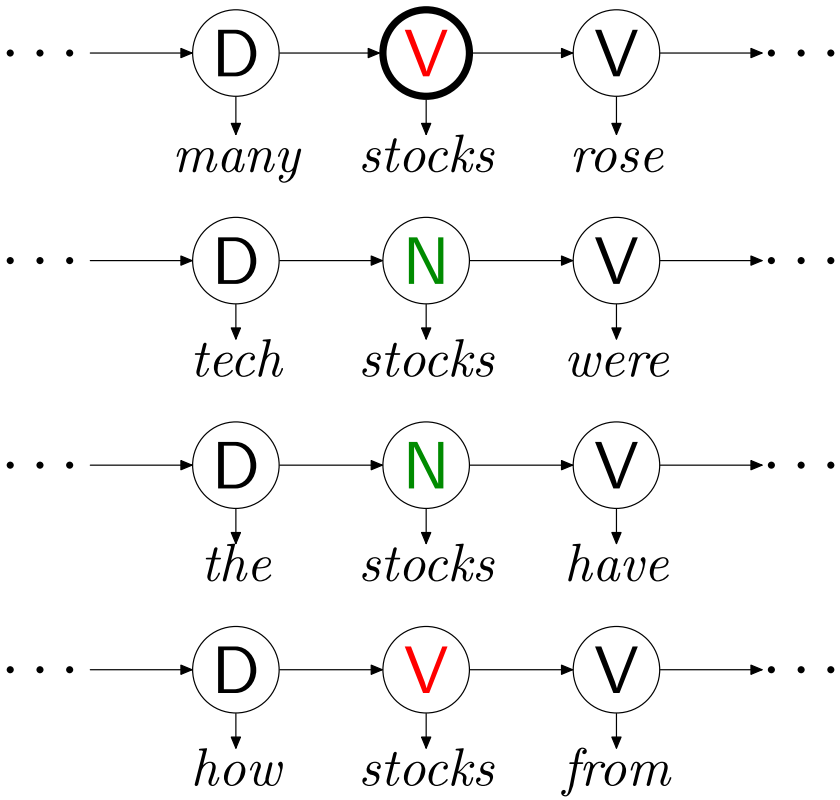
Iterate:



Full algorithm

Iterate:

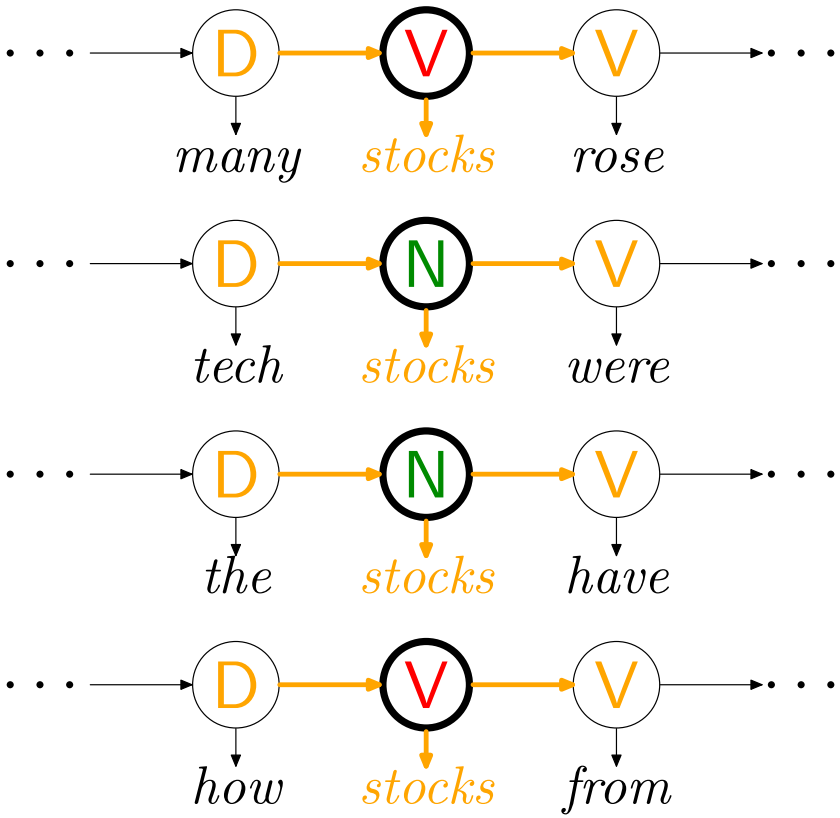
1. Choose a position
   e.g., 2



Full algorithm

Iterate:

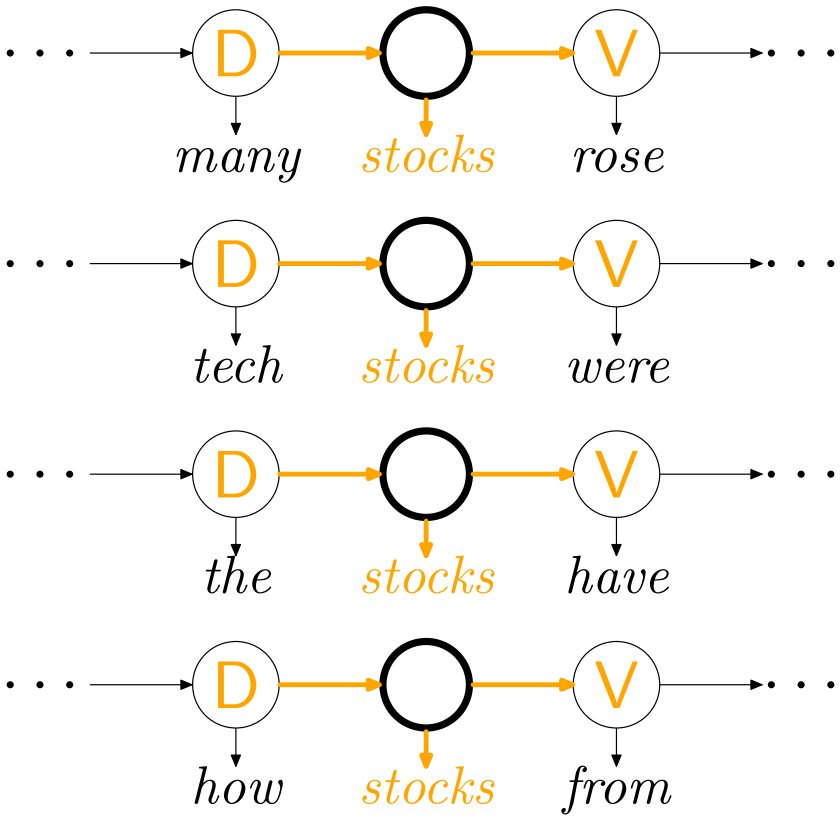
1. Choose a position
   e.g., 2
2. Add variables with same type
   e.g., (D, *stocks*, V)



Full algorithm

Iterate:

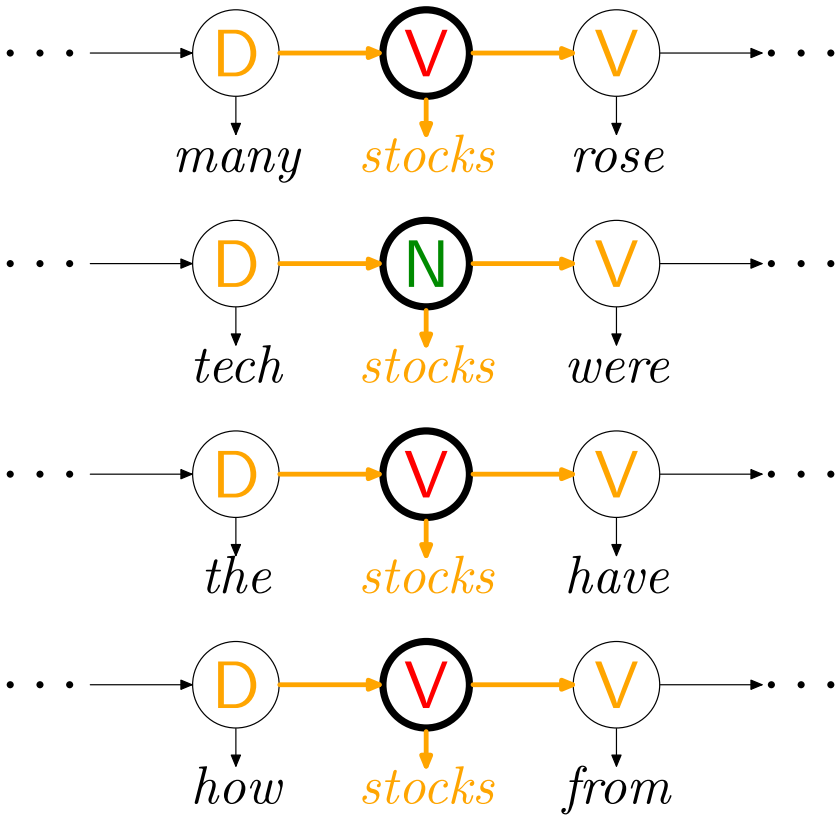
1. Choose a position
   e.g., 2
2. Add variables with same type
   e.g., (D, *stocks*, V)
3. Sample m
   e.g., 1 $\Rightarrow$ {V, V, V, N}



Full algorithm

Iterate:

1. Choose a position
   e.g., 2
2. Add variables with same type
   e.g., (D, *stocks*, V)
3. Sample m
   e.g., 1 $\Rightarrow$ {V, V, V, N}
4. Sample assignment
   e.g., [VNVV]



Experimental setup: sampling algorithms

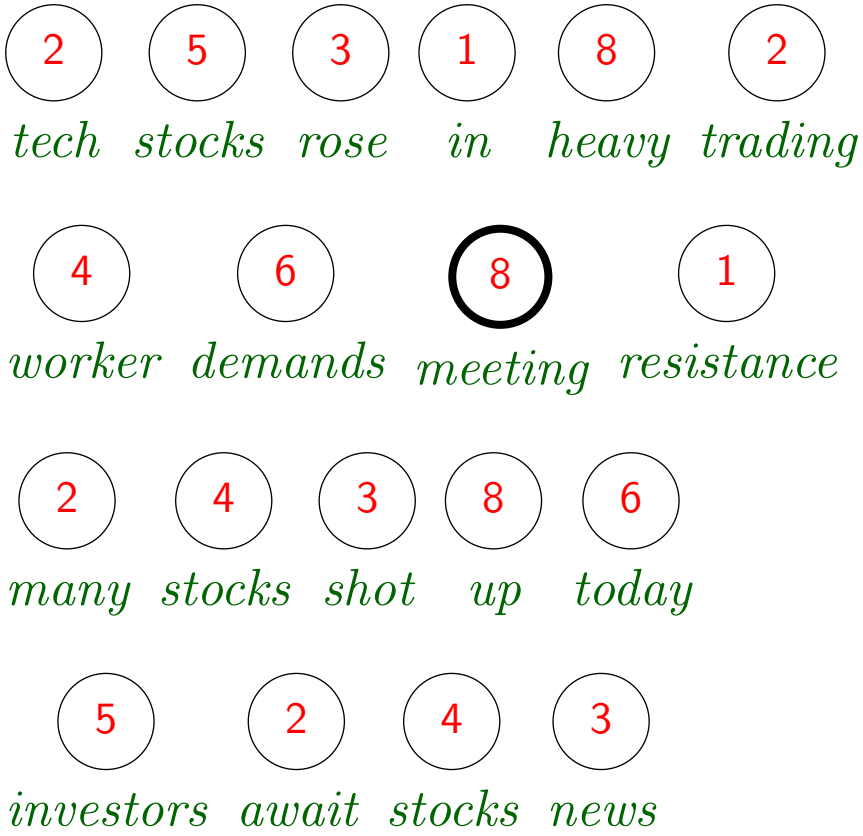
2	5	3	1	8	2
tech	*stocks*	*rose*	*in*	*heavy*	*trading*

