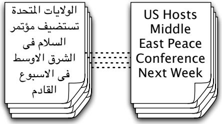
Fully Distributed EM for Very Large Datasets

Jason Wolfe   Aria Haghighi   Dan Klein

Computer Science Division
UC Berkeley

Overview

- Task: unsupervised learning via EM

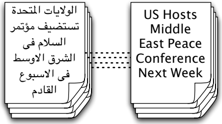


- Focus: models w/ many local parameters
  (relevant to few datums)

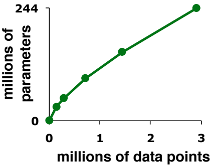
- Approach: fully distributed, localized EM
  - ⋆ *parameter locality* → less bandwidth

Overview

- Task: unsupervised learning via EM



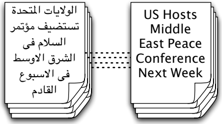
- Focus: models w/ many local parameters
  (relevant to few datums)



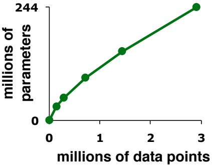
- Approach: fully distributed, localized EM
  - ⋆ *parameter locality* → less bandwidth

Overview

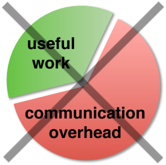
- Task: unsupervised learning via EM



- Focus: models w/ many local parameters
  (relevant to few datums)



- Approach: fully distributed, localized EM
  - ⋆ *parameter locality* → less bandwidth



Outline

- Running example: IBM Model 1 for word alignment
- Naive distributed EM
- Efficiently distributed EM

Word alignment for machine translation

- Goal: parallel sentences $\longrightarrow$
  word-level translation model
- Parameters $\theta_{s \to t}$:
  probability that Spanish word s
  translates to English word t

$$\boldsymbol{\theta} = \left\{ \begin{array}{l} \theta_{\text{la}\to\text{the}} \\ \theta_{\text{la}\to\text{chair}} \\ \theta_{\text{la}\to\text{table}} \\ \hline \theta_{\text{silla}\to\text{the}} \\ \theta_{\text{silla}\to\text{chair}} \\ \hline \theta_{\text{mesa}\to\text{the}} \\ \theta_{\text{mesa}\to\text{table}} \end{array} \right.$$

la silla $\Downarrow$ **the chair**

la mesa $\Downarrow$ **the table**

corpus of parallel sentences

Word alignment for machine translation

- Goal: parallel sentences $\longrightarrow$ word-level translation model
- Parameters $\theta_{s \to t}$: probability that Spanish word s translates to English word t

$$\boldsymbol{\theta} = \left\{ \begin{array}{l} \theta_{\text{la}\to\text{the}} \\ \theta_{\text{la}\to\text{chair}} \\ \theta_{\text{la}\to\text{table}} \\ \hline \theta_{\text{silla}\to\text{the}} \\ \theta_{\text{silla}\to\text{chair}} \\ \hline \theta_{\text{mesa}\to\text{the}} \\ \theta_{\text{mesa}\to\text{table}} \end{array} \right.$$

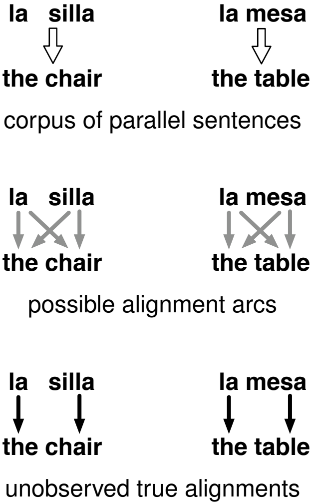
corpus of parallel sentences

possible alignment arcs

Word alignment for machine translation

- Goal: parallel sentences $\rightarrow$ word-level translation model
- Parameters $\theta_{s \rightarrow t}$: probability that Spanish word s translates to English word t

$$\boldsymbol{\theta} = \left\{ \begin{array}{lcl} \theta_{\text{la}\rightarrow\text{the}} & = & 1.0 \\ \theta_{\text{la}\rightarrow\text{chair}} & = & 0.0 \\ \theta_{\text{la}\rightarrow\text{table}} & = & 0.0 \\ \hline \theta_{\text{silla}\rightarrow\text{the}} & = & 0.0 \\ \theta_{\text{silla}\rightarrow\text{chair}} & = & 1.0 \\ \hline \theta_{\text{mesa}\rightarrow\text{the}} & = & 0.0 \\ \theta_{\text{mesa}\rightarrow\text{table}} & = & 1.0 \end{array} \right.$$



IBM Model 1 for word alignment

a Steve no le gustan las ferias grandes

? ? ? ? ? ? ?

each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{\mathrm{S}_{\mathrm{a_i}}\rightarrow\cdot}$

IBM Model 1 for word alignment

a Steve no le gustan las ferias grandes

?   ?   ?   ?   ?   ?   ?

each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{\mathrm{s}_{\mathrm{a_i}} \rightarrow}$.

