

CS11-747 Neural Networks for NLP  
Unsupervised and Semi-supervised  
Learning of Structure

Graham Neubig



**Carnegie Mellon University**

Language Technologies Institute

Site

<https://phontron.com/class/nn4nlp2019/>

# Supervised, Unsupervised, Semi-supervised

- Most models handled here are **supervised** learning
  - Model  $P(Y|X)$ , at training time given both
- Sometimes we are interested in **unsupervised** learning
  - Model  $P(Y|X)$ , at training time given only  $X$
- Or **semi-supervised** learning
  - Model  $P(Y|X)$ , at training time given both or only  $X$

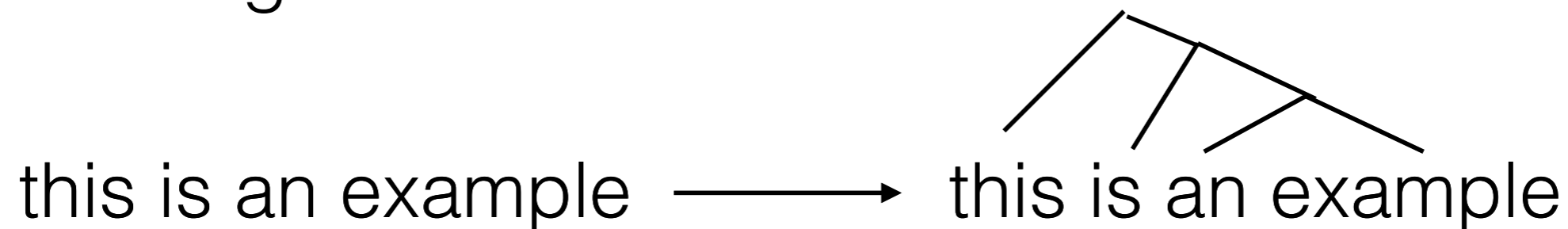
# Learning Features vs. Learning Structure

# Learning Features vs. Learning Discrete Structure

- Learning features, e.g. word/sentence embeddings:



- Learning discrete structure:



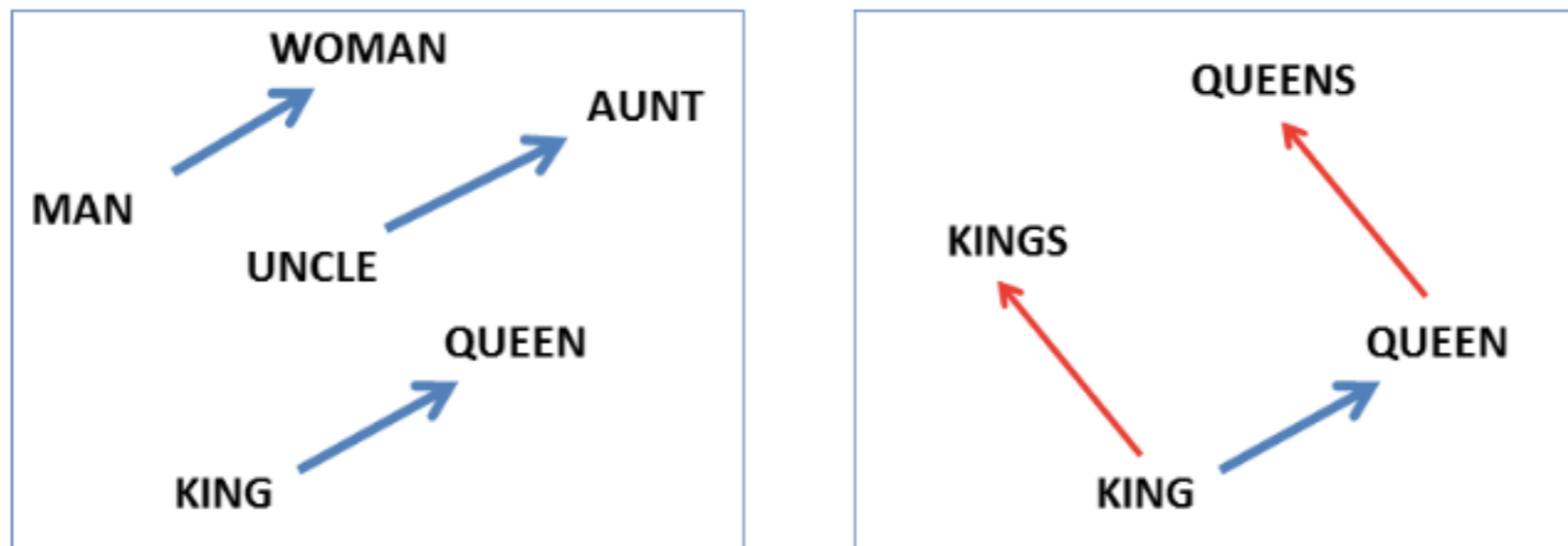
- Why discrete structure?
  - We may want to model information flow differently
  - More interpretable than features?

# Unsupervised Feature Learning (Review)

- When learning embeddings, we have an objective and use the intermediate states of this objective
  - CBOW
  - Skip-gram
  - Sentence-level auto-encoder
  - Skip-thought vectors
  - Variational auto-encoder

# How do we Use Learned Features?

- To solve tasks directly (Mikolov et al. 2013)



- And by proxy, knowledge base completion, etc., to be covered in a few classes
- To initialize downstream models

# What About Discrete Structure?

- We can cluster words
- We can cluster words in context (POS/NER)
- We can learn structure

# What is our Objective?

- Basically, a generative model of the data  $X$
- Sometimes factorized  $P(X|Y)P(Y)$ , a traditional generative model
- Sometimes factorized  $P(X|Y)P(Y|X)$ , an auto-encoder
  - This can be made mathematically correct through variational autoencoder  $P(X|Y)Q(Y|X)$



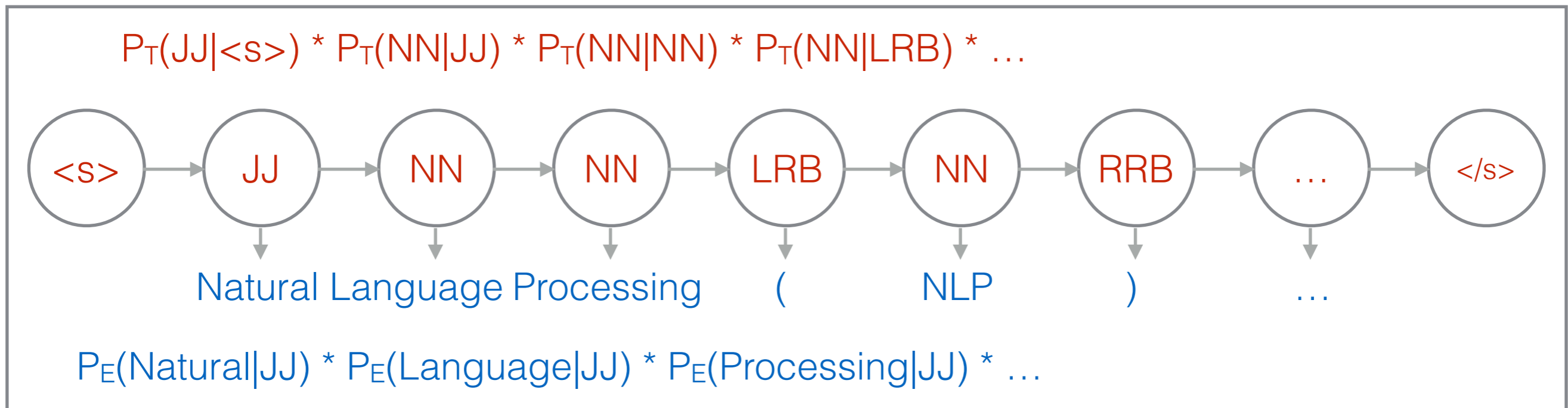
# Clustering Words in Context

# A Simple First Attempt

- Train word embeddings
- Perform k-means clustering on them
- Implemented in word2vec (-classes option)
- But what if we want single words to appear in different classes (same surface form, different values)