4	6	8	1
worker	*demands*	*meeting*	*resistance*

2	4	3	8	6
many	*stocks*	*shot*	*up*	*today*

5	2	4	3
investors	*await*	*stocks*	*news*

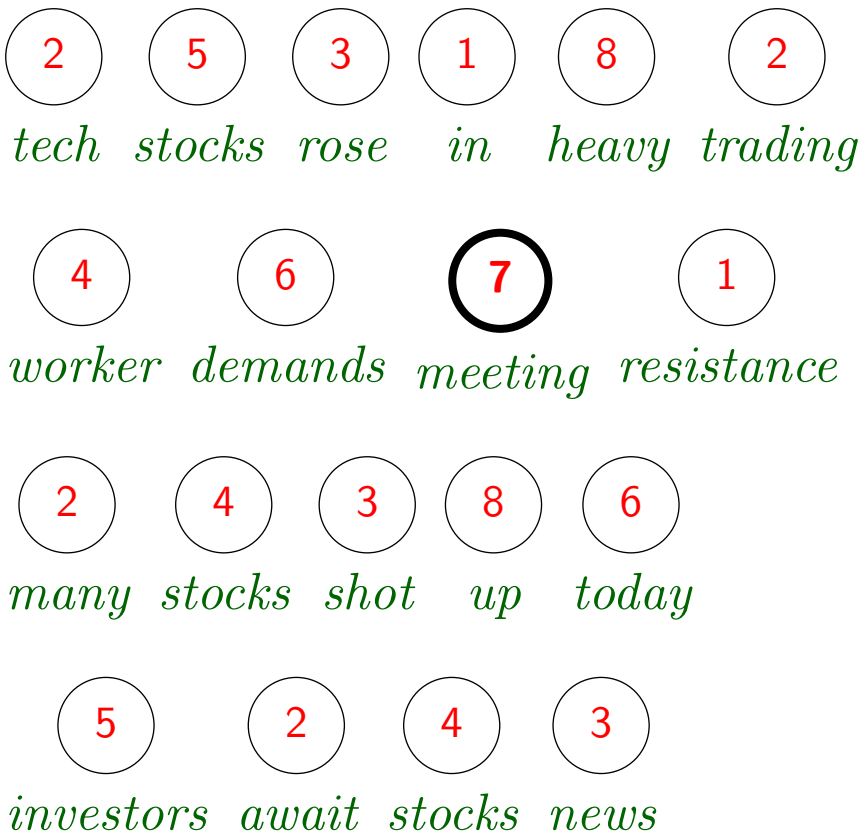
Experimental setup: sampling algorithms



Token-based sampler (TOKEN)

1. Choose **token**

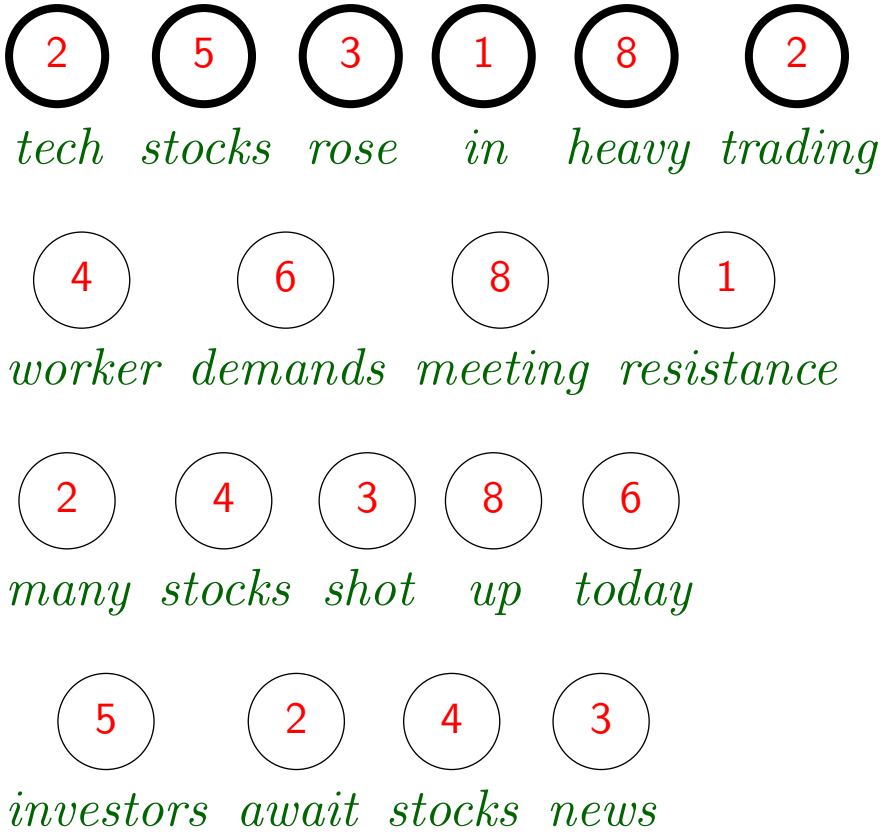
Experimental setup: sampling algorithms



Token-based sampler (TOKEN)

1. Choose **token**
2. Update conditioned on rest

Experimental setup: sampling algorithms



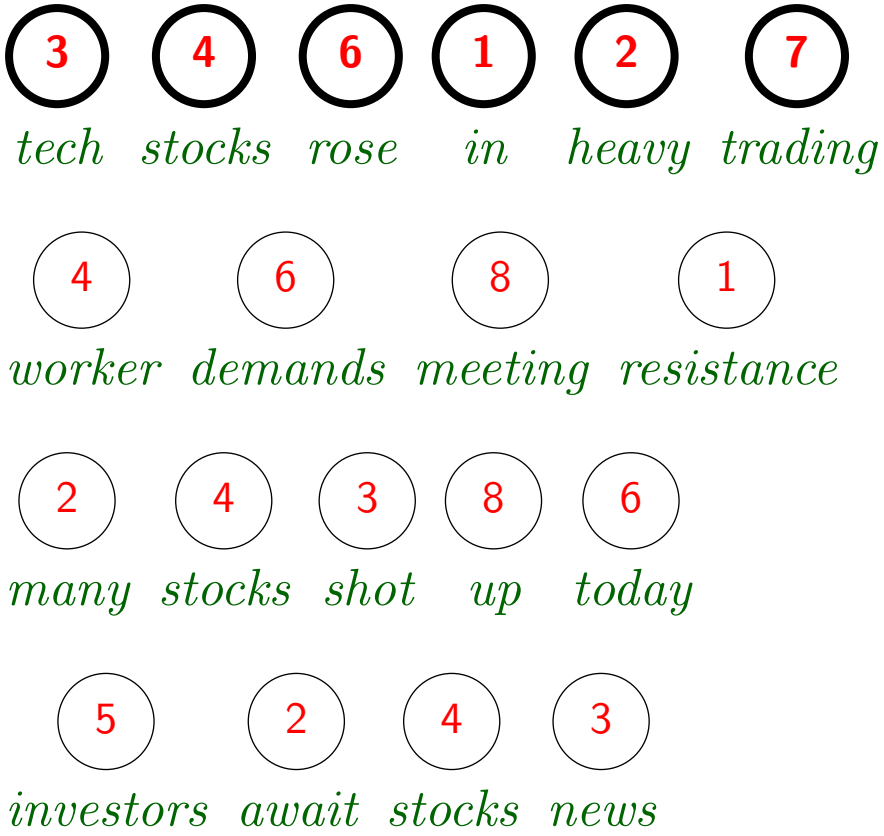
Token-based sampler (TOKEN)

1. Choose **token**
2. Update conditioned on rest

Sentence-based sampler (SENTENCE)

1. Choose **sentence**

Experimental setup: sampling algorithms



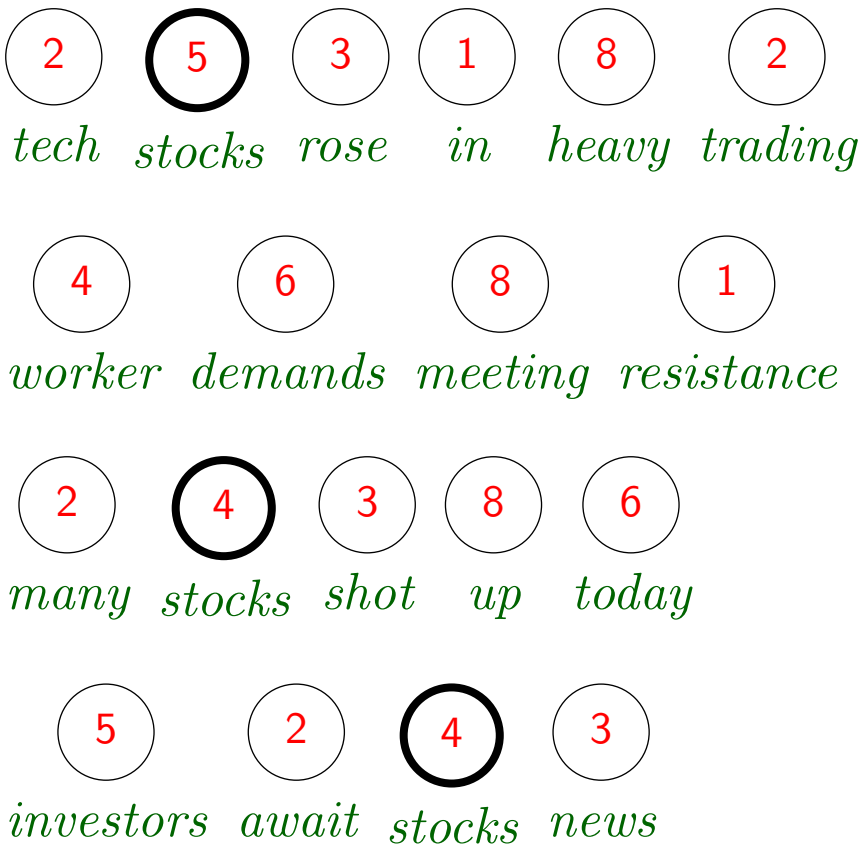
Token-based sampler (TOKEN)