IBM Model 1 for word alignment

a Steve no le gustan las ferias grandes

Steve ? ? ? ? ? ?

each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{\mathrm{s}_{a_i}\rightarrow}$.

IBM Model 1 for word alignment

a Steve no le gustan las ferias grandes

Steve ? ? ? ? ? ?

each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{s_{a_i} \rightarrow \cdot}$

IBM Model 1 for word alignment

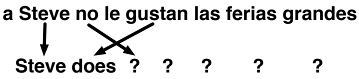
a Steve no le gustan las ferias grandes

Steve does ? ? ? ? ?

each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{\mathrm{s}_{\mathrm{a_i}} \rightarrow \cdot}$

IBM Model 1 for word alignment



each target word is generated by exactly one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{S_{a_i} \rightarrow \cdot}$

IBM Model 1 for word alignment

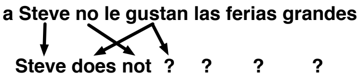
a Steve no le gustan las ferias grandes

Steve does not ? ? ? ?

each target word is generated by exactly one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{s_{a_i} \rightarrow}$.

IBM Model 1 for word alignment



each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{s_{a_i} \rightarrow \cdot}$

IBM Model 1 for word alignment

a Steve no le gustan las ferias grandes

Steve does not like ? ? ?

each target word is generated by exactly
one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{s_{a_i} \rightarrow \cdot}$.

IBM Model 1 for word alignment

each target word is generated by exactly one source word chosen u.a.r

- IBM Model 1: a simple generative model
  - For each target position i, independently
    - choose a source index a_i u.a.r.
    - choose a target word $T_i \sim \theta_{S_{a_i} \rightarrow \cdot}$.

EM algorithm for IBM Model 1

- $\boldsymbol{\theta} \leftarrow$ some initial guess

$\theta_{\text{la}\to\text{the}}$=.33, $\theta_{\text{la}\to\text{chair}}$=.33,
$\theta_{\text{la}\to\text{table}}$=.33, $\theta_{\text{silla}\to\text{the}}$=.5,...

EM algorithm for IBM Model 1

- $\boldsymbol{\theta} \leftarrow$ some initial guess

$\theta_{\text{la}\to\text{the}}$=.33, $\theta_{\text{la}\to\text{chair}}$=.33,
$\theta_{\text{la}\to\text{table}}$=.33, $\theta_{\text{silla}\to\text{the}}$=.5,...

- Iterate:
  1. E-step: estimate alignment counts $\boldsymbol{\eta}$
     1. compute posteriors $p(a_i|\boldsymbol{\theta})$

la **silla**

$\frac{.33}{.33+.5} = .4$ $\quad$ $.6 = \frac{.5}{.33+.5}$

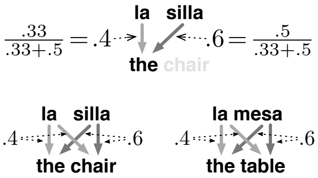
the chair

EM algorithm for IBM Model 1

- $\boldsymbol{\theta} \leftarrow$ some initial guess

$\theta_{\text{la}\to\text{the}}$=.33, $\theta_{\text{la}\to\text{chair}}$=.33,
$\theta_{\text{la}\to\text{table}}$=.33, $\theta_{\text{silla}\to\text{the}}$=.5,...

- Iterate:
  1. E-step: estimate alignment counts $\boldsymbol{\eta}$
     1. compute posteriors $p(a_i|\boldsymbol{\theta})$



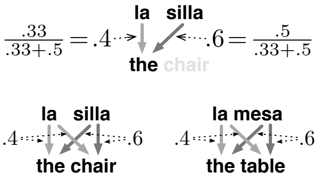
EM algorithm for IBM Model 1

- $\boldsymbol{\theta} \leftarrow$ some initial guess

$\theta_{\text{la}\to\text{the}}$=.33, $\theta_{\text{la}\to\text{chair}}$=.33,
$\theta_{\text{la}\to\text{table}}$=.33, $\theta_{\text{silla}\to\text{the}}$=.5,...