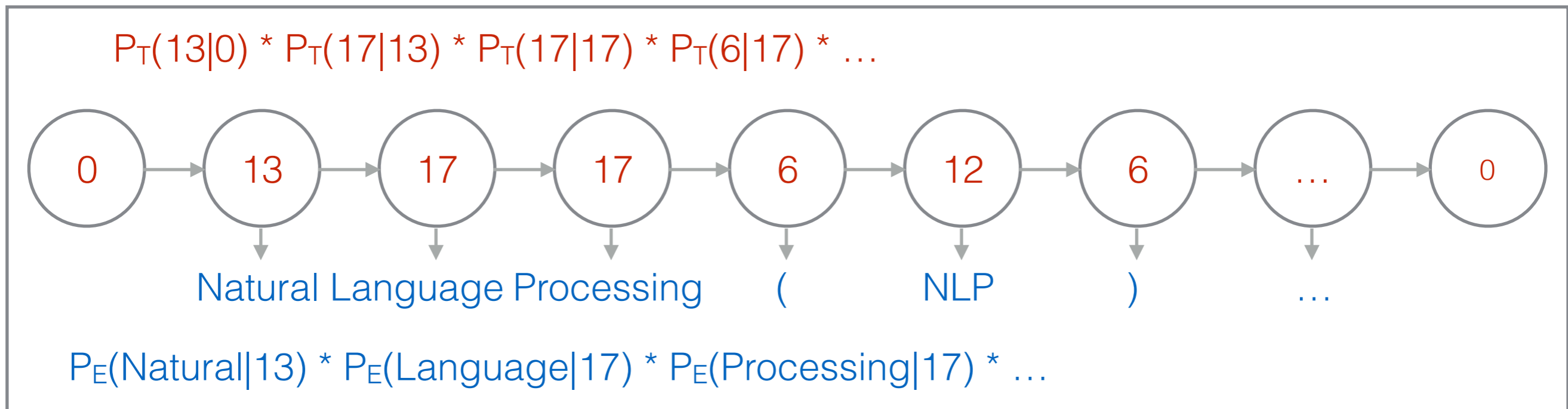
# Hidden Markov Models

- Factored model of  $P(X|Y)P(Y)$
- State  $\rightarrow$  state **transition** probabilities
- State  $\rightarrow$  word **emission** probabilities



# Unsupervised Hidden Markov Models

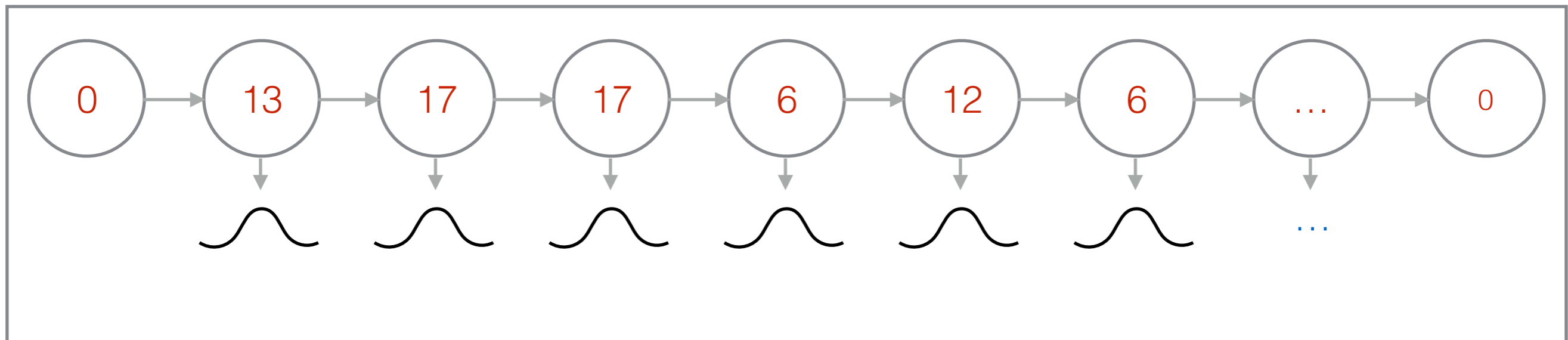
- Change label states to unlabeled numbers



- Can be trained with forward-backward algorithm

# Hidden Markov Models w/ Gaussian Emissions

- Instead of parameterizing each state with a categorical distribution, we can use a Gaussian (or Gaussian mixture)!



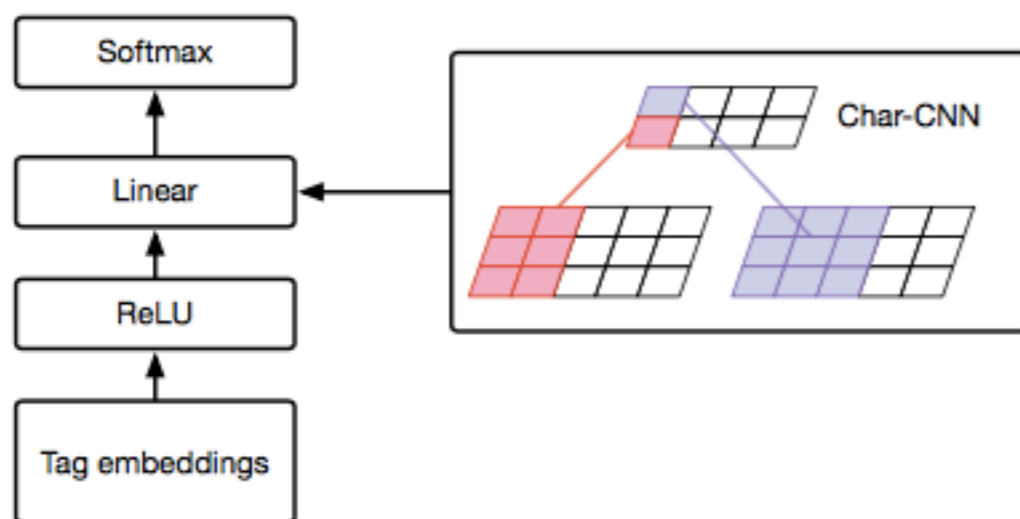
- Long the defacto-standard for speech
- Applied to POS tagging by training to emit word embeddings by Lin et al. (2015)

# A Simple Approximation: State Clustering (Giles et al. 1992)

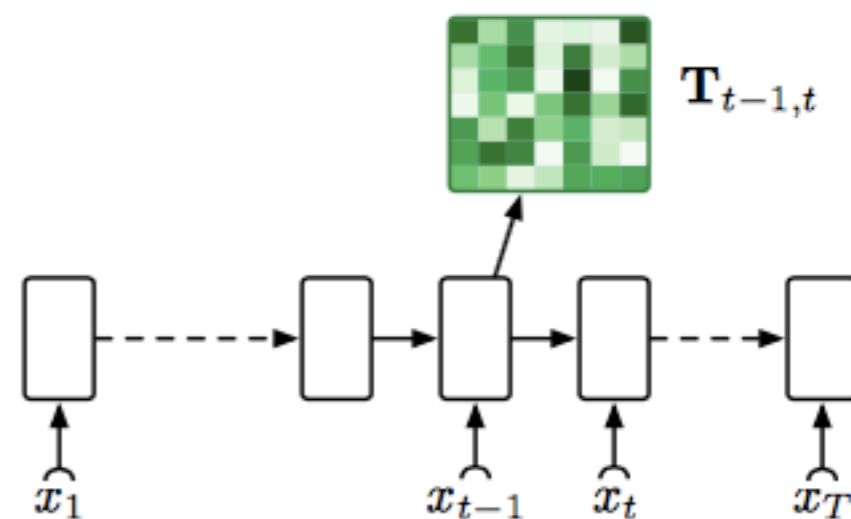
- Simply train an RNN according to a standard loss function (e.g. language model)
- Then cluster the hidden states according to k-means, etc.