1. Choose **token**
2. Update conditioned on rest

Sentence-based sampler (SENTENCE)

1. Choose **sentence**
2. Update conditioned on rest

Experimental setup: sampling algorithms



Token-based sampler (TOKEN)

1. Choose **token**
2. Update conditioned on rest

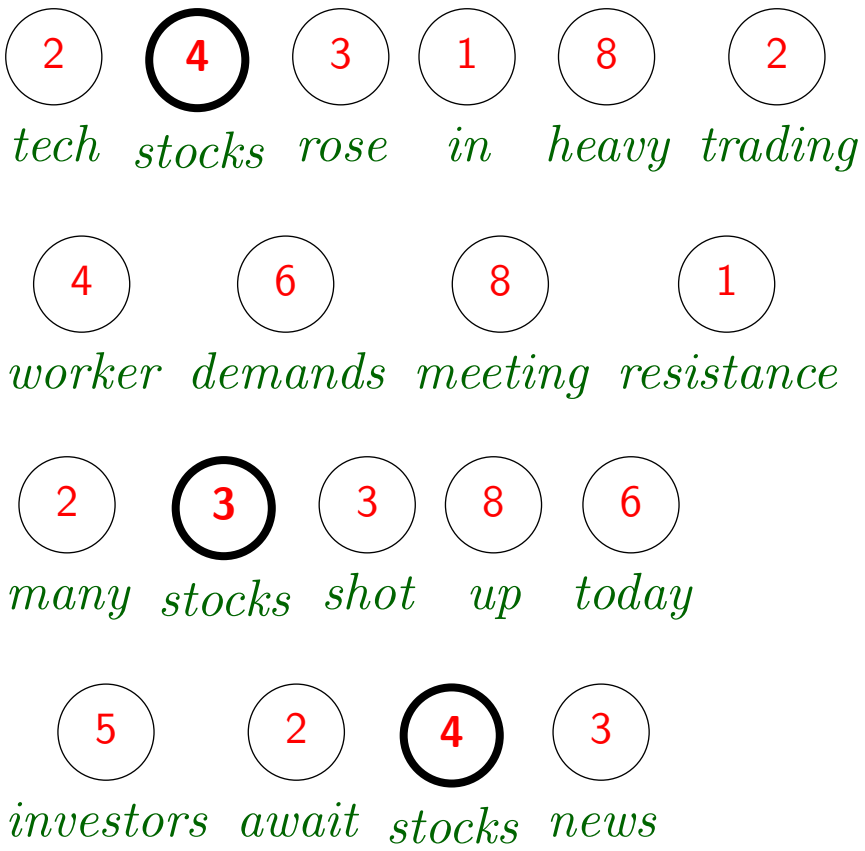
Sentence-based sampler (SENTENCE)

1. Choose **sentence**
2. Update conditioned on rest

Type-based sampler (TYPE)

1. Choose **type**

Experimental setup: sampling algorithms



Token-based sampler (TOKEN)

1. Choose **token**
2. Update conditioned on rest

Sentence-based sampler (SENTENCE)

1. Choose **sentence**
2. Update conditioned on rest

Type-based sampler (TYPE)

1. Choose **type**
2. Update conditioned on rest

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

worker demands meeting resistance

Part-of-speech induction

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

Unigram segmentation model (**USM**) [Goldwater et al., 2006]

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k a t t h e b o o k

Word segmentation

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k | a t | t h e | b o o k

Word segmentation

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k | a t | t h e | b o o k

Word segmentation

Dataset: CHILDES
(9.7K sentences)

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k | a t | t h e | b o o k

Word segmentation

Dataset: CHILDES
(9.7K sentences)

Probabilistic tree-substitution grammar (**PTSG**) [Cohn et al., 2009]

Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

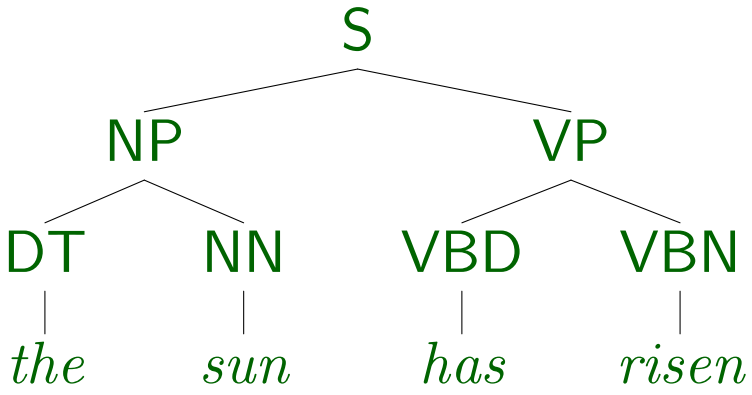
Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k | a t | t h e | b o o k

Word segmentation

Dataset: CHILDES
(9.7K sentences)

Probabilistic tree-substitution grammar (**PTSG**) [Cohn et al., 2009]



Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

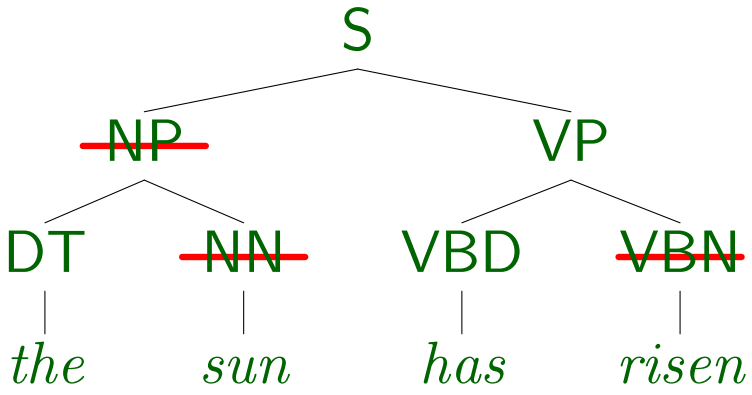
Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k | a t | t h e | b o o k

Word segmentation

Dataset: CHILDES
(9.7K sentences)

Probabilistic tree-substitution grammar (**PTSG**) [Cohn et al., 2009]



Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)

NN	NNS	VBG	NN
worker	*demands*	*meeting*	*resistance*

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

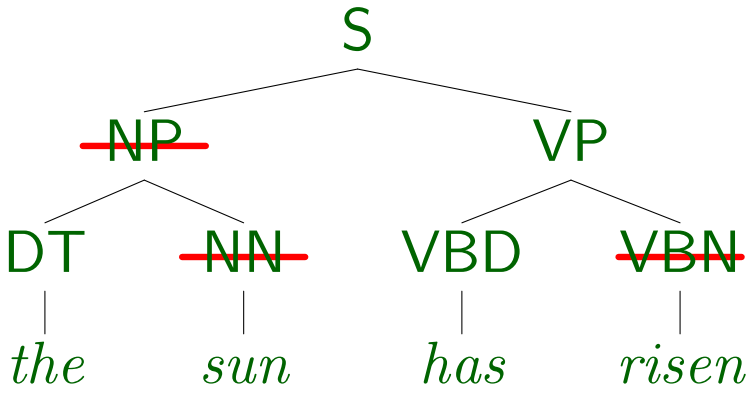
Unigram segmentation model (**USM**) [Goldwater et al., 2006]

l o o k | a t | t h e | b o o k

Word segmentation

Dataset: CHILDES
(9.7K sentences)

Probabilistic tree-substitution grammar (**PTSG**) [Cohn et al., 2009]



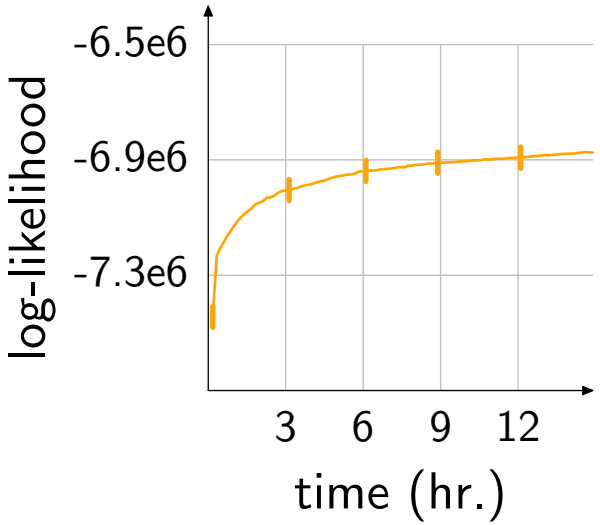
Dataset: WSJ
(49K sentences, 45 tags)

