Type-Based MCMC

NAACL 2010 – Los Angeles

Percy Liang  Michael I. Jordan  Dan Klein
Type-Based MCMC

NAACL 2010 – Los Angeles

Percy Liang    Michael I. Jordan    Dan Klein
Learning latent-variable models

... some stocks rose ...

... tech stocks fell ...

... the stocks soared ...

... how stocks dipped ...

... many stocks created ...
Learning latent-variable models

- some stocks rose
- tech stocks fell
- the stocks soared
- how stocks dipped
- many stocks created
Learning latent-variable models

\[ \theta \]

... DT NNS VBD ...
... some stocks rose ...
... NN NNS VBD ...
... tech stocks fell ...
... DT NNS VBD ...
... the stocks soared ...
... WRB NNS VBD ...
... how stocks dipped ...
... DT NNS VBD ...
... many stocks created ...
Learning latent-variable models

- Some stocks rose.
- Tech stocks fell.
- The stocks soared.
- How stocks dipped.
- Many stocks created.
Learning latent-variable models

Variables are highly dependent

... some stocks rose ...
... tech stocks fell ...
... the stocks soared ...
... how stocks dipped ...
... many stocks created ...
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) ⇒ 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) ⇒ local optima, slow mixing

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

Structural dependencies

... some stocks rose ...
... tech stocks fell ...
... the stocks soared ...
... how stocks dipped ...
... many stocks created ...
Learning latent-variable models

Variables are highly dependent

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

Classic: sentence-based (e.g., EM)
- Structural dependencies
- dynamic programming

New: type-based

... DT NNS VBD ...
... some stocks rose ...
... NN NNS VBD ...
... tech stocks fell ...
... DT NNS VBD ...
... the stocks soared ...
... WRB NNS VBD ...
... how stocks dipped ...
... DT NNS VBD ...
... many stocks created ...
Learning latent-variable models

Variables are highly dependent

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

worker demands meeting resistance
Structural dependencies

Dependencies between adjacent variables

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

<table>
<thead>
<tr>
<th>VBZ</th>
<th>VBD</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>stocks</td>
<td>rose</td>
<td>...</td>
</tr>
<tr>
<td>VBZ</td>
<td>VBD</td>
<td>...</td>
</tr>
<tr>
<td>stocks</td>
<td>fell</td>
<td>...</td>
</tr>
<tr>
<td>VBZ</td>
<td>VBD</td>
<td>...</td>
</tr>
<tr>
<td>stocks</td>
<td>took</td>
<td>...</td>
</tr>
<tr>
<td>VBZ</td>
<td>VBD</td>
<td>...</td>
</tr>
<tr>
<td>stocks</td>
<td>shot</td>
<td>...</td>
</tr>
</tbody>
</table>
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

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

1. Parameter dependencies create deeper valleys
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 $\theta$: vector of conditional probabilities
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

Parameters $\theta$: vector of conditional probabilities

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

$p(\theta)$ is product of Dirichlets [prior]
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

\begin{align*}
\cdots & \quad \text{V} \quad \text{N} \quad \cdots \\
\text{stocks}
\end{align*}
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

$z \in z$

[T:V,N at 8]
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

$z \in z$ $j(z)$ [parameter used] $\ldots$ $V$ $\rightarrow$ $N$ $\rightarrow$ $\ldots$

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

stocks
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

$z \in z$

[parameter used]

$p(z | \theta) = \prod_{z \in z} \theta_{j(z)}$ [likelihood]
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

$z \in z$ [parameter used] $\cdots \xrightarrow{} V \xrightarrow{} N \xrightarrow{} \cdots$

$T:V,N$ at 8 $T:V,N$

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

$p(z) = \int p(z | \theta)p(\theta)d\theta$ [marginal likelihood]
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

$z \in z$ $j(z)$ [parameter used] 