- Iterate:
  1. E-step: estimate alignment counts $\boldsymbol{\eta}$
     1. compute posteriors $p(a_i|\boldsymbol{\theta})$
     2. aggregate into expected counts $\eta_{s\to t}$ (expected # times $s \to t$ under $\boldsymbol{\theta}$)

$$\eta_{s\to t} \leftarrow \sum_{\mathcal{C}} \frac{\theta_{s\to t}}{\sum_{i'} \theta_{s_{i'}\to t}}$$



$\eta_{\text{la}\to\text{the}}$=.8, $\eta_{\text{la}\to\text{chair}}$=.4,
$\eta_{\text{la}\to\text{table}}$=.4, $\eta_{\text{silla}\to\text{the}}$=.6,...

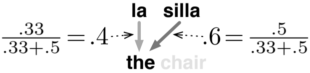
EM algorithm for IBM Model 1

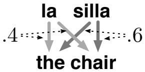
- $\boldsymbol{\theta} \leftarrow$ some initial guess

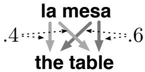
$\theta_{\text{la}\to\text{the}}$=.33, $\theta_{\text{la}\to\text{chair}}$=.33,
$\theta_{\text{la}\to\text{table}}$=.33, $\theta_{\text{silla}\to\text{the}}$=.5,...

- Iterate:
  1. E-step: estimate alignment counts $\boldsymbol{\eta}$
     1. compute posteriors $p(a_i|\boldsymbol{\theta})$
     2. aggregate into expected counts $\eta_{s\to t}$ (expected # times $s\to t$ under $\boldsymbol{\theta}$)

$$\eta_{s\to t} \leftarrow \sum_{\mathcal{C}} \frac{\theta_{s\to t}}{\sum_{i'} \theta_{s_{i'}\to t}}$$







$\eta_{\text{la}\to\text{the}}$=.8, $\eta_{\text{la}\to\text{chair}}$=.4,
$\eta_{\text{la}\to\text{table}}$=.4, $\eta_{\text{silla}\to\text{the}}$=.6,...

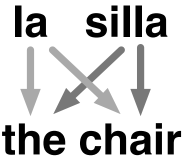
  2. M-step: normalize $\boldsymbol{\eta}$ to get new ML $\boldsymbol{\theta}$

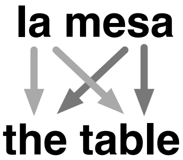
$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

$\theta_{\text{la}\to\text{the}}$=.5, $\theta_{\text{la}\to\text{chair}}$=.25,
$\theta_{\text{la}\to\text{table}}$=.25, $\theta_{\text{silla}\to\text{the}}$=.5,...

EM example continued

E-Step 1





E-Step 2

EM example continued

E-Step 3

E-Step 4

EM example continued

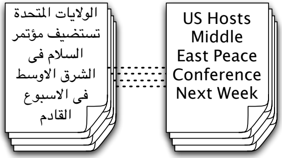
E-Step 5

EM example continued

E-Step ∞

la	**silla**	**la**	**mesa**
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
the	**chair**	**the**	**table**

UN Arabic English TIDES v2 corpus



- 2.9 million sentence pairs from UN proceedings
- 243 million *unique* word pairs
  (translations possible in some sentence pair)
  - 243 M parameters in θ
  - 243 M counts in η
- Even fitting all (indexed) parameters in
  32-bit memory can be challenging

Outline

- Running example: IBM Model 1 for word alignment
- Naive distributed EM
- Efficiently distributed EM

Previous approach: distributing the E-step

1. E-step computations distribute easily
   - partition data over k nodes
   - alignments independent given θ
2. Nodes communicate partial counts to central Reduce node
3. Reduce node does global M-step
4. Reduce sends new parameters back

- Remaining problems:
  - Memory at Reduce node
  - C-step (communication) bandwidth: 5.5 B numbers per iteration (on full dataset with 20 nodes)

(Chu *et al.* 2006, Dyer *et al.* 2008, Newman *et al.* 2008, ...)