Token versus Type

HMM **USM** **PTSG**

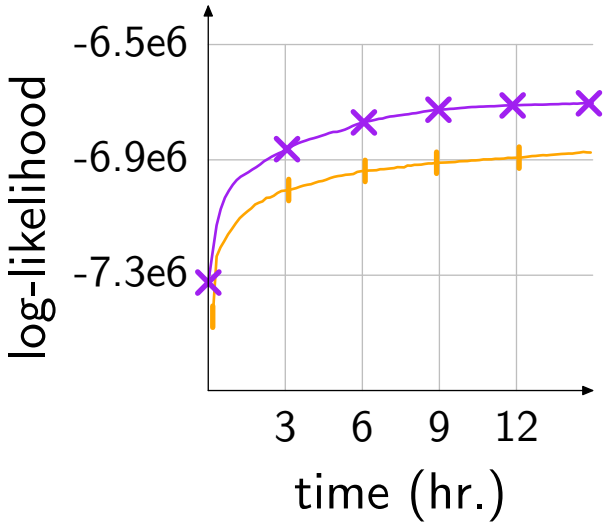
Token versus Type

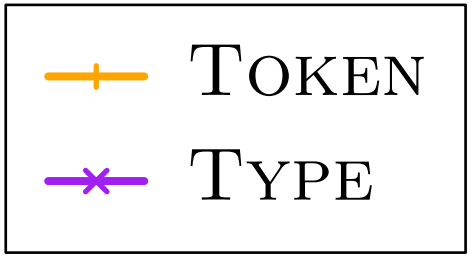
HMM **USM** **PTSG**



Token versus Type

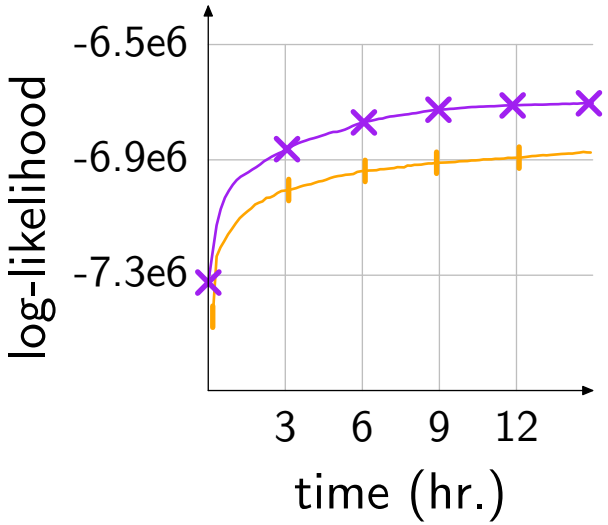
HMM **USM** **PTSG**



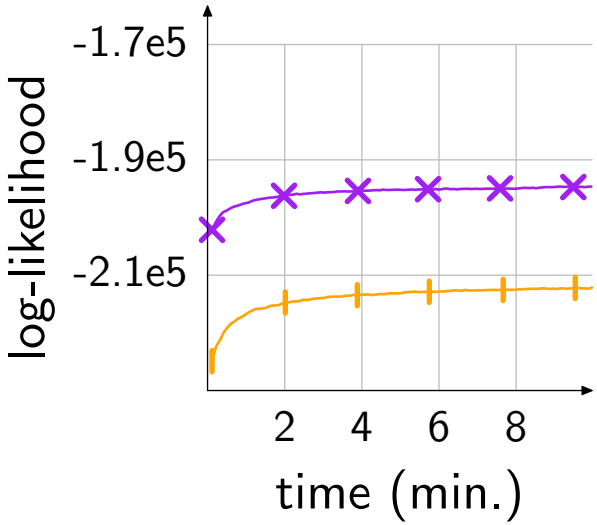


Token versus Type

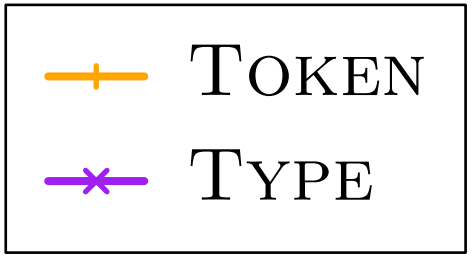
HMM



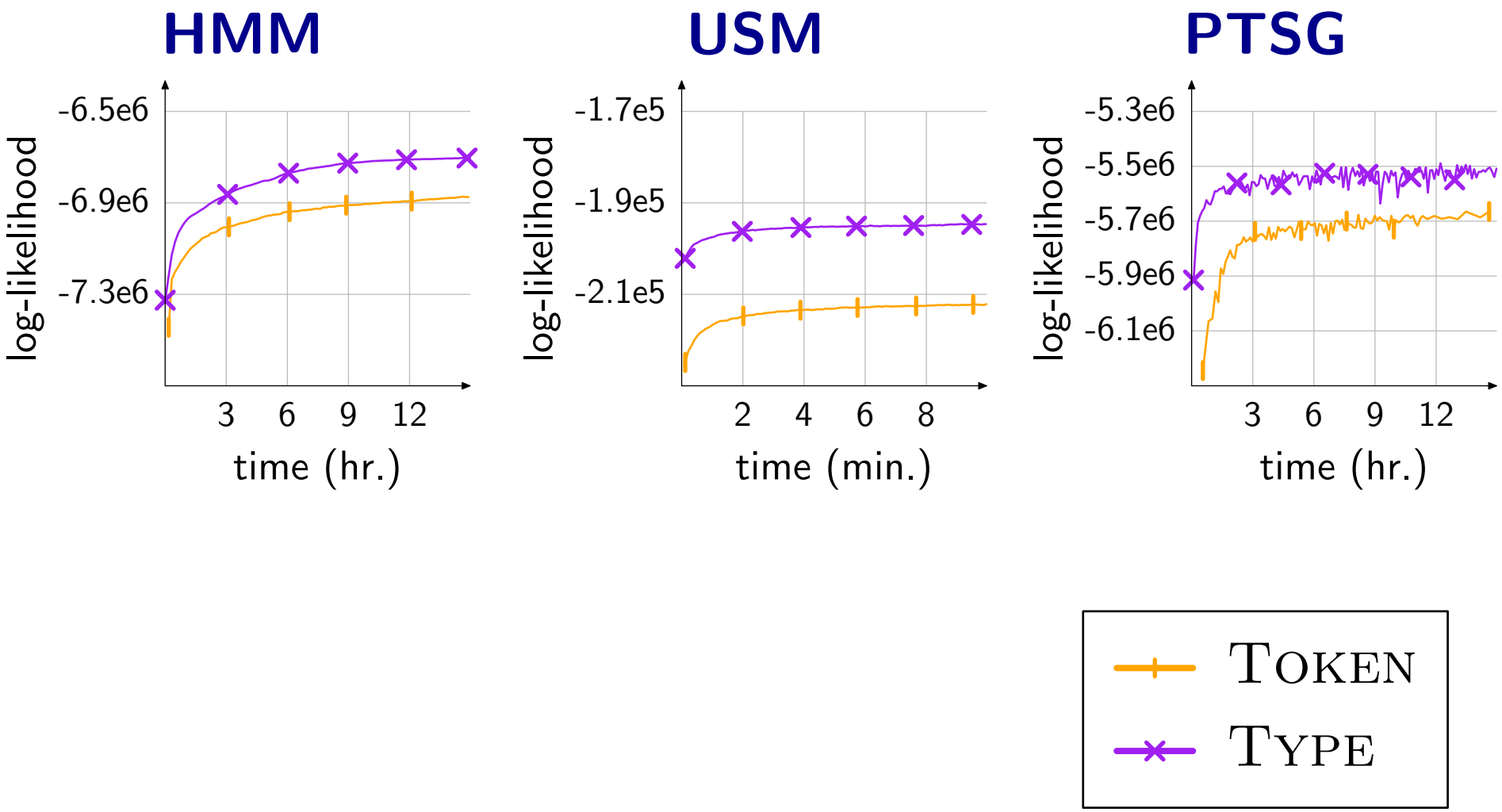
USM



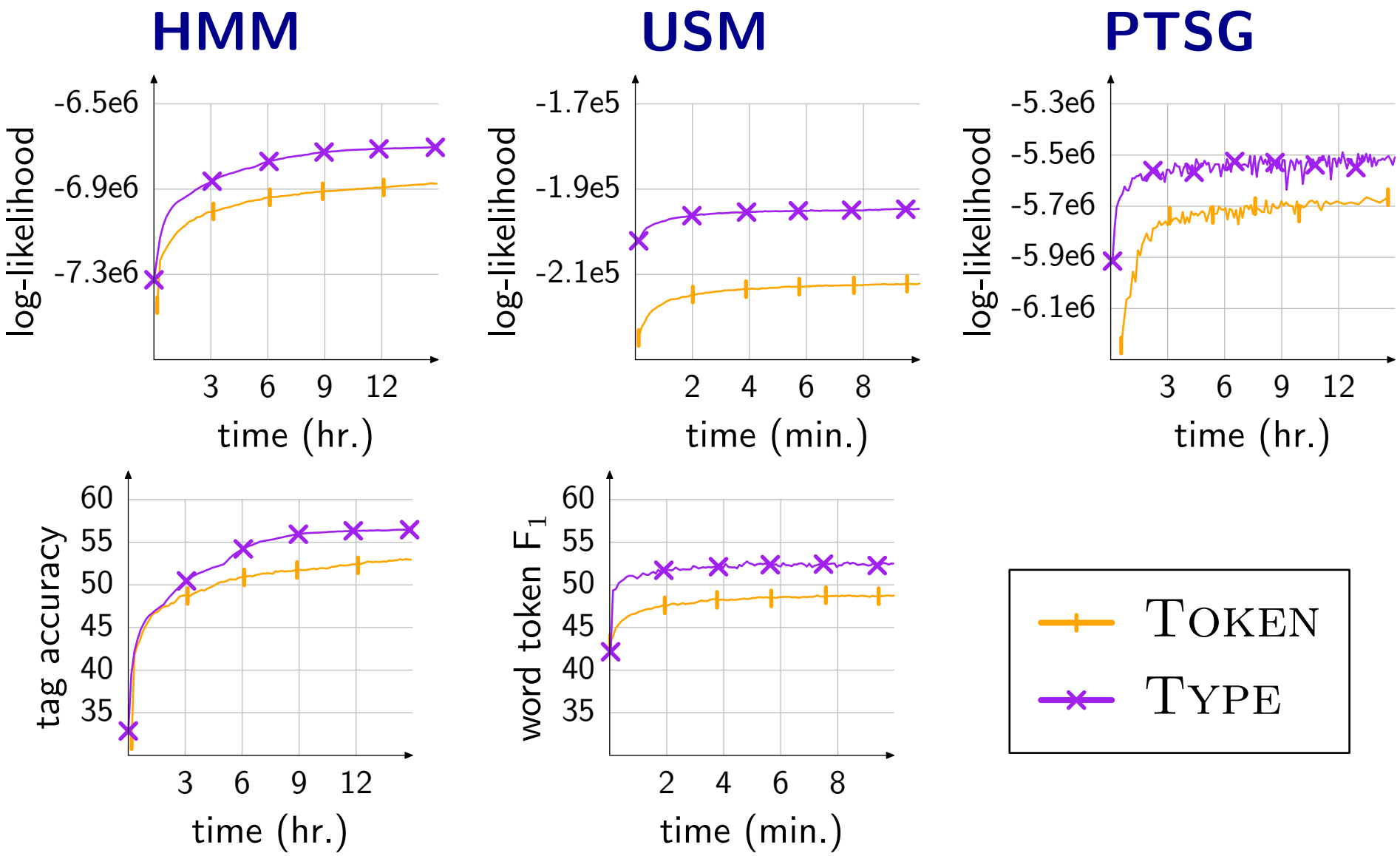
PTSG



Token versus Type



Token versus Type



Sensitivity to initialization

Sensitivity to initialization

(use few params.)

$\eta = 0$ $\longleftrightarrow$

Sensitivity to initialization

(use few params.)

$\eta = 0$ ⟷

V V V V

worker *demands* *meeting* *resistance*

Sensitivity to initialization

(use few params.)