[T:V,N at 8] $\rightarrow$ $\rightarrow$ 

V N 

stocks

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

$p(z) = \int p(z \mid \theta)p(\theta)d\theta$ [marginal likelihood]

Observations $x$: observed part of $z$ (e.g., the words)
Formal setup

Parameters $\theta$: vector of conditional probabilities

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

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

Choices $z$: specifies values of latent and observed variables

$z \in z$ [parameter used] $j(z)$

$[T:V,N \text{ at 8}] \quad T:V,N$

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

$p(z) = \int p(z | \theta)p(\theta)d\theta$ [marginal likelihood]

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

Goal: sample from $p(z | x)$ [not $p(z | x, \theta)$]
Exchangeability

Sufficient statistics $n$: \# times parameters were used in $z$
Exchangeability

**Sufficient statistics** $n$: $\#$ times parameters were used in $z$

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

Sufficient statistics $n$: # times parameters were used in $z$

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

Rewrite likelihood:

$$p(z \mid \theta) = \prod_j \theta_{nj}^{n_j}$$
Exchangeability

Sufficient statistics \( n \): \# times parameters were used in \( z \)

\( n_{T:V,N} \): \# times that state \( V \) transitioned to state \( N \)

Rewrite likelihood:

\[
p(z \mid \theta) = \prod_j \theta_j^{n_j} = \text{simple function of } n \text{ and } \theta
\]
Exchangeability

Sufficient statistics \( n \): \# times parameters were used in \( z \)

\( n_{T:V,N} \): \# times that state \( V \) transitioned to state \( N \)

Rewrite likelihood:

\[
p(z \mid \theta) = \prod_j \theta_j^{n_j} = \text{simple function of } n \text{ and } \theta
\]

\[
p(z) = \text{simple function of } n
\]
Exchangeability

Sufficient statistics $n$: # times parameters were used in $z$

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

Rewrite likelihood:

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

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

Sufficient statistics \( n \): \# times parameters were used in \( z \)

\( n_{T:V,N} \): \# times that state \( V \) transitioned to state \( N \)

Rewrite likelihood:

\[
p(z \mid \theta) = \prod_j \theta_j^{n_j} = \text{simple function of } n \text{ and } \theta
\]

\[
p(z) = \text{simple function of } n \text{ (key: exchangeability)}
\]
Exchangeability

Sufficient statistics $n$: # times parameters were used in $z$

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

Rewrite likelihood:

$$p(z | \theta) = \prod_j \theta_j^{n_j} = \text{simple function of } n \text{ and } \theta$$

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

Sufficient statistics $n$: # times parameters were used in $z$

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

Rewrite likelihood:

$$p(z | \theta) = \prod_j \theta_j^{n_j} = \text{simple function of } n \text{ and } \theta$$

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

Sufficient statistics \( n \): \# times parameters were used in \( z \)

\( n_{T:V,N} \): \# times that state \( V \) transitioned to state \( N \)

Rewrite likelihood:

\[
p(z \mid \theta) = \prod_j \theta_j^{n_j} = \text{simple function of } n \text{ and } \theta
\]

\[
p(z) = \text{simple function of } n \text{ (key: exchangeability)}
\]

\( z_1 \) and \( z_2 \) have same sufficient statistics

\[
\begin{align*}
\cdots & \rightarrow D \rightarrow V \rightarrow V \rightarrow \cdots \\
& \text{many \ stocks \ rose} \\
\cdots & \rightarrow D \rightarrow N \rightarrow V \rightarrow \cdots \\
& \text{tech \ stocks \ were}
\end{align*}
\]
Exchangeability

Sufficient statistics $n$: # times parameters were used in $z$

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

Rewrite likelihood:

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

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

$z_1$ and $z_2$ have same sufficient statistics $\Rightarrow p(z_1) = p(z_2)$
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

\[ \cdot \cdot \cdot D \cdot \cdot \cdot \rightarrow \cdot \cdot \cdot V \cdot \cdot \cdot \]

\[many \quad stocks \quad rose\]
Types

**Type** of a variable: its dependent parameter components
For HMM, type = assignment to Markov blanket

\[
type(\bigcirc) = (D, stocks, V)
\]

\[
\cdots \rightarrow D \rightarrow \bigcirc \rightarrow V \rightarrow \cdots
\]

many \quad stocks \quad rose
Types

**Type** of a variable: its dependent parameter components
For HMM, type = assignment to Markov blanket

\[
type(\bigcirc) = (D, stocks, V)
\]

![Diagram showing the relationship between D, stocks, and V with arrows indicating dependencies.](attachment:diagram.png)
Types

**Type** of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

\[
\text{type}(\bigcirc) = (D, \text{stocks}, V)
\]

Assignments

\[\text{[VVNN]}\]
Types

**Type** of a variable: its dependent parameter components
For HMM, type = assignment to Markov blanket

\[ \text{type( } \bigcirc \text{ )} = (D, \ stocks, \ V) \]

Assignments

\[ [VVNN] \]
\[ [VNVN] \]

• • •

D → V → V

many stocks rose

tech stocks were

the stocks have

how stocks from
Types

**Type** of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

\[
\text{type}(\bigcirc) = (D, \ stocks, \ V)
\]

\[\cdots \rightarrow D \rightarrow V \rightarrow V \rightarrow \cdots\]

\[\begin{array}{c}
\text{many} \\
\text{tech} \\
\text{the} \\
\text{how}
\end{array} \rightarrow \begin{array}{c}
\text{stocks} \\
\text{stocks} \\
\text{stocks} \\
\text{stocks}
\end{array} \rightarrow \begin{array}{c}
\text{rose} \\
\text{were} \\
\text{have} \\
\text{from}
\end{array} \rightarrow \cdots
\]

Assignments

\[\begin{bmatrix}
VVNN \\
VNVN \\
VNNV
\end{bmatrix}\]
Types

**Type** of a variable: its dependent parameter components

For HMM, type = assignment to Markov blanket

\[
\text{type}(\bigcirc) = (D, \text{ stocks, V})
\]

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

[VVNNN]
[VNNNN]
[VNNVN]
[NVVNN]
[NVNVT]
Types

Type of a variable: its dependent parameter components
For HMM, type = assignment to Markov blanket

\[ \text{type}(\bigcirc) = (D, \text{stocks}, V) \]

Assignments

\[ [VVNN] \]
\[ [VNVN] \]
\[ [VNNV] \]
\[ [NVVN] \]
\[ [NVNV] \]
\[ [NNVV] \]
**Types**

**Type** of a variable: its dependent parameter components
For HMM, type = assignment to Markov blanket

\[
\text{type}(\bigcirc) = (D, \text{stocks}, V)
\]

\[
\begin{array}{ccc}
\ldots & D & \rightarrow & N & \rightarrow & V & \rightarrow & \ldots \\
\downarrow & & & \downarrow & & & \downarrow & \\
\text{many} & \downarrow & \text{stocks} & \rightarrow & \text{rose} & \\
\ldots & D & \rightarrow & N & \rightarrow & V & \rightarrow & \ldots \\
\downarrow & & & \downarrow & & & \downarrow & \\
\text{tech} & \downarrow & \text{stocks} & \rightarrow & \text{were} & \\
\ldots & D & \rightarrow & V & \rightarrow & V & \rightarrow & \ldots \\
\downarrow & & & \downarrow & & & \downarrow & \\
\text{the} & \downarrow & \text{stocks} & \rightarrow & \text{have} & \\
\ldots & D & \rightarrow & V & \rightarrow & V & \rightarrow & \ldots \\
\downarrow & & & \downarrow & & & \downarrow & \\
\text{how} & \downarrow & \text{stocks} & \rightarrow & \text{from} & \\
\end{array}
\]