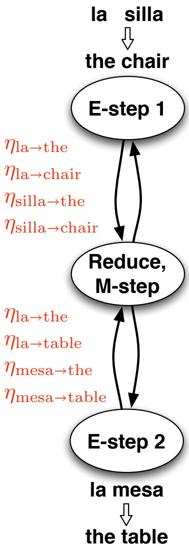
Previous approach: distributing the E-step

1. E-step computations distribute easily
   - partition data over k nodes
   - alignments independent given θ
2. Nodes communicate partial counts to central Reduce node
3. Reduce node does global M-step
4. Reduce sends new parameters back

- Remaining problems:
  - Memory at Reduce node
  - C-step (communication) bandwidth: 5.5 B numbers per iteration (on full dataset with 20 nodes)

(Chu *et al.* 2006, Dyer *et al.* 2008, Newman *et al.* 2008, ...)

Previous approach: distributing the E-step

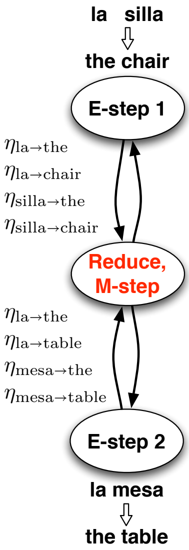


1. E-step computations distribute easily
   - partition data over *k* nodes
   - alignments independent given θ
2. Nodes communicate partial counts to central Reduce node
3. Reduce node does global M-step
4. Reduce sends new parameters back

- Remaining problems:
  - Memory at Reduce node
  - C-step (communication) bandwidth: 5.5 B numbers per iteration (on full dataset with 20 nodes)

(Chu *et al.* 2006, Dyer *et al.* 2008, Newman *et al.* 2008, ...)

Previous approach: distributing the E-step

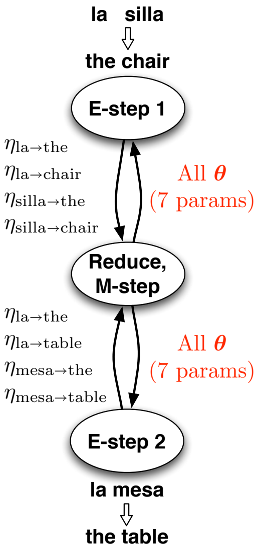


1. E-step computations distribute easily
   - partition data over *k* nodes
   - alignments independent given θ
2. Nodes communicate partial counts to central Reduce node
3. Reduce node does global M-step
4. Reduce sends new parameters back

- Remaining problems:
  - Memory at Reduce node
  - C-step (communication) bandwidth: 5.5 B numbers per iteration (on full dataset with 20 nodes)

(Chu *et al.* 2006, Dyer *et al.* 2008, Newman *et al.* 2008, ...)

Previous approach: distributing the E-step

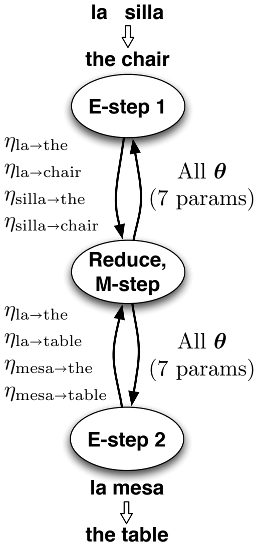


1. E-step computations distribute easily
   - partition data over *k* nodes
   - alignments independent given $\boldsymbol{\theta}$
2. Nodes communicate partial counts to central Reduce node
3. Reduce node does global M-step
4. Reduce sends new parameters back

- Remaining problems:
  - Memory at Reduce node
  - C-step (communication) bandwidth: 5.5 B numbers per iteration (on full dataset with 20 nodes)

(Chu *et al.* 2006, Dyer *et al.* 2008, Newman *et al.* 2008, ...)