$\eta = 0$ ⟷

V V V V

worker demands meeting resistance

| l | o | o | k | a | t | t | h | e | b | o | o | k |

Sensitivity to initialization

(use few params.) $\eta = 0$ $\longleftrightarrow$ (use many params.) $\eta = 1$

V V V V

worker demands meeting resistance

| l | o | o | k | a | t | t | h | e | b | o | o | k |

Sensitivity to initialization

(use few params.) $\eta = 0$ $\longleftrightarrow$ (use many params.) $\eta = 1$

V	V	V	V	P	N	D	V
worker	*demands*	*meeting*	*resistance*	*worker*	*demands*	*meeting*	*resistance*

$|l|o|o|k|a|t|t|h|e|b|o|o|k|$

Sensitivity to initialization

(use few params.) $\eta = 0$ $\longleftrightarrow$ (use many params.) $\eta = 1$

V	V	V	V	P	N	D	V
worker	*demands*	*meeting*	*resistance*	*worker*	*demands*	*meeting*	*resistance*

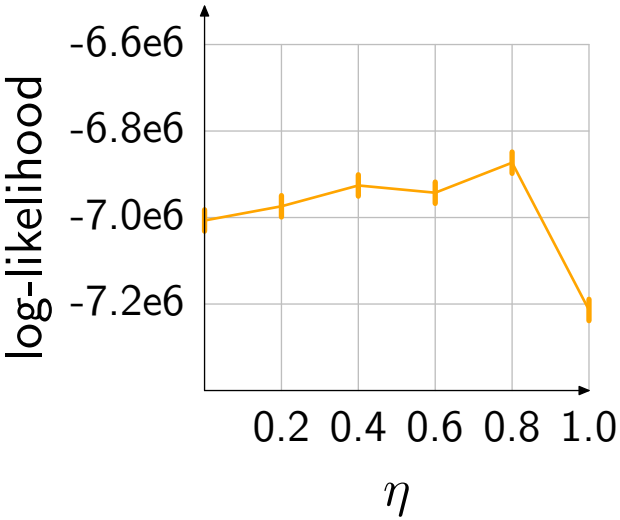
$|l|o|o|k|a|t|t|h|e|b|o|o|k|$ $\qquad$ $|l\ o\ o\ k\ a\ t\ t\ h\ e\ b\ o\ o\ k|$