# Featurized Hidden Markov Models (Tran et al. 2016)

- Calculate the transition/emission probabilities with neural networks!
  - **Emission:** Calculate representation of each word in vocabulary w/ CNN, dot product with tag representation and softmax to calculate emission prob
  - **Transition Matrix:** Calculate w/ LSTMs (breaks Markov assumption)



**Figure 2:** Computational graph of Char-CNN emission network. A character convolutional neural network is used to compute the weight of the linear layer for every minibatch.

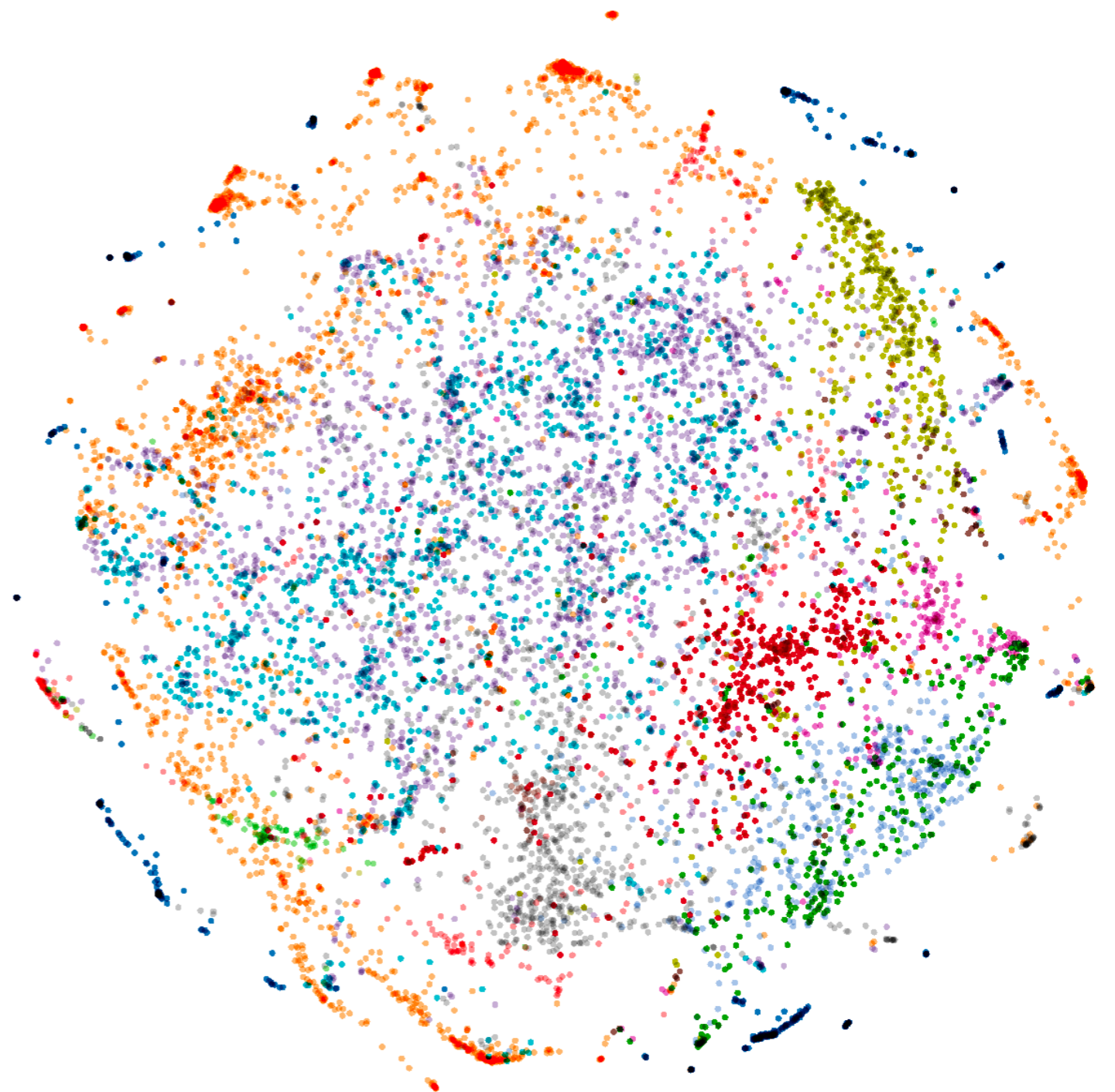


**Figure 3:** A graphical representation of our LSTM transition network. Transition matrix  $\mathbf{T}_{t-1,t}$  from time step  $t-1$  to  $t$  is computed based on the hidden state of the LSTM at time  $t-1$ .

# Problem: Embeddings May Not be Indicative of Syntax

(He et al. 2018)

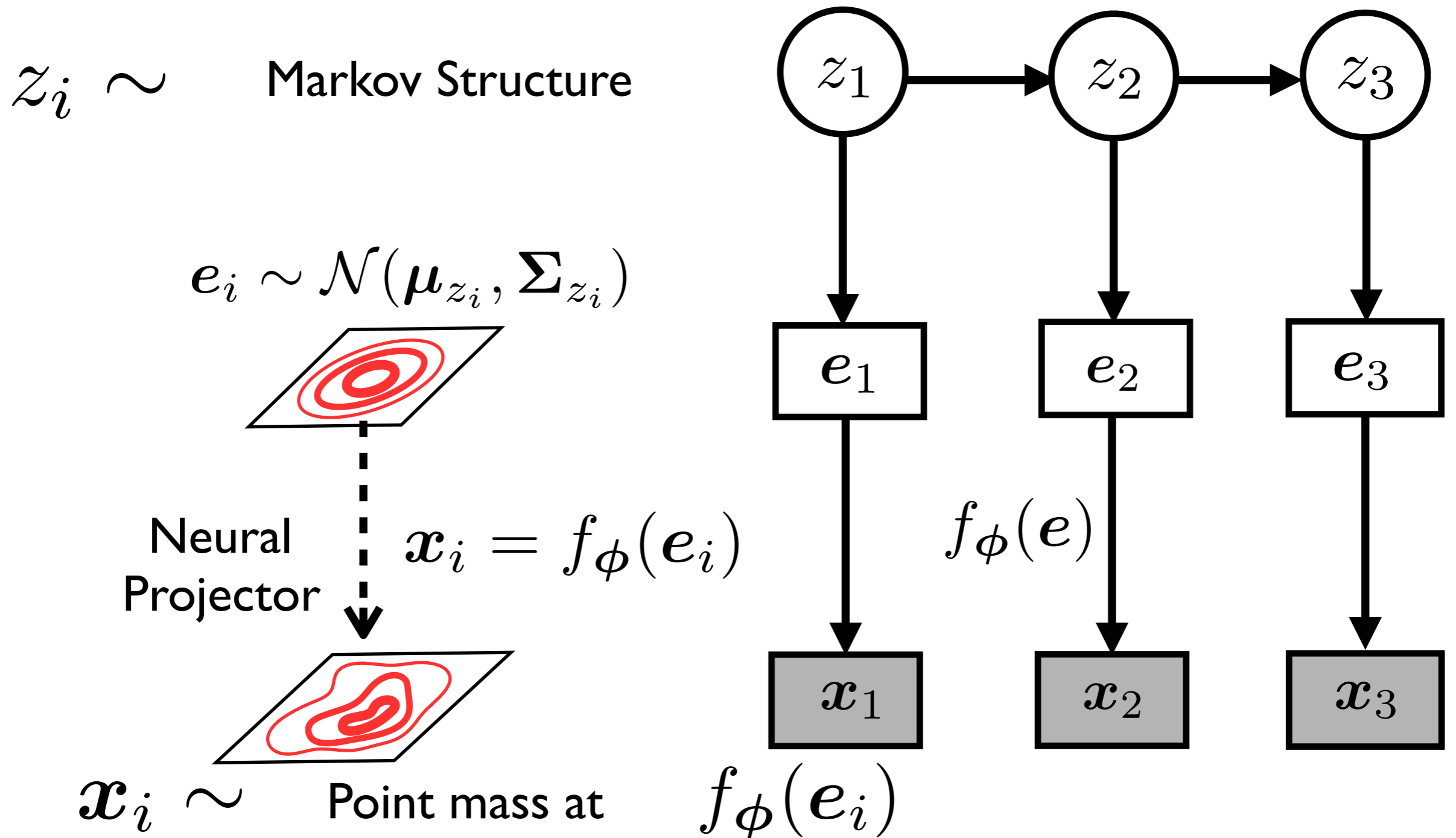
- adjective
- adverb
- noun singular
- noun proper
- noun plural
- verb base
- verb gerund
- verb past tense
- verb past participle
- verb 3rd singular
- cardinal number