Previous approach: distributing the E-step

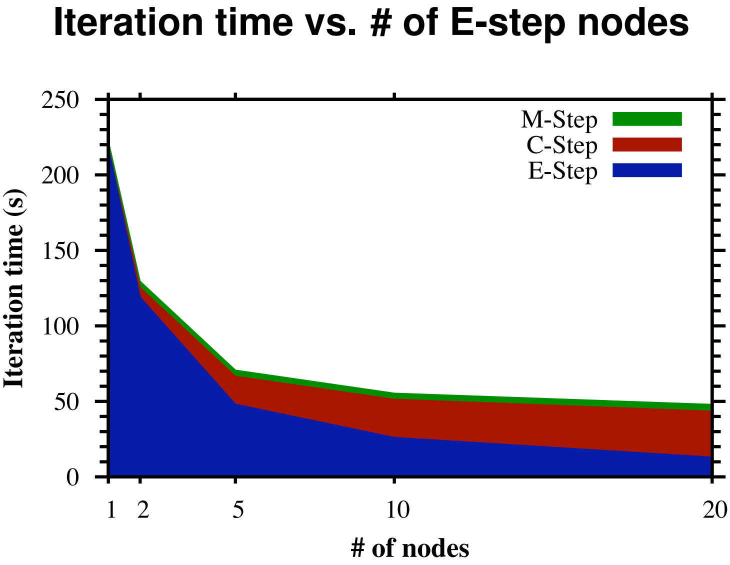


1. E-step computations distribute easily
   - partition data over *k* nodes
   - alignments independent given $\boldsymbol{\theta}$
2. Nodes communicate partial counts to central Reduce node
3. Reduce node does global M-step
4. Reduce sends new parameters back

- Remaining problems:
  - Memory at Reduce node
  - C-step (communication) bandwidth: 5.5 B numbers per iteration (on full dataset with 20 nodes)

(Chu *et al.* 2006, Dyer *et al.* 2008, Newman *et al.* 2008, ...)

Speedup (on 200K total sentence pairs)



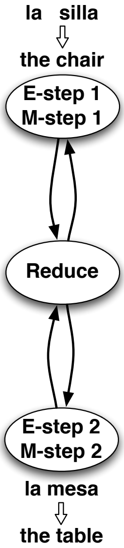
Common practical solutions

- Memory and bandwidth are real problems in practice
- Workarounds
  - Use less data
  - Ignore rare words
  - Train on independent chunks
  - Swap to disk
  - Distribute over multiple machines

Outline

- Running example: IBM Model 1 for word alignment
- Naive distributed EM
- Efficiently distributed EM

Distributing the M-step locally

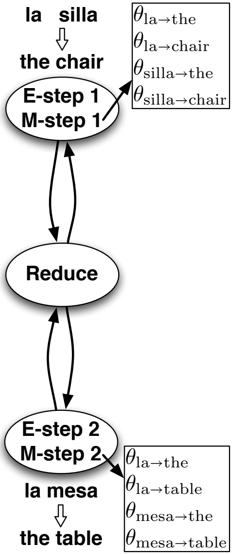


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s \to t} \leftarrow \frac{\eta_{s \to t}}{\sum_{t'} \eta_{s \to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