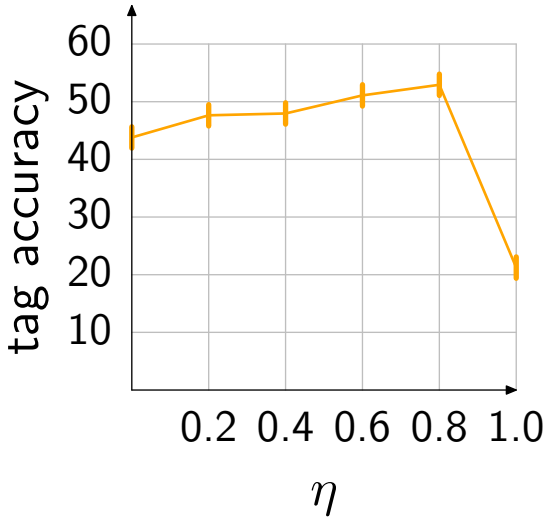
Sensitivity to initialization

HMM **USM** **PTSG**

Sensitivity to initialization

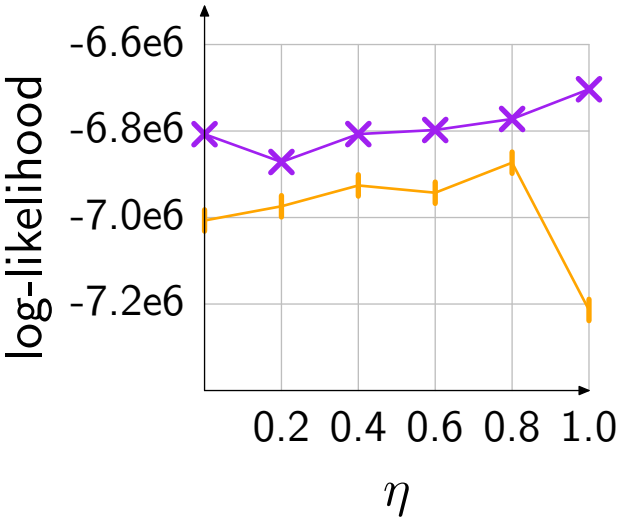
HMM **USM** **PTSG**

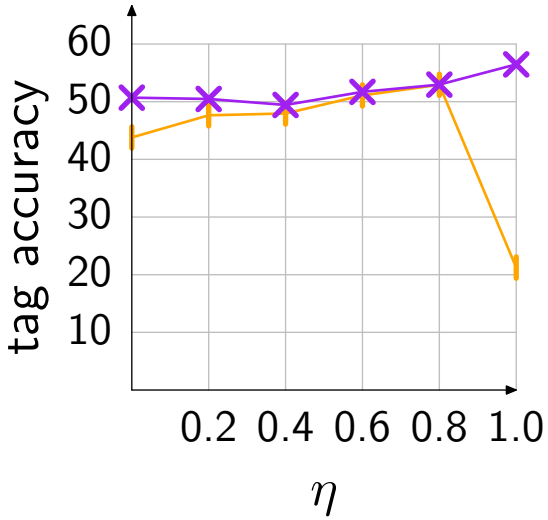


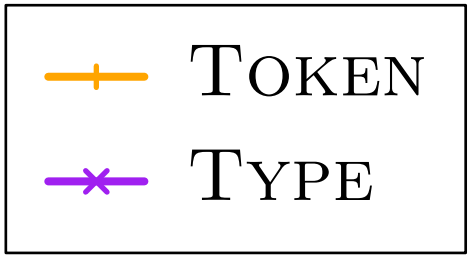


Sensitivity to initialization

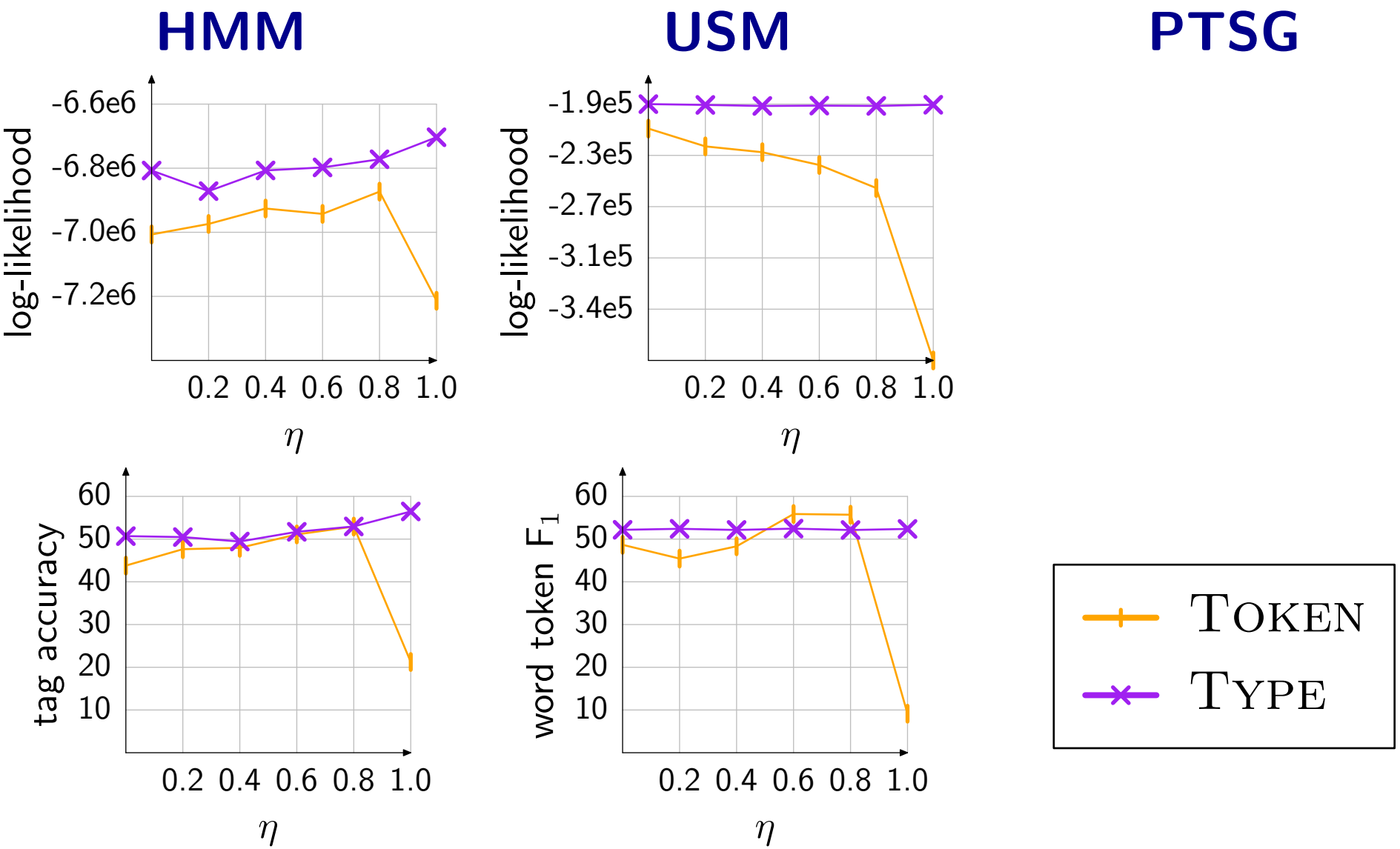
HMM **USM** **PTSG**



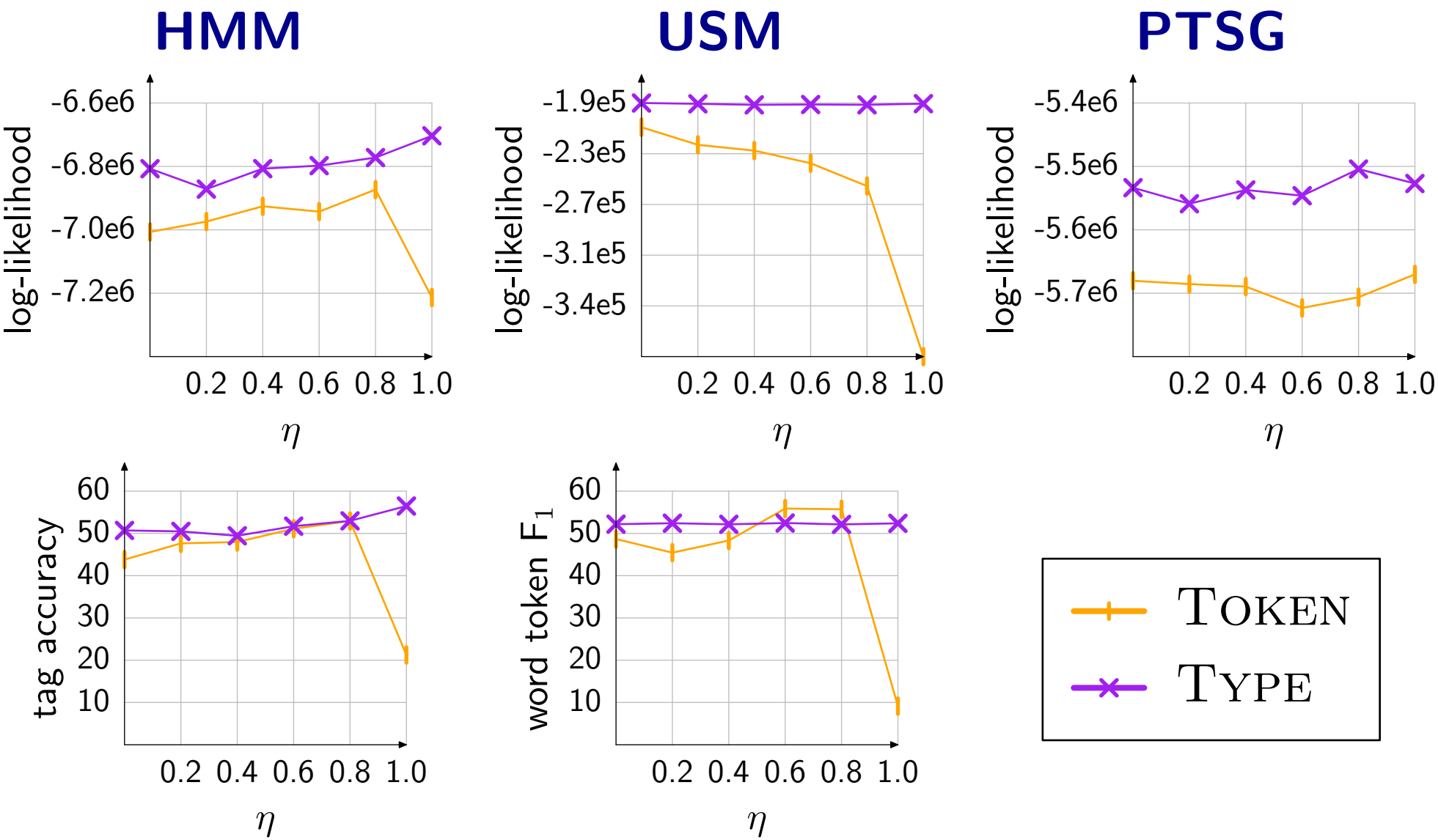




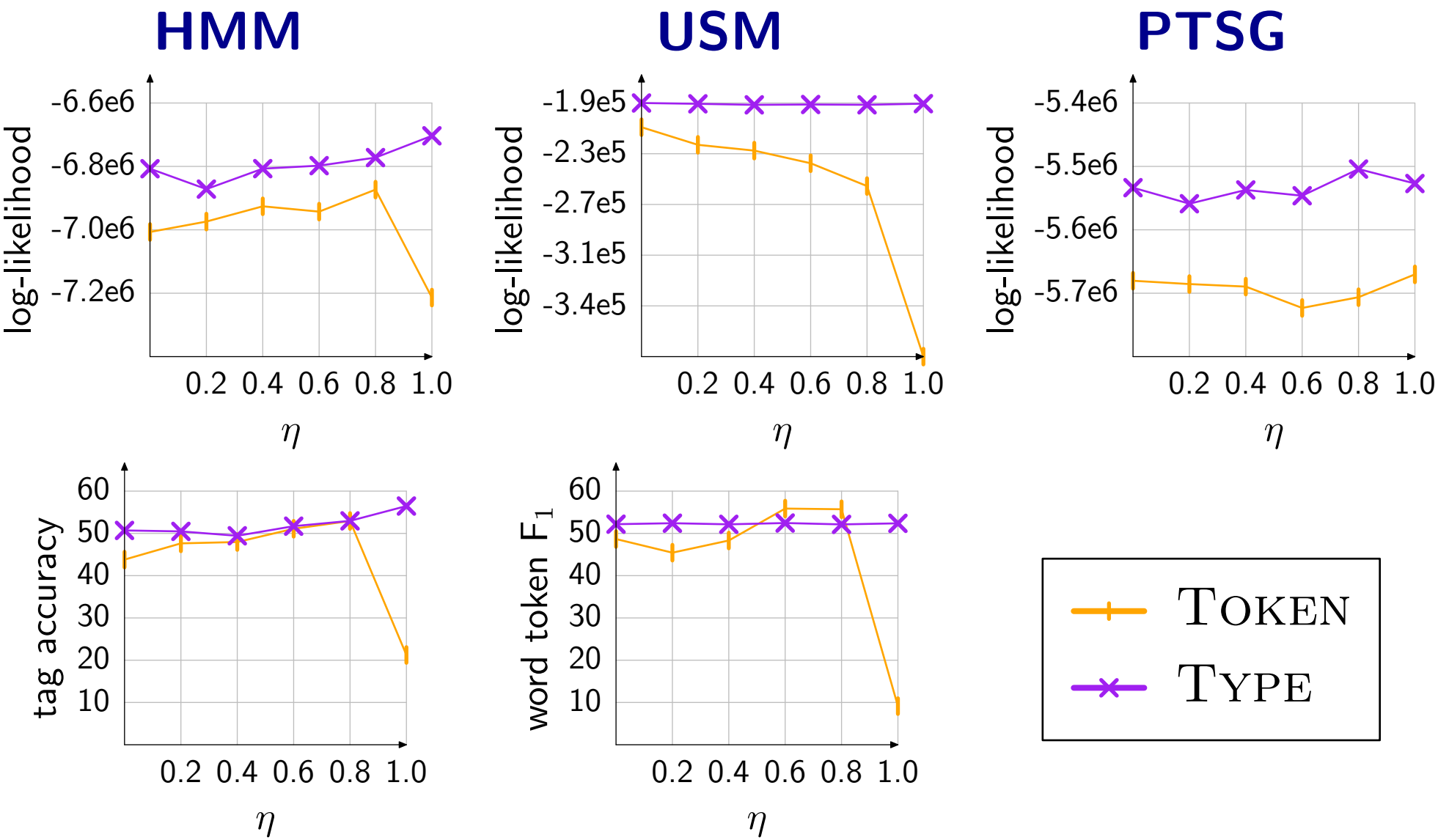
Sensitivity to initialization



Sensitivity to initialization



Sensitivity to initialization



Type less sensitive than Token

Alternatives

Can we get the gains of TYPE via simpler means?

Alternatives

Can we get the gains of TYPE via simpler means?

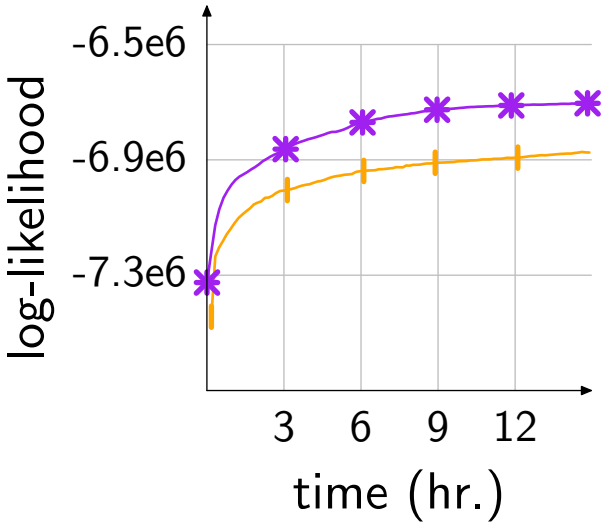
- Annealing ($\text{TOKEN}_{\text{anneal}}$)
  - Use $p(\mathbf{z} \mid \mathbf{x})^{1/T}$, temperature T from 5 to 1