# Learning POS Taggers w/ Latent Embeddings

(He et al. 2018)

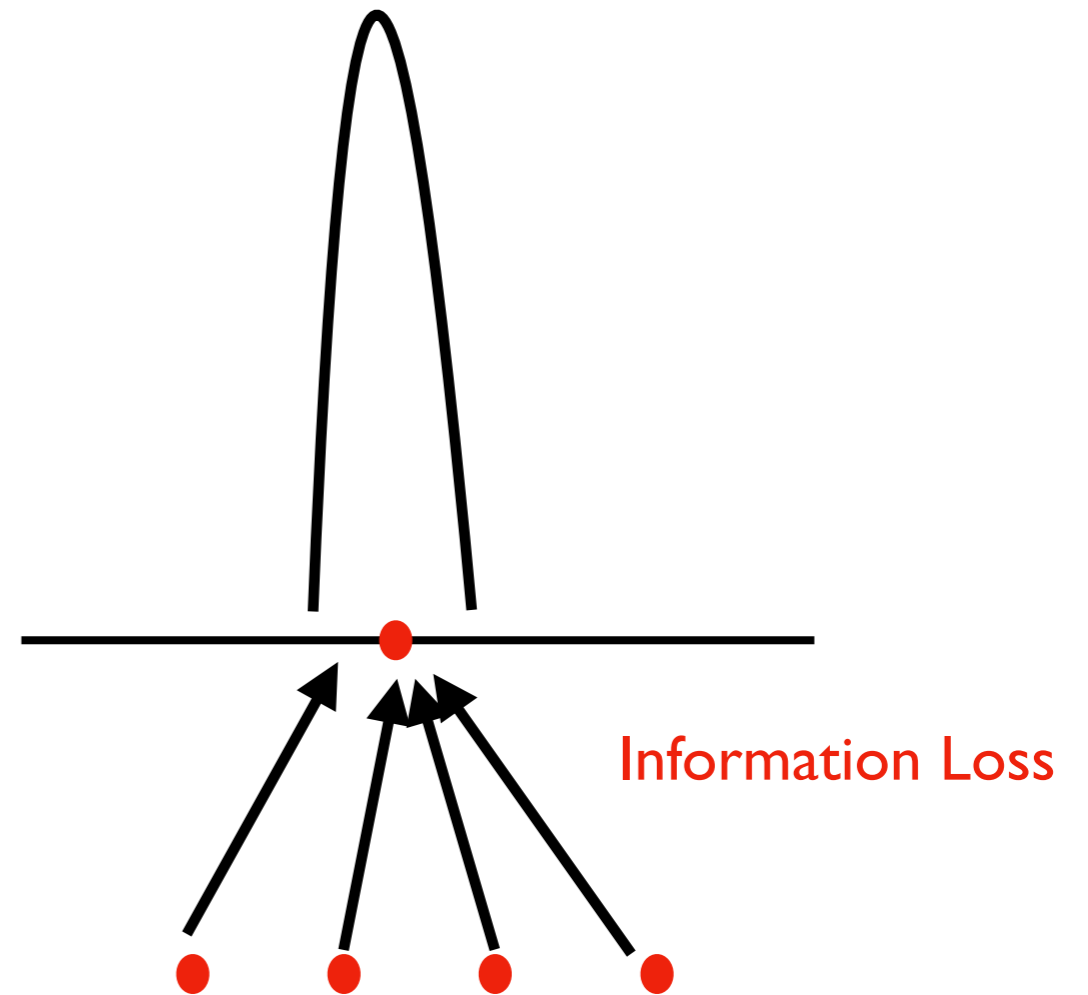
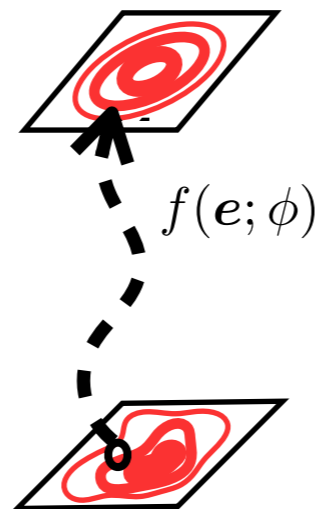


# A Simpler Method: Map Directly to Space of Prior

Example of Markov prior

$$\max \log p_{\text{GHMM}}(f_{\phi}(\mathbf{x}))$$

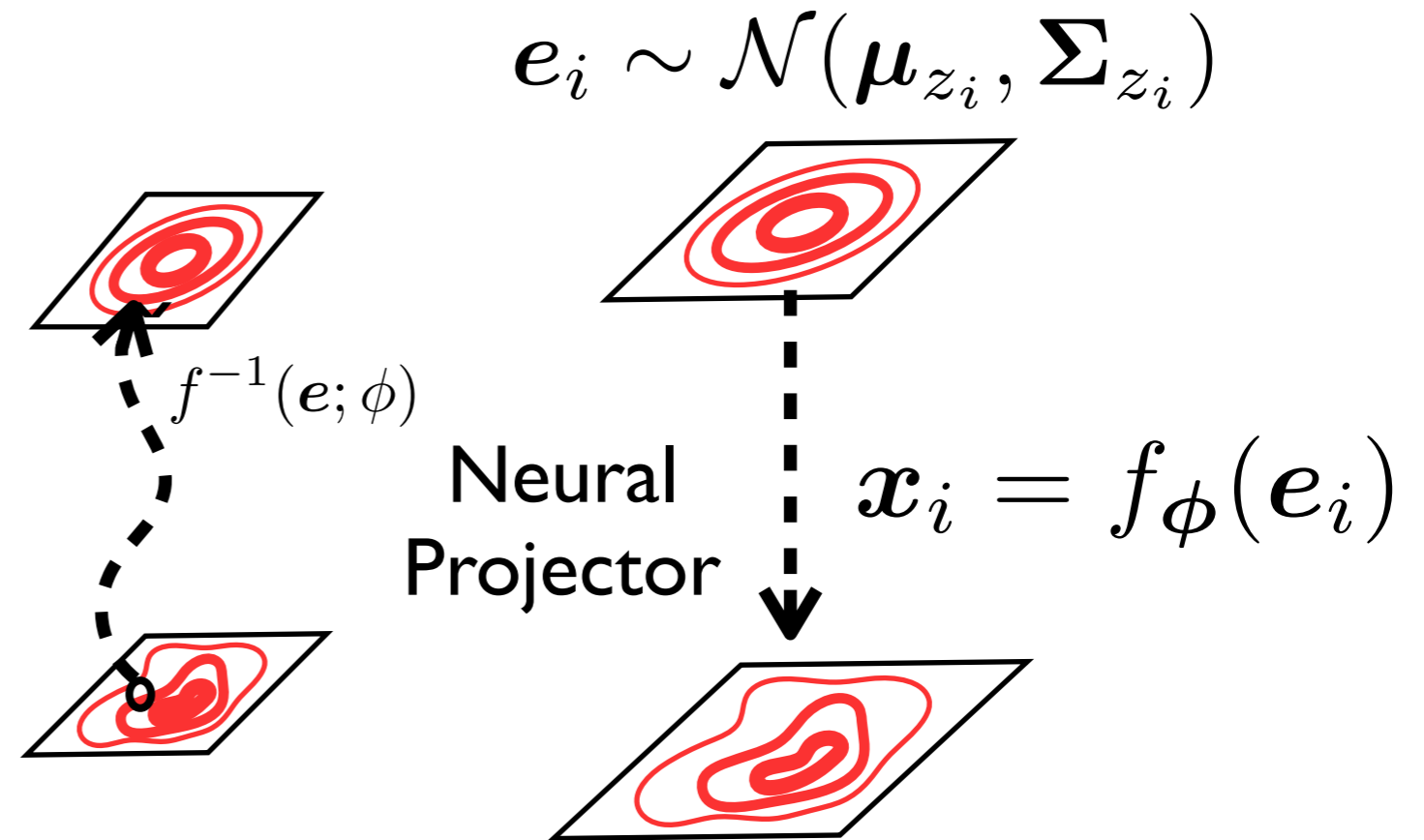
?