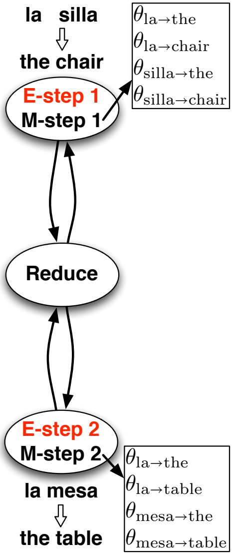


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

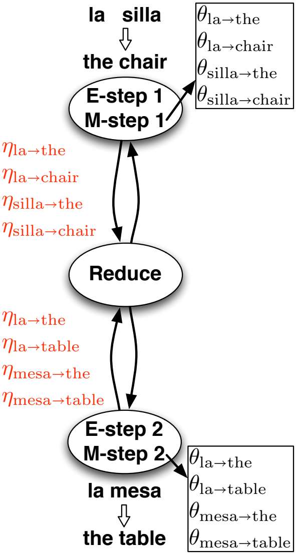


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

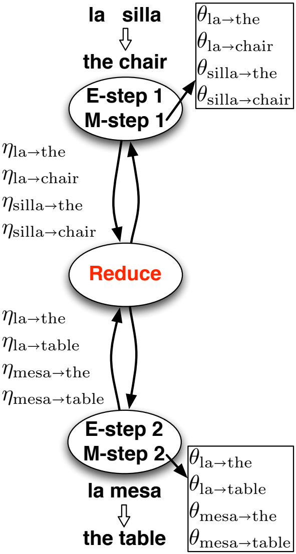


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

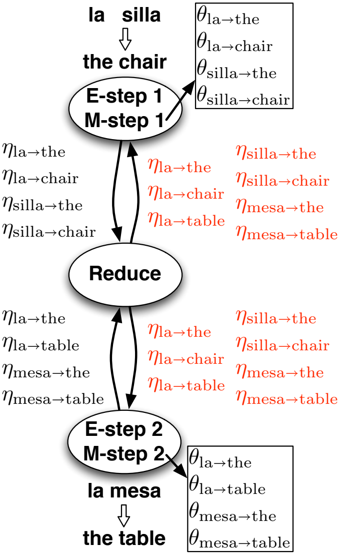


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

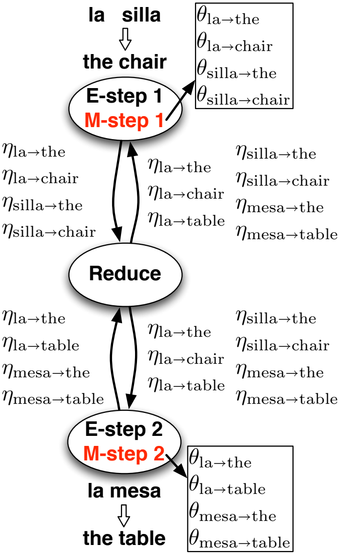


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

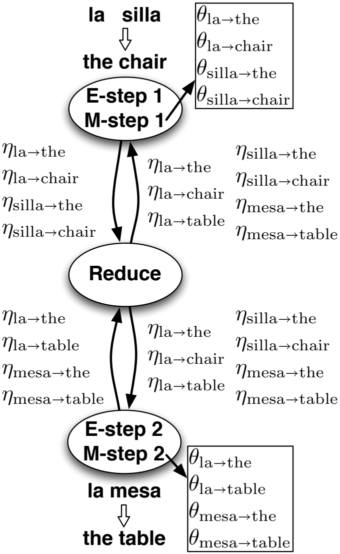


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

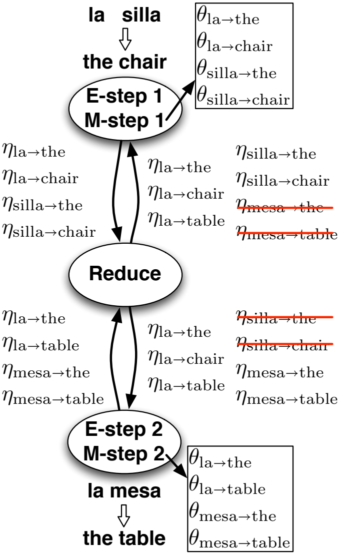


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

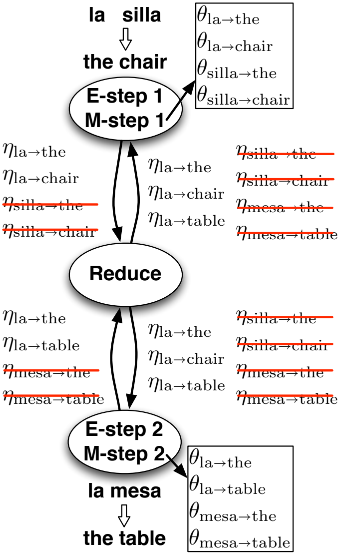


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

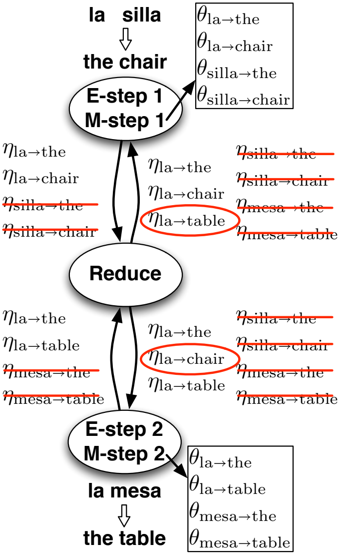


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

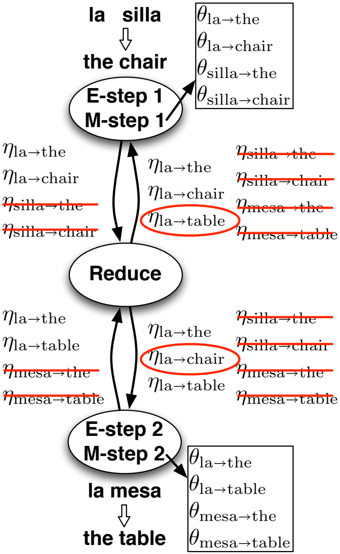


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

  - Bandwidth savings: 30%

Distributing the M-step locally

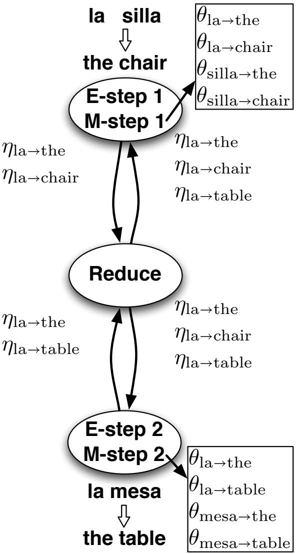


- Distribute M-step alongside E-step
  - Nodes store only needed params, compute them locally
  - Reduce passes back counts
  - Don't need to hear about irrelevant source words
  - Don't need to tell (or hear) about purely local source words
  - Need to hear everything about each source word: M-step denominator

$$\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\sum_{t'} \eta_{s\to t'}}$$