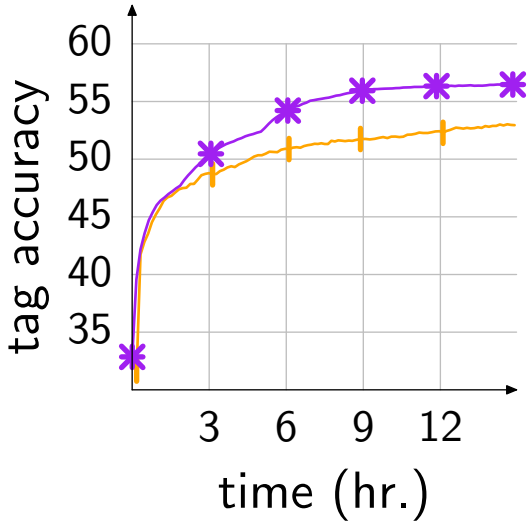
Annealing

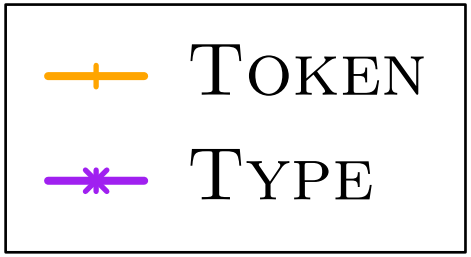
HMM **USM** **PTSG**

Annealing

HMM **USM** **PTSG**

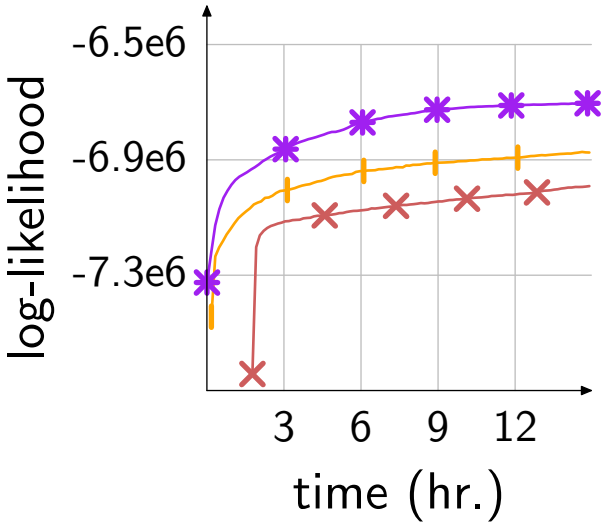


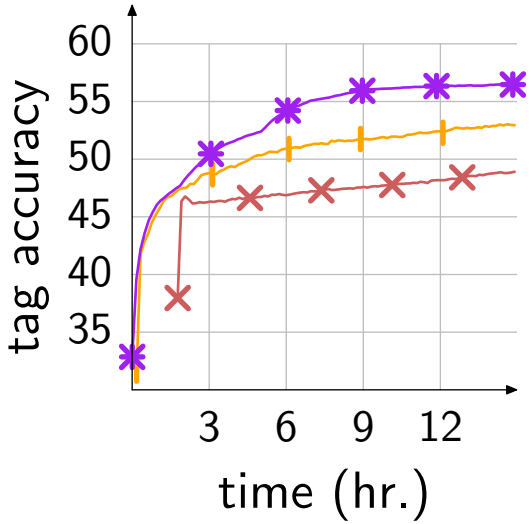




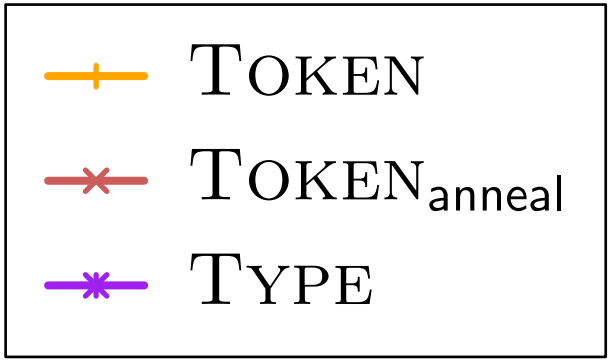
Annealing

HMM **USM** **PTSG**



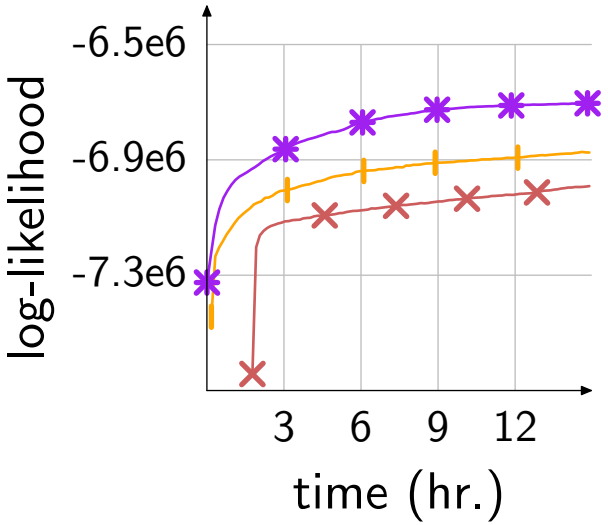


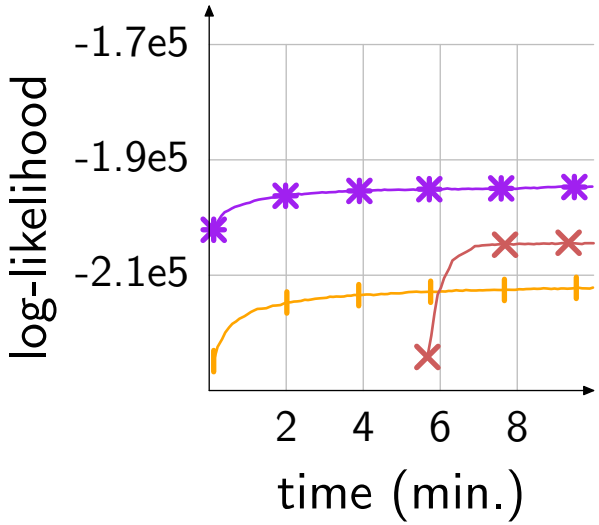
[anneal. hurts]

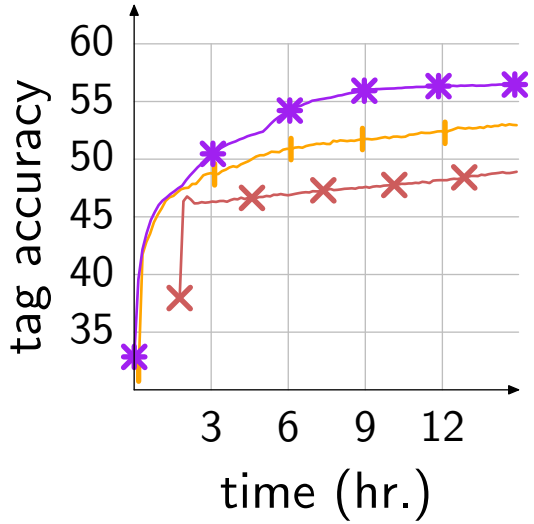


Annealing

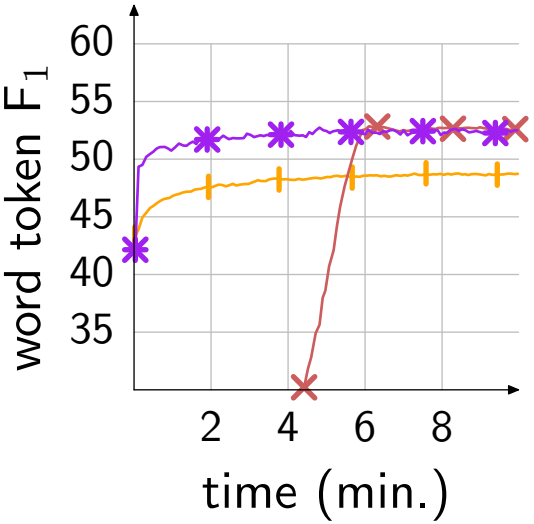
HMM **USM** **PTSG**



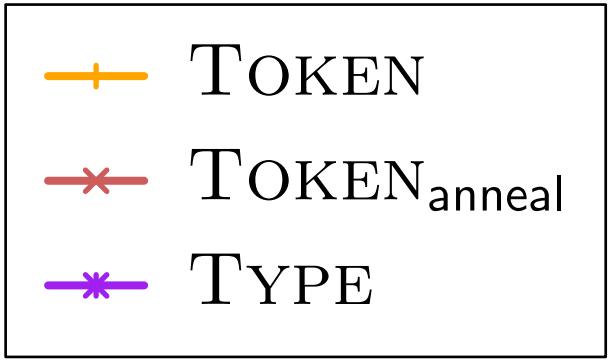




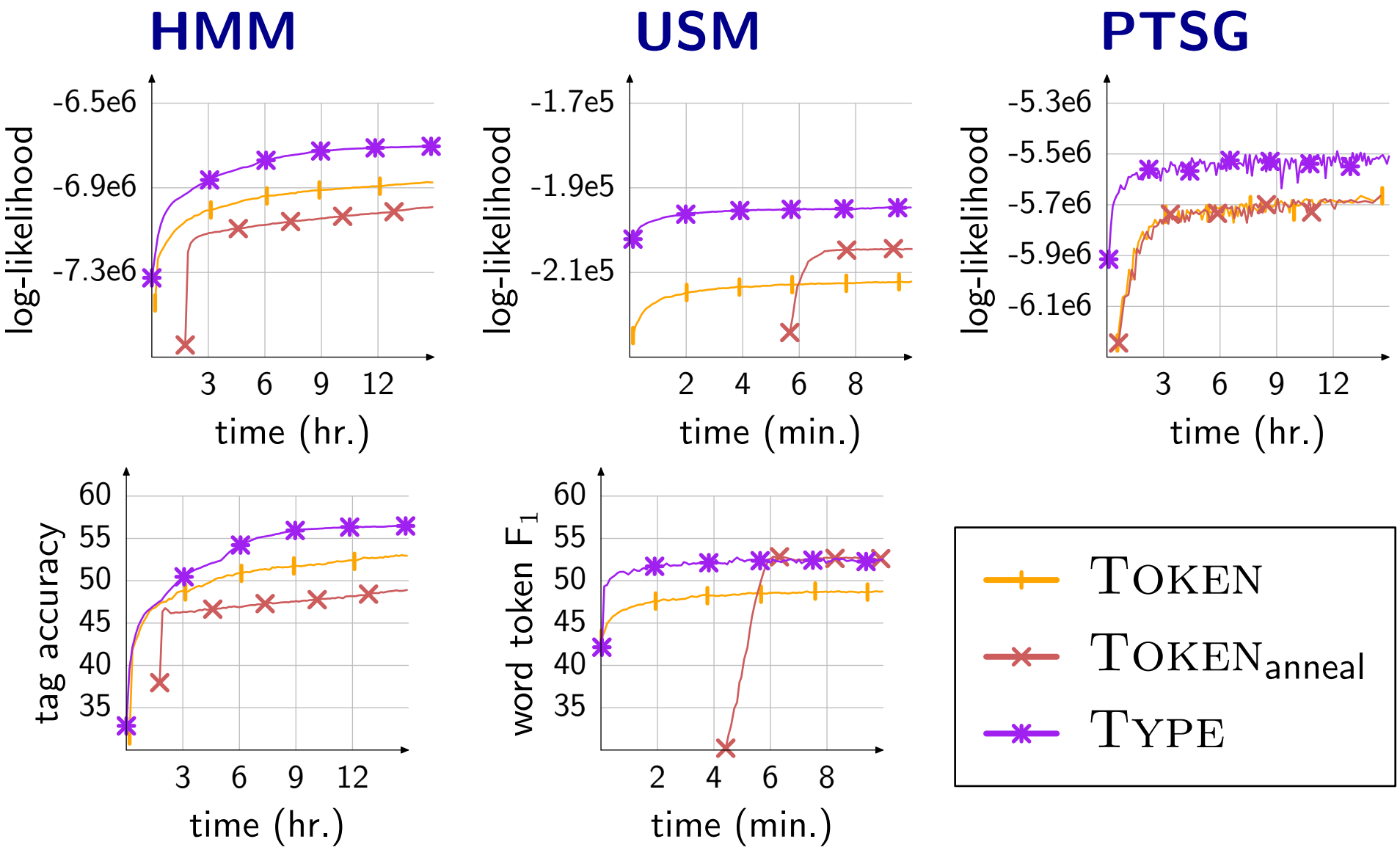
[anneal. hurts]



[anneal. helps]



Annealing



[anneal. hurts] [anneal. helps] [no effect]

Alternatives

Can we get the gains of TYPE via simpler means?

- Annealing ($\text{TOKEN}_{\text{anneal}}$)
  - Use $p(\mathbf{z} \mid \mathbf{x})^{1/T}$, temperature T from 5 to 1
  - More (random) mobility through space, but insufficient

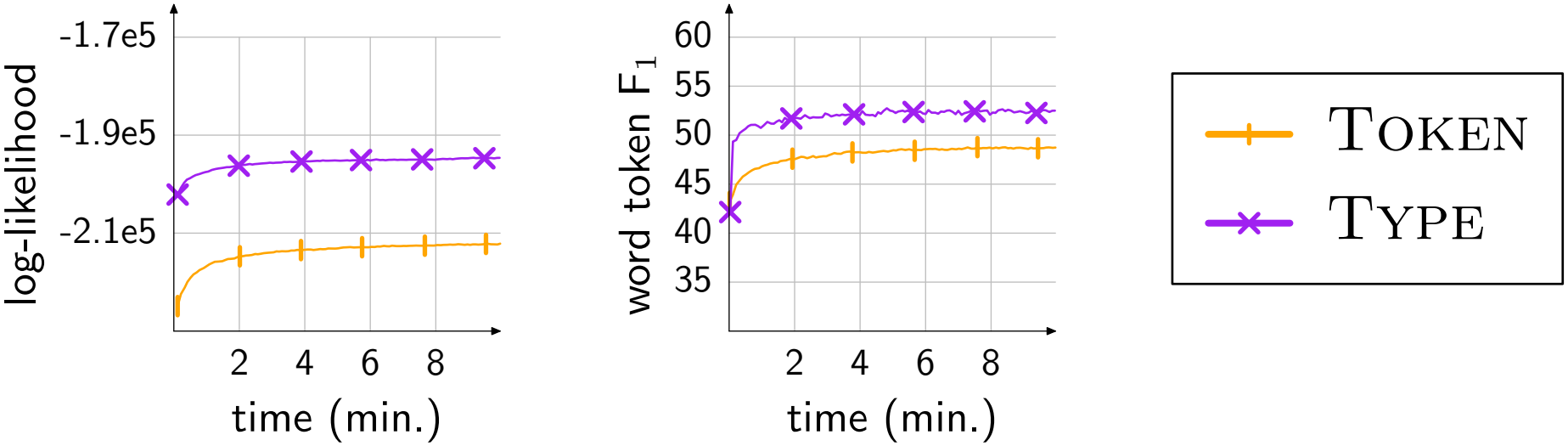
Alternatives

Can we get the gains of TYPE via simpler means?