Assignments

\[
\begin{array}{c}
[VVNN] \\
[VNVN] \\
[VNNV] \\
[NNVV] \\
[NVVN] \\
[NVNV] \\
[NNVV]
\end{array}
\]

\[p(\text{assignment}) \text{ only depends on number of } V \text{ and } N\]
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

<table>
<thead>
<tr>
<th>$m = 0$</th>
<th>$m = 1$</th>
<th>$m = 2$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
</tr>
</thead>
</table>
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. $\propto \binom{B}{m} p_m$
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 $m$
Full algorithm

Iterate:

\[ \text{many stocks rose} \]
\[ \text{tech stocks were} \]
\[ \text{the stocks have} \]
\[ \text{how stocks from} \]
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., \((D, \text{stocks}, V)\)
3. Sample \(m\)
e.g., 1 \(\Rightarrow\) \(\{V, V, V, N\}\)
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., [VNVVV]
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

- tech stocks rose in heavy trading
- worker demands meeting resistance
- many stocks shot up today
- investors await stocks news
Experimental setup: sampling algorithms

Token-based sampler (TOKEN)
1. Choose token
2. Update conditioned on rest

Tech stocks rose in heavy trading
Worker demands meeting resistance
Many stocks shot up today
Investors await stocks news
Experimental setup: sampling algorithms

Token-based sampler (Token)
1. Choose token
2. Update conditioned on rest

Sentence-based sampler (Sentence)
1. Choose sentence

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
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)

Part-of-speech induction
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

- worker demands meeting resistance

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

Dataset: WSJ
(49K sentences, 45 tags)

Part-of-speech induction

Unigram segmentation model (USM) [Goldwater et al., 2006]
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

Dataset: WSJ
(49K sentences, 45 tags)

Part-of-speech induction

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

Word segmentation
Experimental setup: models/tasks/datasets

Hidden Markov model (**HMM**)  
- **worker** demands **meeting** resistance  
- Part-of-speech induction

**Unigram segmentation model (**USM**)** [Goldwater et al., 2006]  
- **look** a t t h e b o o k  
- Word segmentation

**Dataset**: WSJ  
(49K sentences, 45 tags)
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

Part-of-speech induction

Dataset: WSJ
(49K sentences, 45 tags)

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

Word segmentation

Dataset: CHILDES
(9.7K sentences)
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

Worker demands meeting resistance

Part-of-speech induction

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

Look at the book

Word segmentation

Probabilistic tree-substitution grammar (PTSG) [Cohn et al., 2009]
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

worker demands meeting resistance

Part-of-speech induction

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

Dataset: WSJ
(49K sentences, 45 tags)

Word segmentation

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

S

NP

VP

DT NN VBD VBN

the sun has risen
Experimental setup: models/tasks/datasets

Hidden Markov model (HMM)

Worker demands meeting resistance

Part-of-speech induction

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

Look at the book

Word segmentation

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

S

NP

DT the

NN sun

VBP has

VBN risen

Dataset: WSJ
(49K sentences, 45 tags)

Dataset: CHILDES
(9.7K sentences)
Experimental setup: models/tasks/datasets

**Hidden Markov model (HMM)**

- Worker demands meeting resistance

  Dataset: WSJ
  (49K sentences, 45 tags)

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

- Look at the book

  Dataset: CHILDES
  (9.7K sentences)

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

- The sun has risen

  Dataset: WSJ
  (49K sentences, 45 tags)
<table>
<thead>
<tr>
<th>Token versus Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMM</td>
</tr>
</tbody>
</table>
Token versus Type

HMM  USM  PTSG

![Graph showing log-likelihood versus time for HMM, USM, and PTSG with token highlighted.](image-url)
Token versus Type

HMM  USM  PTSG

-7.3e6  -6.9e6  -6.5e6

log-likelihood

time (hr.)

Token
Type
Token versus Type

HMM

log-likelihood

-7.3e6
-6.9e6
-6.5e6

time (hr.)