# Normalizing Flow

(Rezende and Mohamed 2015)

- Basic idea: a way to map from one probability distribution to another in an invertible manner



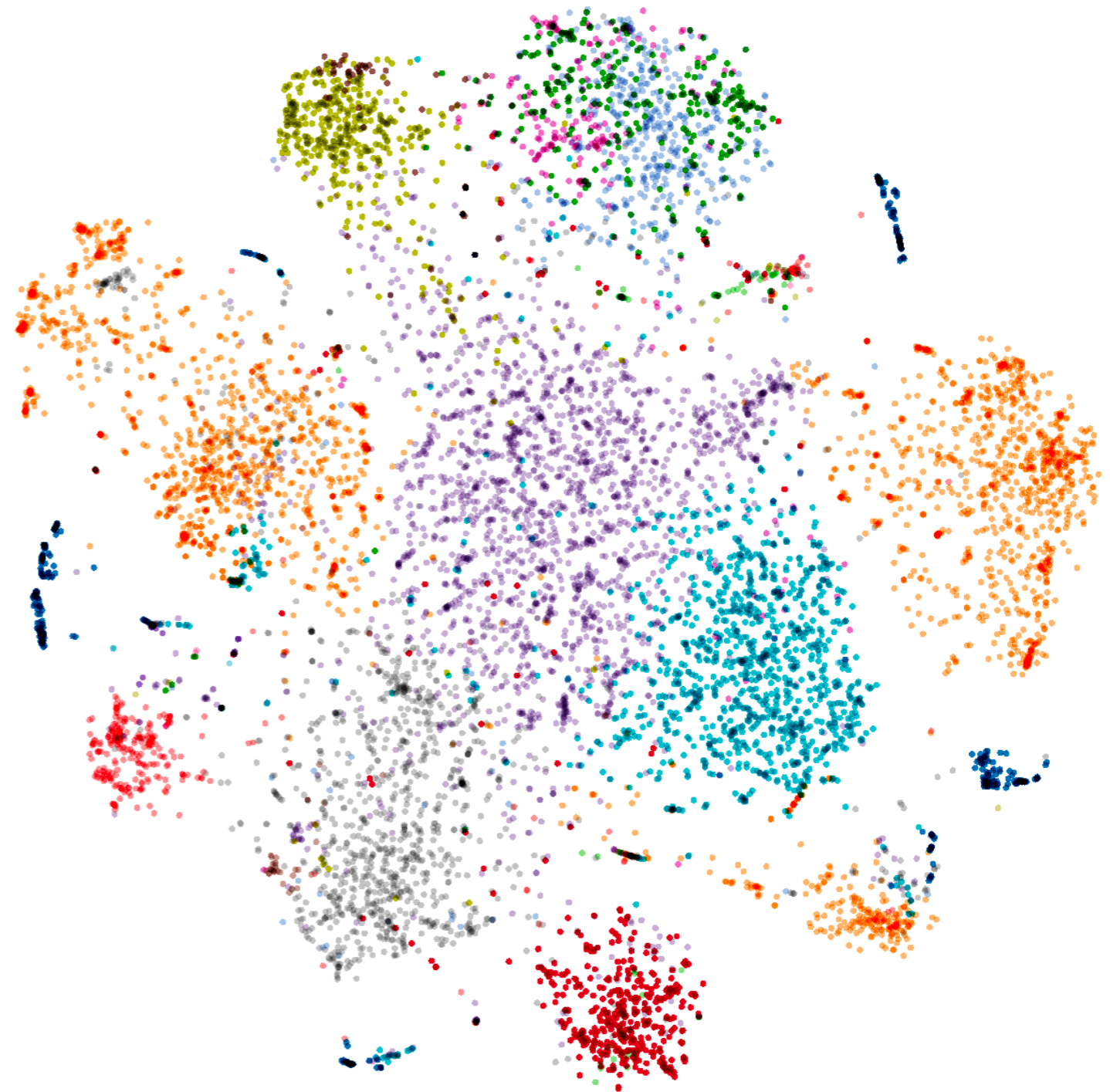
- Advantage: optimizing probability can be shown to be possible by following equation

$$\log p(\mathbf{x}) = \log p_{\text{GHMM}}(f_\phi^{-1}(\mathbf{x})) + \sum \log \left| \det \frac{\partial f_\phi^{-1}}{\partial \mathbf{x}_i} \right|$$

$-\infty$  when  $f$  is not invertible

# Results of Learning w/ HMM

(He et al. 2018)

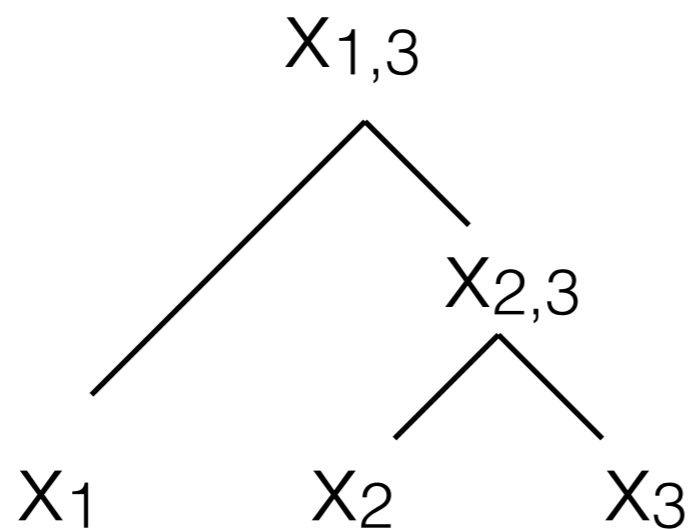


# Unsupervised Phrase-structured Composition Functions

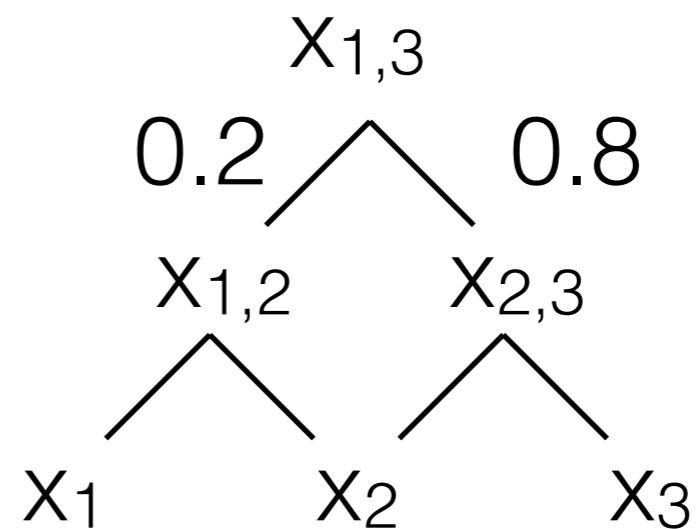
# Soft vs. Hard Tree Structure

- Soft tree structure: use a differentiable gating function
- Hard tree structure: non-differentiable, but allows for more complicated composition methods