- Annealing ($\text{TOKEN}_{\text{anneal}}$)
  - Use $p(\mathbf{z} \mid \mathbf{x})^{1/T}$, temperature T from 5 to 1
  - More (random) mobility through space, but insufficient
- Sentence-based (SENTENCE)

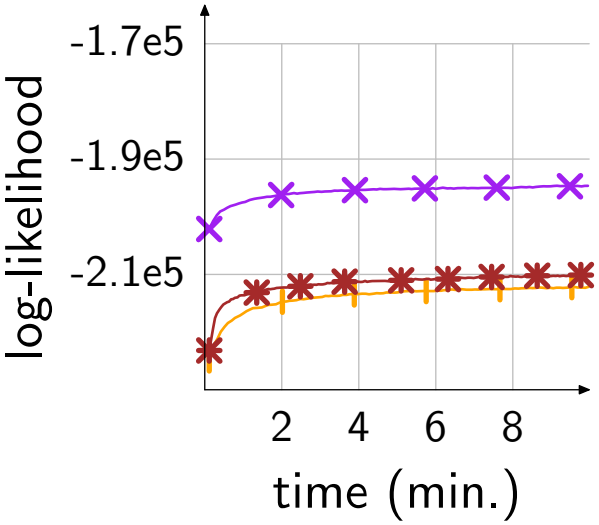
Sentence

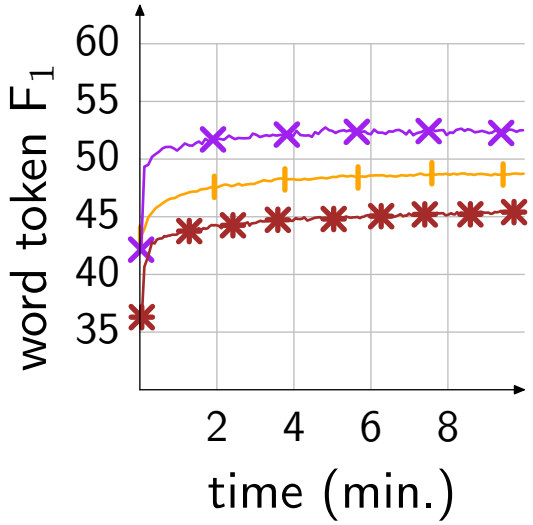
USM:

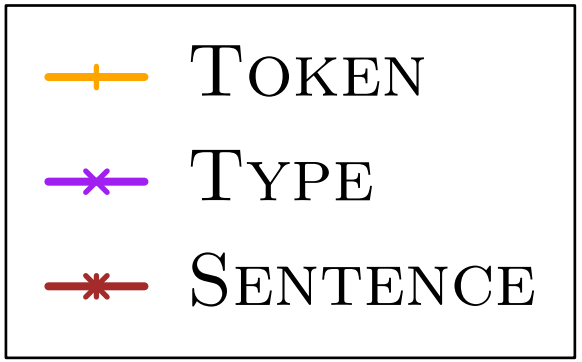


Sentence

USM:

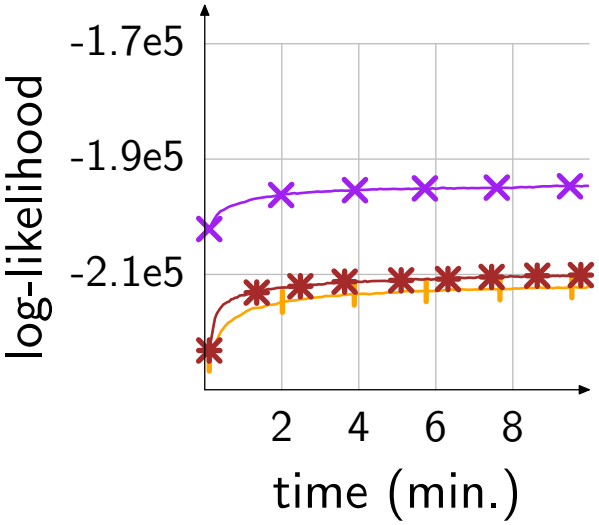


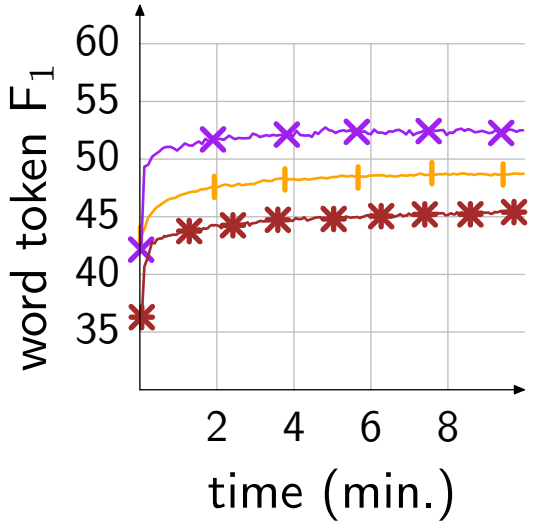


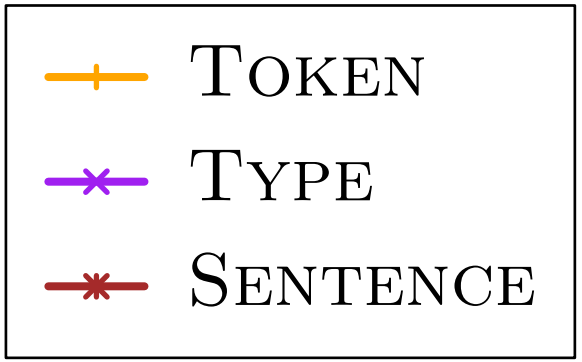


Sentence

USM:





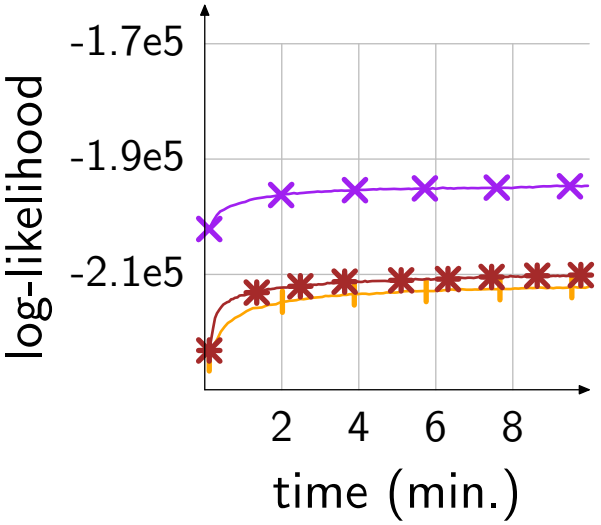


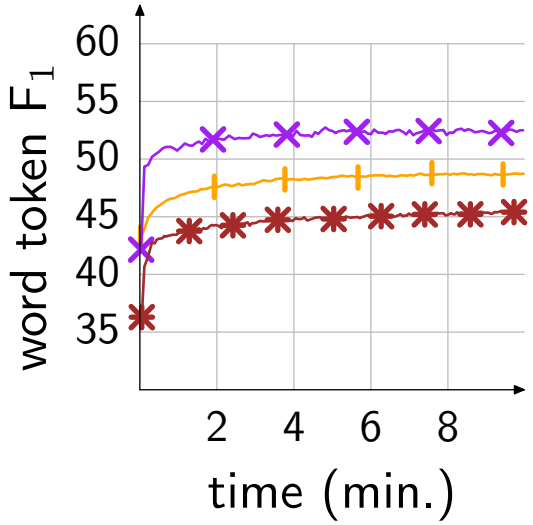
Sentence performs comparably to Token,

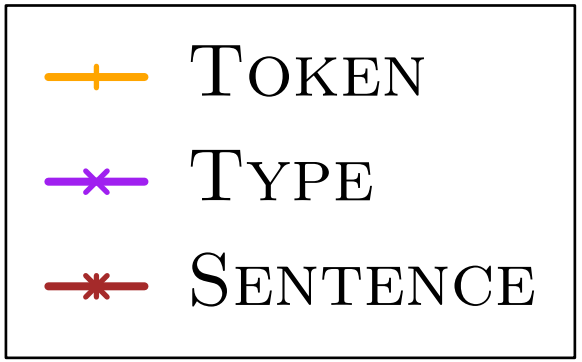
but worse than Type

Sentence

USM:







Sentence performs comparably to Token,
but worse than Type

Sentence requires dynamic programming,
computationally more expensive than Token and Type

Alternatives

Can we get the gains of TYPE via simpler means?

- Annealing ($\text{TOKEN}_{\text{anneal}}$)
  - Use $p(\mathbf{z} \mid \mathbf{x})^{1/T}$, temperature T from 5 to 1
  - More (random) mobility through space, but insufficient
- Sentence-based (SENTENCE)
  - SENTENCE handles structural dependencies
  - TYPE handles parameter dependencies

Alternatives

Can we get the gains of TYPE via simpler means?

- Annealing ($\text{TOKEN}_{\text{anneal}}$)
  - Use $p(\mathbf{z} \mid \mathbf{x})^{1/T}$, temperature T from 5 to 1
  - More (random) mobility through space, but insufficient
- Sentence-based (SENTENCE)
  - SENTENCE handles structural dependencies
  - TYPE handles parameter dependencies
  - Intuition: parameter dependencies more important in unsupervised learning

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure
New: think about **exchangeability**

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure
New: think about **exchangeability**

Tokens versus types:

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure
New: think about **exchangeability**

Tokens versus types:

- Older work operate on **types** (e.g., model merging)

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure
New: think about **exchangeability**

Tokens versus types:

- Older work operate on **types** (e.g., model merging)
  Larger updates, but greedy and brittle

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure
New: think about **exchangeability**

Tokens versus types:

- Older work operate on **types** (e.g., model merging)
  Larger updates, but greedy and brittle
- Recent methods operate on sentences or tokens
  Smaller updates, but softer and more robust

Summary and outlook

General strategy: update many **dependent** variables **tractably**

Variables sharing parameters very dependent

Type-based sampler updates exactly these variables

Techniques for tractability:

Old: exploit dynamic programming structure
New: think about **exchangeability**

Tokens versus types:

- Older work operate on **types** (e.g., model merging)
  Larger updates, but greedy and brittle
- Recent methods operate on sentences or tokens
  Smaller updates, but softer and more robust

Type-based sampling combines advantages of both

Thank you!