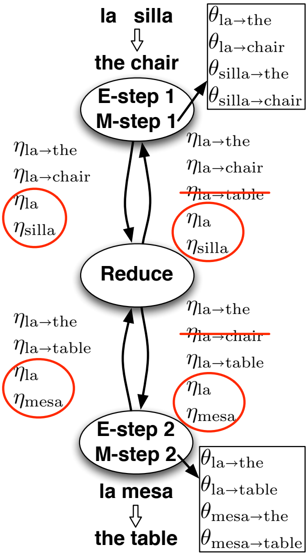
  - Bandwidth savings: 30%

Augmenting η to increase locality



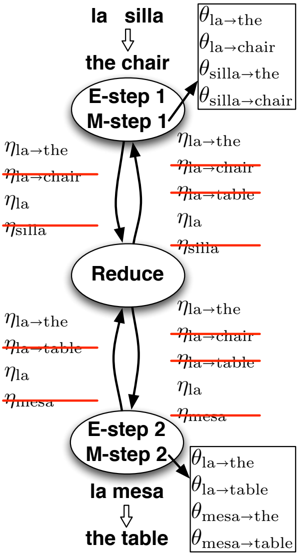
- Augment η with redundant $\eta_s = \sum_{t'} \eta_{s\to t'}$ in E-step
- M-step becomes $\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\eta_s}$
- Increases locality
- Total bandwidth savings: 84% (bigger if more nodes)
- Similar tricks for other models

Augmenting η to increase locality



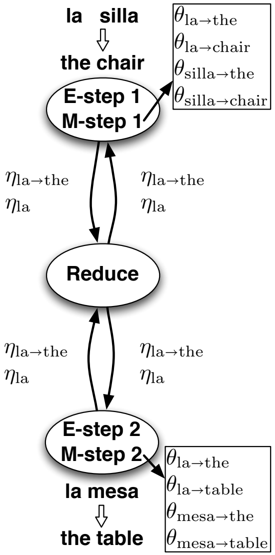
- Augment η with redundant $\eta_s = \sum_{t'} \eta_{s\to t'}$ in E-step
- M-step becomes $\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\eta_s}$
- Increases locality
- Total bandwidth savings: 84% (bigger if more nodes)
- Similar tricks for other models

Augmenting η to increase locality



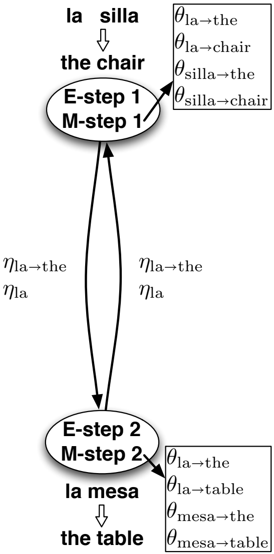
- Augment η with redundant $\eta_s = \sum_{t'} \eta_{s\to t'}$ in E-step
- M-step becomes $\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\eta_s}$
- Increases locality
- Total bandwidth savings: 84% (bigger if more nodes)
- Similar tricks for other models

Augmenting η to increase locality



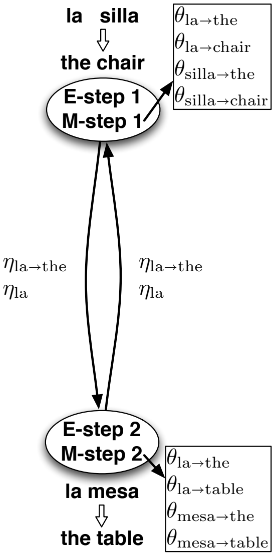
- Augment η with redundant $\eta_s = \sum_{t'} \eta_{s\to t'}$ in E-step
- M-step becomes $\theta_{s\to t} \leftarrow \frac{\eta_{s\to t}}{\eta_s}$
- Increases locality
- Total bandwidth savings: 84% (bigger if more nodes)
- Similar tricks for other models

Choice of C-step topology

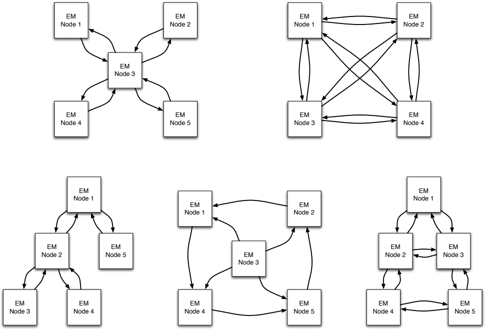


- No need for separate Reduce nodes
- By choosing connectivity, can trade off
  - bandwidth
  - latency
  - locality
  - ...