Hard



Soft



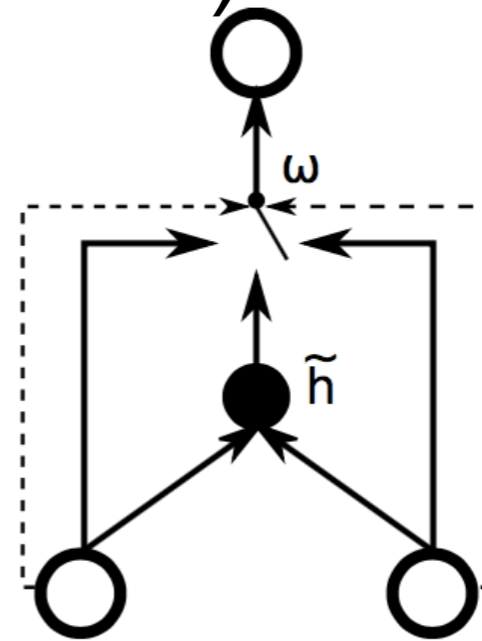
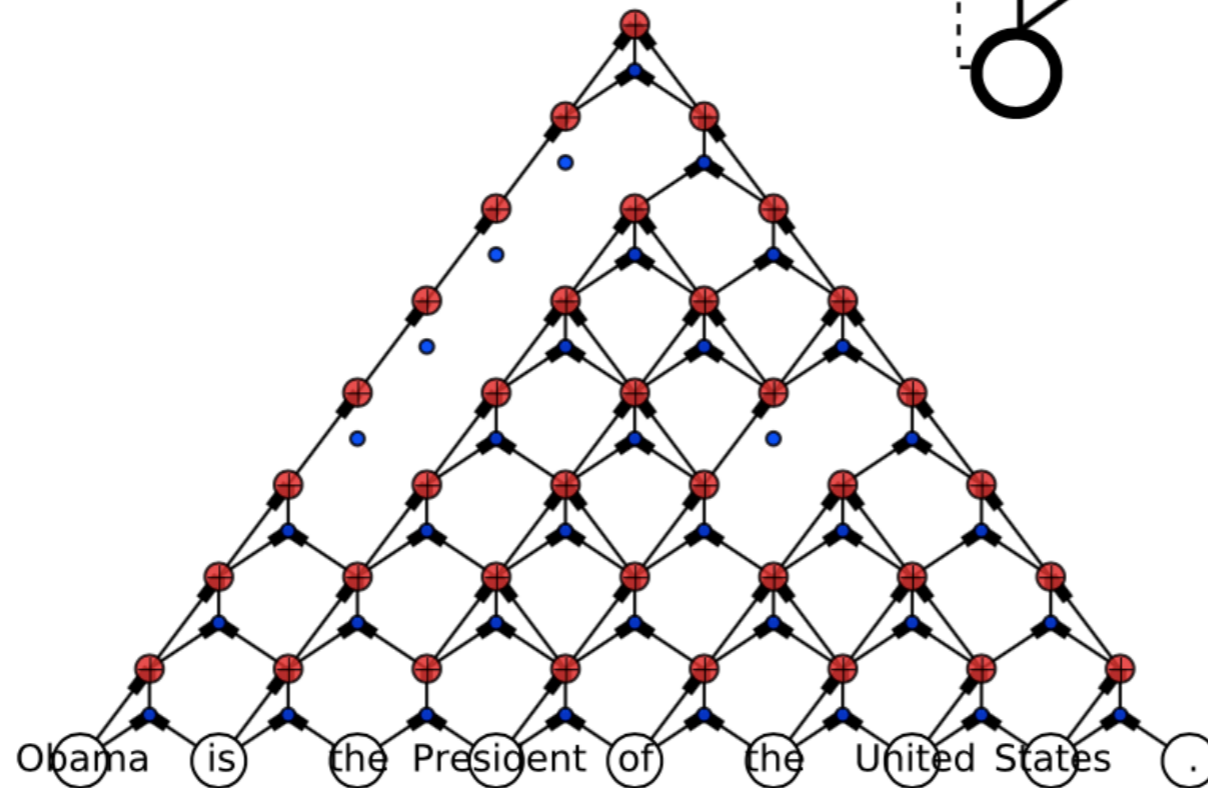
# One Other Paradigm: Weak Supervision

- **Supervised:** given  $X, Y$  to model  $P(Y|X)$
- **Unsupervised:** given  $X$  to model  $P(Y|X)$
- **Weakly Supervised:** given  $X$  and  $V$  to model  $P(Y|X)$ , under assumption that  $Y$  and  $V$  are correlated
  - Note: different from multi-task or transfer learning because we are given no  $Y$
  - Note: different from supervised learning with latent variables, because we care about  $Y$ , not  $V$

# Gated Convolution

(Cho et al. 2014)

- Can choose whether to use left node, right node, or combination of both



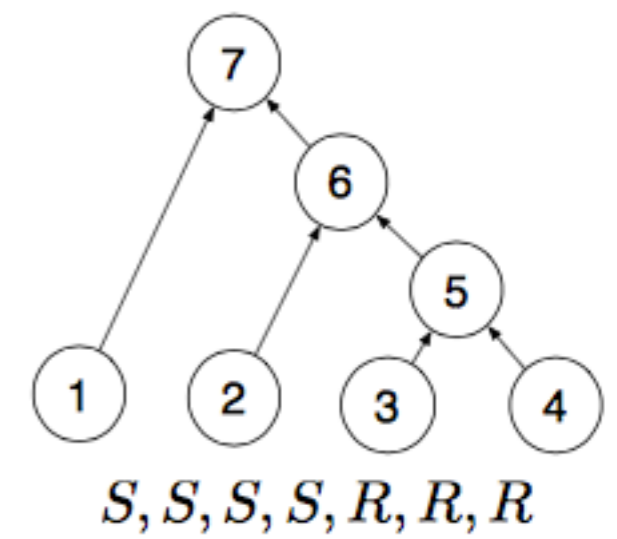
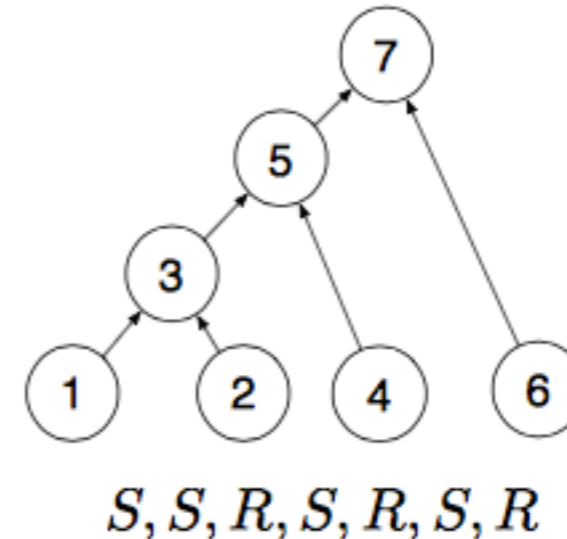
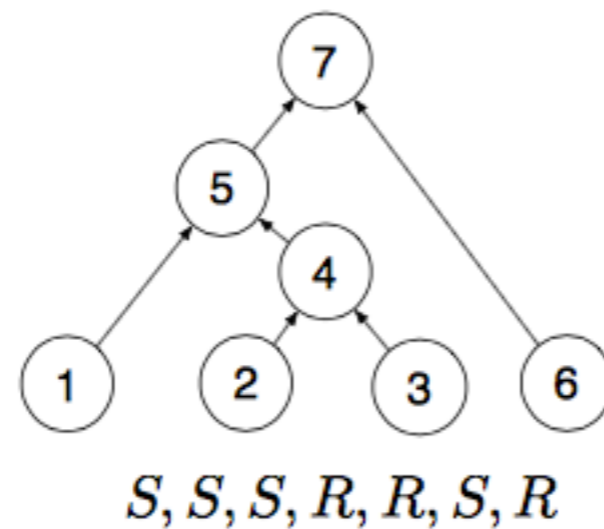
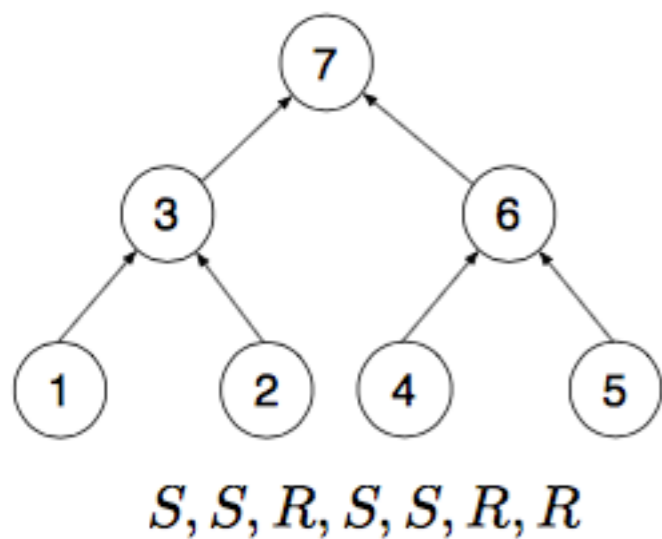
- Trained using MT loss