3 6 9 12

USM

log-likelihood

-1.7e5
-1.9e5
-2.1e5

time (min.)

2 4 6 8

PTSG

Token

Type

-0x-0
Token versus Type

HMM

USM

PTSG

Token

Type

log-likelihood

log-likelihood

log-likelihood

3 6 9 12

2 4 6 8

3 6 9 12

time (hr.)

time (min.)

time (hr.)

-7.3e6

-6.9e6

-6.5e6

-1.7e5

-1.9e5

-2.1e5

-5.3e6

-5.5e6

-5.7e6

-5.9e6

-6.1e6

-6.5e6

-6.9e6

-7.3e6

-2.1e5

-1.9e5

-1.7e5

-5.3e6

-5.5e6

-5.7e6

-5.9e6

-6.1e6

-2.1e5

-1.9e5

-1.7e5

-5.3e6

-5.5e6

-5.7e6

-5.9e6

-6.1e6

3 6 9 12

2 4 6 8

3 6 9 12

time (hr.)

time (min.)

time (hr.)

Token

Type
Token versus Type

HMM

USM

PTSG

<table>
<thead>
<tr>
<th>Time (hr.)</th>
<th>Log-likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-7.3e6</td>
</tr>
<tr>
<td>6</td>
<td>-6.9e6</td>
</tr>
<tr>
<td>9</td>
<td>-6.5e6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time (min.)</th>
<th>Log-likelihood</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-1.7e5</td>
</tr>
<tr>
<td>4</td>
<td>-1.9e5</td>
</tr>
<tr>
<td>6</td>
<td>-2.1e5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time (hr.)</th>
<th>Tag Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>12</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time (hr.)</th>
<th>Word Token F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>12</td>
<td>50</td>
</tr>
</tbody>
</table>
Sensitivity to initialization
Sensitivity to initialization

(Use few params.)

\[ \eta = 0 \]
Sensitivity to initialization

(use few params.)

$\eta = 0$

worker demands meeting resistance
Sensitivity to initialization

(use few params.)

\[ \eta = 0 \]

worker demands meeting resistance

look at the book
Sensitivity to initialization

(use few params.)
\[ \eta = 0 \]

(use many params.)
\[ \eta = 1 \]

\[ \text{look at the book} \]
Sensitivity to initialization

(\text{use few params.}) \quad \eta = 0 \quad \text{\text{\hspace{1cm}}} \quad (\text{use many params.}) \quad \eta = 1

\begin{align*}
\text{V V V V V} & \quad \text{P N D V} \\
\text{worker demands meeting resistance} & \quad \text{worker demands meeting resistance}
\end{align*}

| l o o k a t t h e b o o k |
Sensitivity to initialization

(use few params.)
\[ \eta = 0 \]

(use many params.)
\[ \eta = 1 \]

\[ \text{worker demands meeting resistance} \]

look at the book

look at the book
<table>
<thead>
<tr>
<th>HMM</th>
<th>USM</th>
<th>PTSG</th>
</tr>
</thead>
</table>