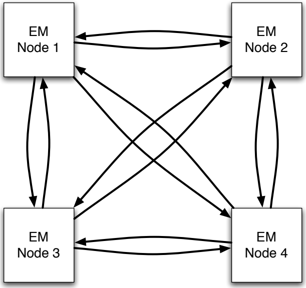
Choice of C-step topology

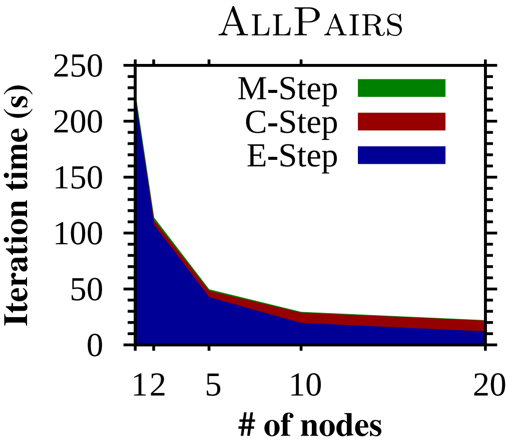


- No need for separate Reduce nodes
- By choosing connectivity, can trade off
  - bandwidth
  - latency
  - locality
  - ...



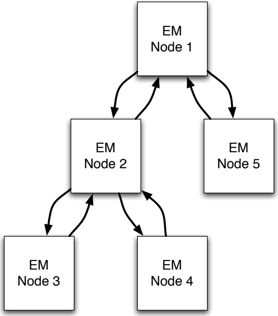
AllPairs topology





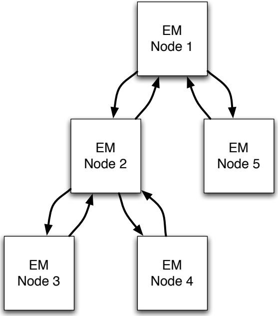
Total Bandwidth: 3.6 B counts per iteration

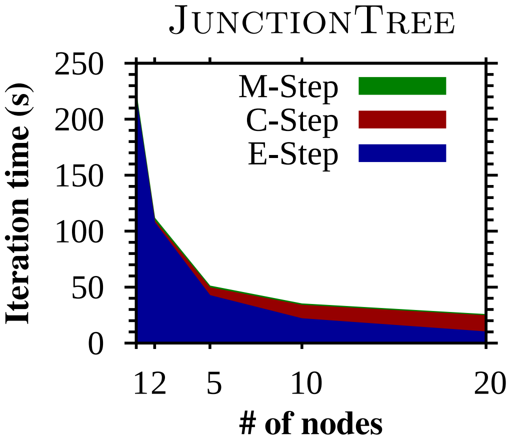
JunctionTree topology



- Nodes embedded in arbitrary tree structure
- Messages contain counts needed by nodes in both subtrees
- Tree can optimize for
  - bandwidth
  - locality
  - ...
- We use maximum spanning tree to heuristically minimize bandwidth
- Future work: multiple trees

JUNCTIONTREE topology





Total Bandwidth: 1.4 B counts per iteration

Locality in other models

- Ex: Latent Dirichlet Allocation (LDA) for topic modeling
  - Parameters: unigram distributions for each topic $p(w|t)$
  - Topic-word parameters local
  - Similar augmentation trick to Model 1
  - Details and results in paper
- Also applies to other EM models, beyond EM
  - Word locality is extremely common in NLP applications
  - Variational inference
  - Other computations that make sparse use of expectations

Conclusion

- A fully distributed, maximally localized EM algorithm
  - exploits parameter locality for significant speedup
  - is general; just define η for each datum
  - is flexible with respect to communication topology
- Many further improvements possible
  - intelligent partitioning of data
  - running E- and C-steps in parallel
  - better topologies (e.g., multiple trees)
  - exploiting approximate sparsity/locality
  - ...