# Learning with RL

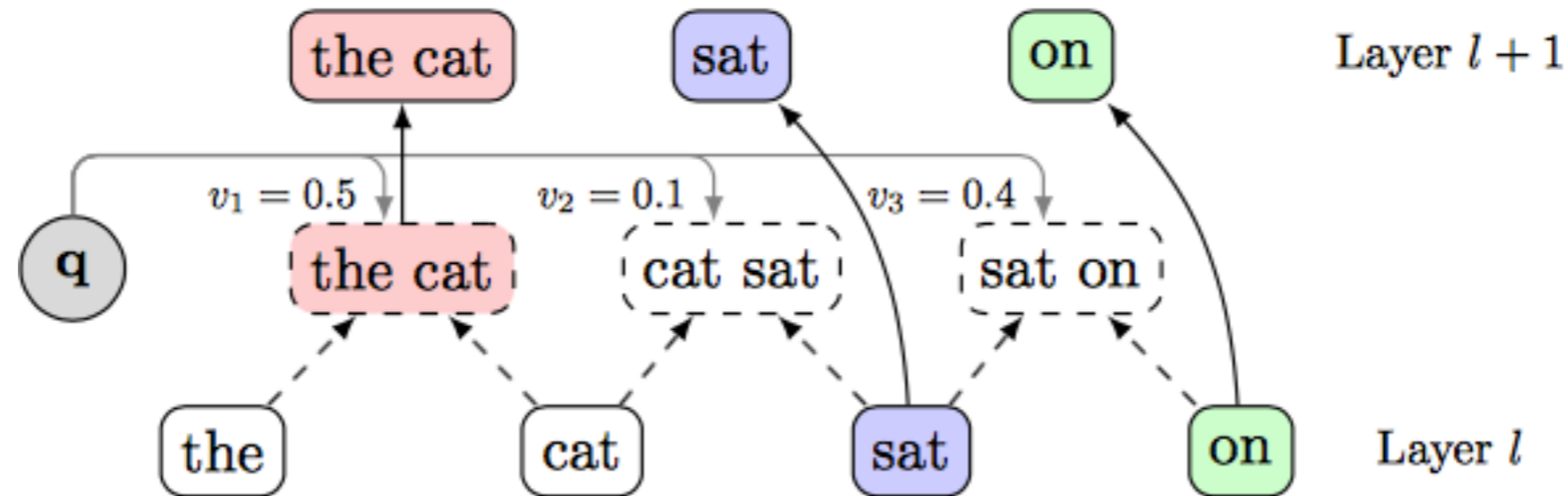
(Yogatama et al. 2016)

- Intermediate tree-structured representation for language modeling
- Predict that tree using shift-reduce parsing, sentence representation composed in tree-structured manner
- Reinforcement learning with supervised loss, prediction loss



# Learning w/ Layer-wise Reductions (Choi et al. 2017)

- Choose one parent at each layer, reducing size by one

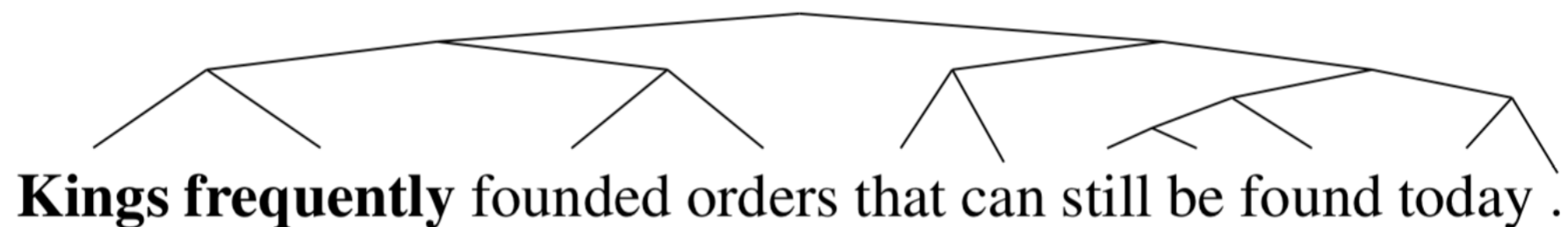
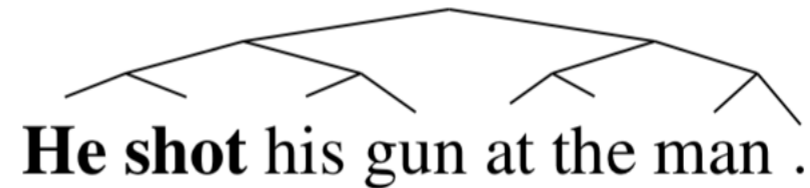


- Train using Gumbel-straightthrough reparameterization trick
- Faster and more effective than RL?
- Williams et al. (2017) find that this gives less trivial trees as well

# Difficulties in Learning Latent Structure

(Williams et al. 2018)

- Unfortunately, many models learn trivial structure...
- e.g. balanced binary, left/right branching



- Why? One explanation: tension between (untrained) parser, and (untrained) downstream task model

# Learning Dependencies

# Phrase Structure vs. Dependency Structure

- Previous methods attempt to learn representations of phrases in tree-structured manner
- We might also want to learn dependencies, that tell which words depend on others

# Dependency Model w/ Valence (Klein and Manning 2004)

- Basic idea: top-down dependency based language model that generates left and right sides, then stops



- For both the right and left side, calculate whether to continue generating words, and if yes generate

e.g., a slightly simplified view for word “saw”

$P_d(\langle \text{cont} \rangle \mid \text{saw}, \leftarrow, \text{false}) * P_w(I \mid \text{saw}, \leftarrow, \text{false}) *$