Sensitivity to initialization
Sensitivity to initialization

- HMM
- USM
- PTSG

- Log-likelihood
- Tag accuracy

η
Sensitivity to initialization

HMM  USM  PTSG

-7.2e6
-7.0e6
-6.8e6
-6.6e6

log-likelihood

0.2  0.4  0.6  0.8  1.0

η

-6.6e6
-6.8e6
-7.0e6
-7.2e6

tag accuracy

0.2  0.4  0.6  0.8  1.0

η

10  20  30  40  50  60

tag accuracy Token Type

15
Sensitivity to initialization

HMM

USM

PTSG

log-likelihood

η

-7.2e6
-7.0e6
-6.8e6
-6.6e6

log-likelihood

η

-3.4e5
-3.1e5
-2.7e5
-2.3e5

η

-1.9e5
-2.3e5
-2.7e5
-3.1e5

η

10
20
30
40
50
60
tag accuracy

0.2 0.4 0.6 0.8 1.0

η

15

word token F1

Token

Type

0.2 0.4 0.6 0.8 1.0

η
Sensitivity to initialization

HMM

USM

PTSG

-7.2e6
-7.0e6
-6.8e6
-6.6e6

log-likelihood

0.2 0.4 0.6 0.8 1.0

η

-3.4e5
-3.1e5
-2.7e5
-2.3e5

log-likelihood

0.2 0.4 0.6 0.8 1.0

η

-5.7e6
-5.6e6
-5.5e6
-5.4e6

log-likelihood

0.2 0.4 0.6 0.8 1.0

η

10
20
30
40
50
60
tag accuracy

0.2 0.4 0.6 0.8 1.0

η

10
20
30
40
50
60
word token F1

Token

Type

Figures showing the sensitivity of log-likelihood and tag accuracy to initialization for HMM, USM, and PTSG models.
Sensitivity to initialization

**HMM**

- Log-likelihood
- Tag accuracy

**USM**

- Log-likelihood
- Word token F1

**PTSG**

- Log-likelihood

*Type* less sensitive than *Token*
Alternatives

Can we get the gains of Type via simpler means?
Alternatives

Can we get the gains of \textbf{Type} via simpler means?

• Annealing (\textit{Token}_{\text{anneal}})
  
  − Use $p(z \mid x)^{1/T}$, temperature $T$ from 5 to 1
Annealing

HMM  USM  PTSG
Annealing

HMM

USM

PTSG

log-likelihood

3 6 9 12
time (hr.)

-7.3e6
-6.9e6
-6.5e6

Tag accuracy

3 6 9 12
time (hr.)

35
40
45
50
55
60

Token

Type
Annealing

HMM

USM

PTSG

log-likelihood

time (hr.)

-7.3e6

-6.9e6

-6.5e6

3 6 9 12

tag accuracy

Token

Token

anneal

Type

[anneal. hurts]
Annealing

HMM

log-likelihood

-7.3e6

-6.9e6

-6.5e6

3 6 9 12
time (hr.)

USM

log-likelihood

-1.7e5

-1.9e5

-2.1e5

2 4 6 8
time (min.)

PTSG

word token F1

Token

Token

anneal

Type

[anneal. hurts] [anneal. helps]

[Token] [Tokenₐnnₐnₑₐₙ] [Type]

accuracy

35

40

45

50

55

60

time (hr.)

log-likelihood

3 6 9 12
time (hr.)
Annealing

HMM

-7.3e6
-6.9e6
-6.5e6

log-likelihood

2 4 6 8

time (min.)

3 6 9 12
time (hr.)

USM

-1.7e5
-1.9e5
-2.1e5

log-likelihood

3 6 9 12
time (hr.)

PTSG

-5.3e6
-5.5e6
-5.7e6
-5.9e6
-6.1e6

log-likelihood

3 6 9 12
time (hr.)

Tag accuracy

60
55
50
45
40
35

3 6 9 12
time (hr.)

[anneal. hurts]

Word token F1

60
55
50
45
40
35

2 4 6 8
time (min.)

[anneal. helps]

[no effect]
Alternatives

Can we get the gains of \textbf{Type} via simpler means?

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

Can we get the gains of \textbf{TYPE} via simpler means?

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

• Sentence-based (\texttt{SENTENCE})
USM:

- Log-likelihood vs. time (min.)
- Word token F1 vs. time (min.)

-2.1e5, -1.9e5, -1.7e5

Token and Type graphs
**Sentence** performs comparably to **Token**, but worse than **Type**
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** *(Token\text{anneal})*
  - Use $p(z \mid x)^{1/T}$, 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 $p(z \mid x)^{1/T}$, temperature $T$ from 5 to 1
  - More (random) mobility through space, but insufficient

- Sentence-based ($\text{Sentence}$)
  - $\text{Sentence}$ 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!