$P_d(\langle \text{stop} \rangle \mid \text{saw}, \leftarrow, \text{true}) *$

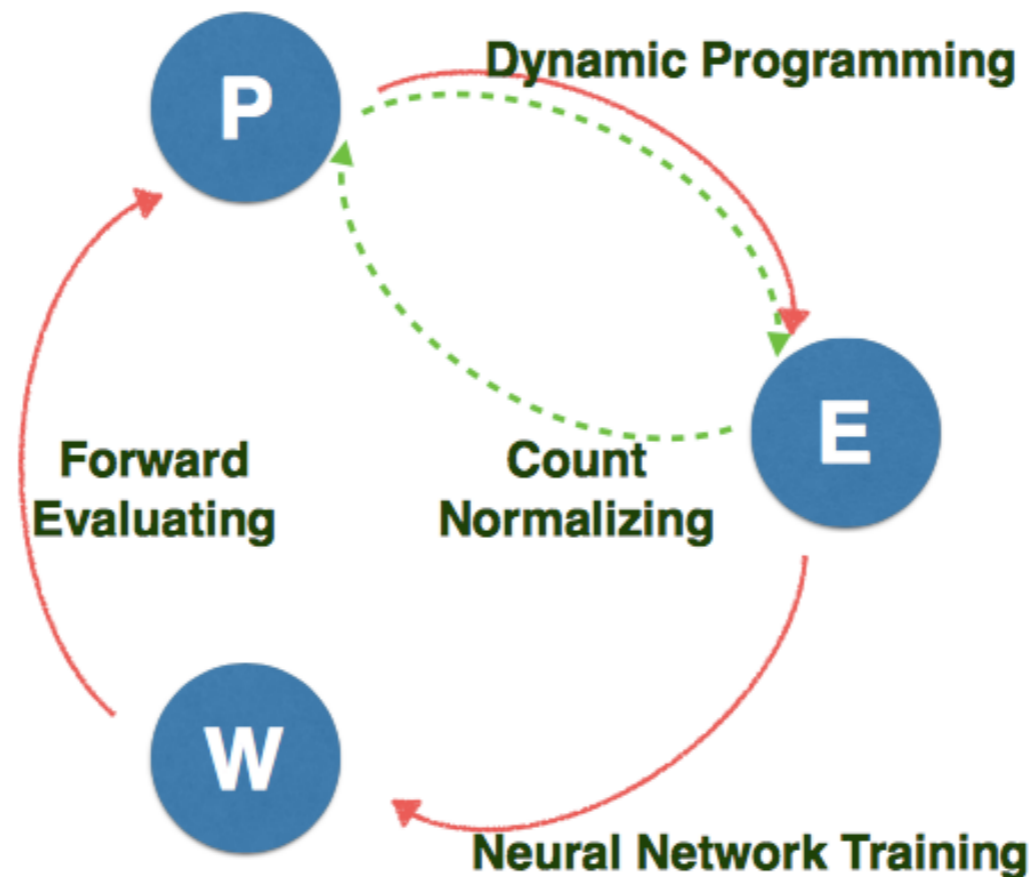
$P_d(\langle \text{cont} \rangle \mid \text{saw}, \rightarrow, \text{false}) * P_w(\text{girl} \mid \text{saw}, \leftarrow, \text{false}) *$

$P_d(\langle \text{cont} \rangle \mid \text{saw}, \rightarrow, \text{true}) * P_w(\text{with} \mid \text{saw}, \leftarrow, \text{true}) *$

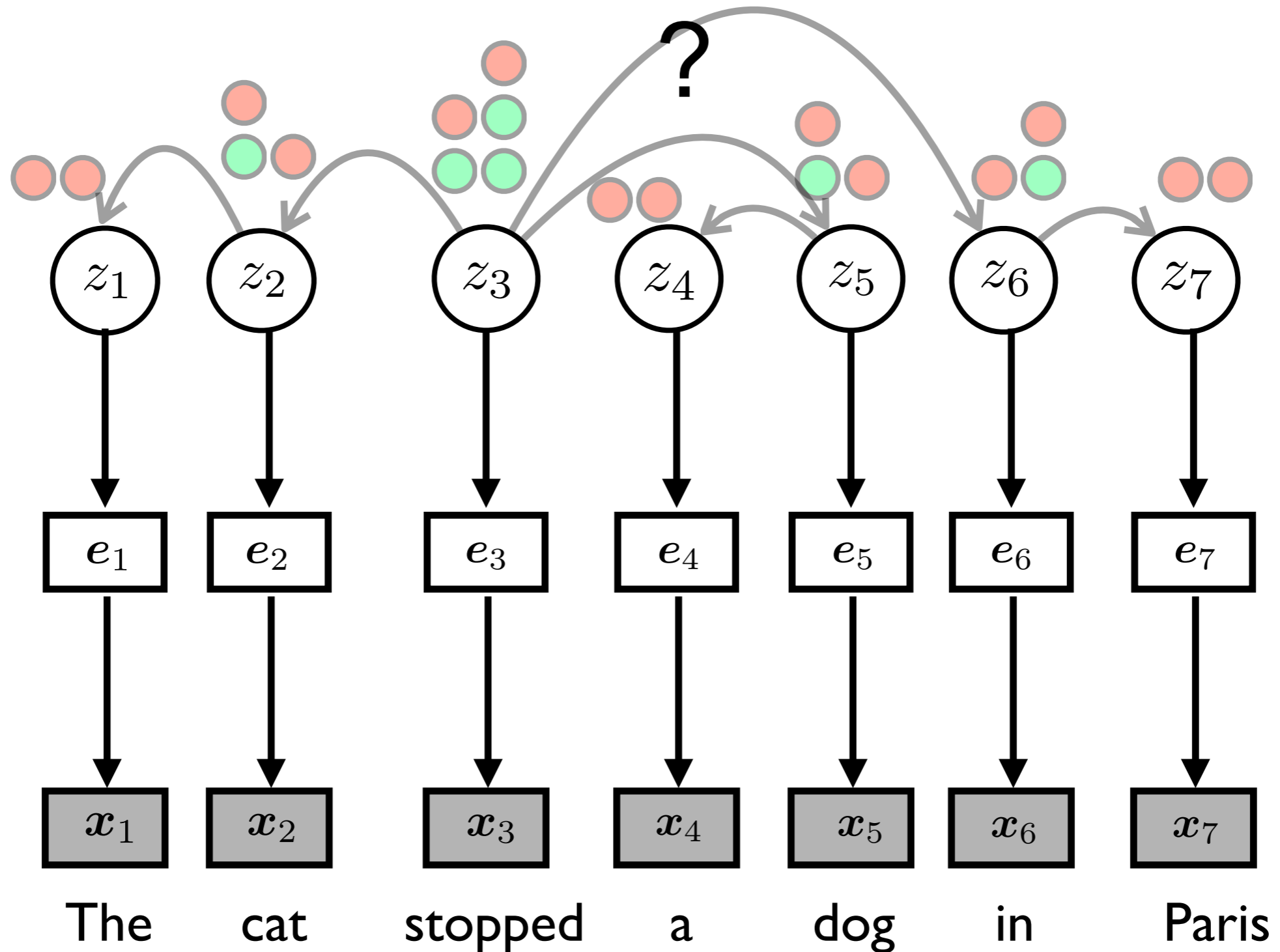
$P_d(\langle \text{stop} \rangle \mid \text{saw}, \leftarrow, \text{true})$

# Unsupervised Dependency Induction w/ Neural Nets (Jiang et al. 2016)

- Simple: parameterize the decision with neural nets instead of with count-based distributions
- Like DMV, train with EM algorithm



# Invertible Projections for DMV (He et al. 2018)





# Learning Dependency Heads

## w/ Attention (Kuncoro et al. 2017)

- Given a phrase structure tree, what child is the head word, the most important word in the phrase?
- Idea: create a phrase composition function that uses attention: examine if attention weights follow heads defined by linguistics

### Noun phrases

Canadian (0.09) <b>Auto (0.31)</b> Workers (0.2) union (0.22) president (0.18) no (0.29) major (0.05) <b>Eurobond (0.32)</b> or (0.01) foreign (0.01) bond (0.1) offerings (0.22) Saatchi (0.12) client (0.14) Philips (0.21) Lighting (0.24) <b>Co. (0.29)</b> nonperforming (0.18) commercial (0.23) <b>real (0.25)</b> estate (0.1) <b>assets (0.25)</b> the (0.1) Jamaica (0.1) Tourist (0.03) Board (0.17) ad (0.20) <b>account (0.40)</b>
the (0.0) final (0.18) <b>hour (0.81)</b> their (0.0) first (0.23) <b>test (0.77)</b> <b>Apple (0.62)</b> , (0.02) Compaq (0.1) and (0.01) IBM (0.25) both (0.02) stocks (0.03) and (0.06) <b>futures (0.88)</b> NP (0.01) , (0.0) <b>and (0.98)</b> NP (0.01)

### Verb phrases

<b>buying (0.31)</b> and (0.25) selling (0.21) NP (0.23) ADVP (0.27) <b>show (0.29)</b> PRT (0.23) PP (0.21) <b>pleaded (0.48)</b> ADJP (0.23) PP (0.15) PP (0.08) PP (0.06) <b>received (0.33)</b> PP (0.18) NP (0.32) PP (0.17) cut (0.27) <b>NP (0.37)</b> PP (0.22) PP (0.14)
<b>to (0.99)</b> VP (0.01) <b>were (0.77)</b> n't (0.22) VP (0.01) did (0.39) <b>n't (0.60)</b> VP (0.01) handle (0.09) <b>NP (0.91)</b> VP (0.15) <b>and (0.83)</b> VP 0.02)

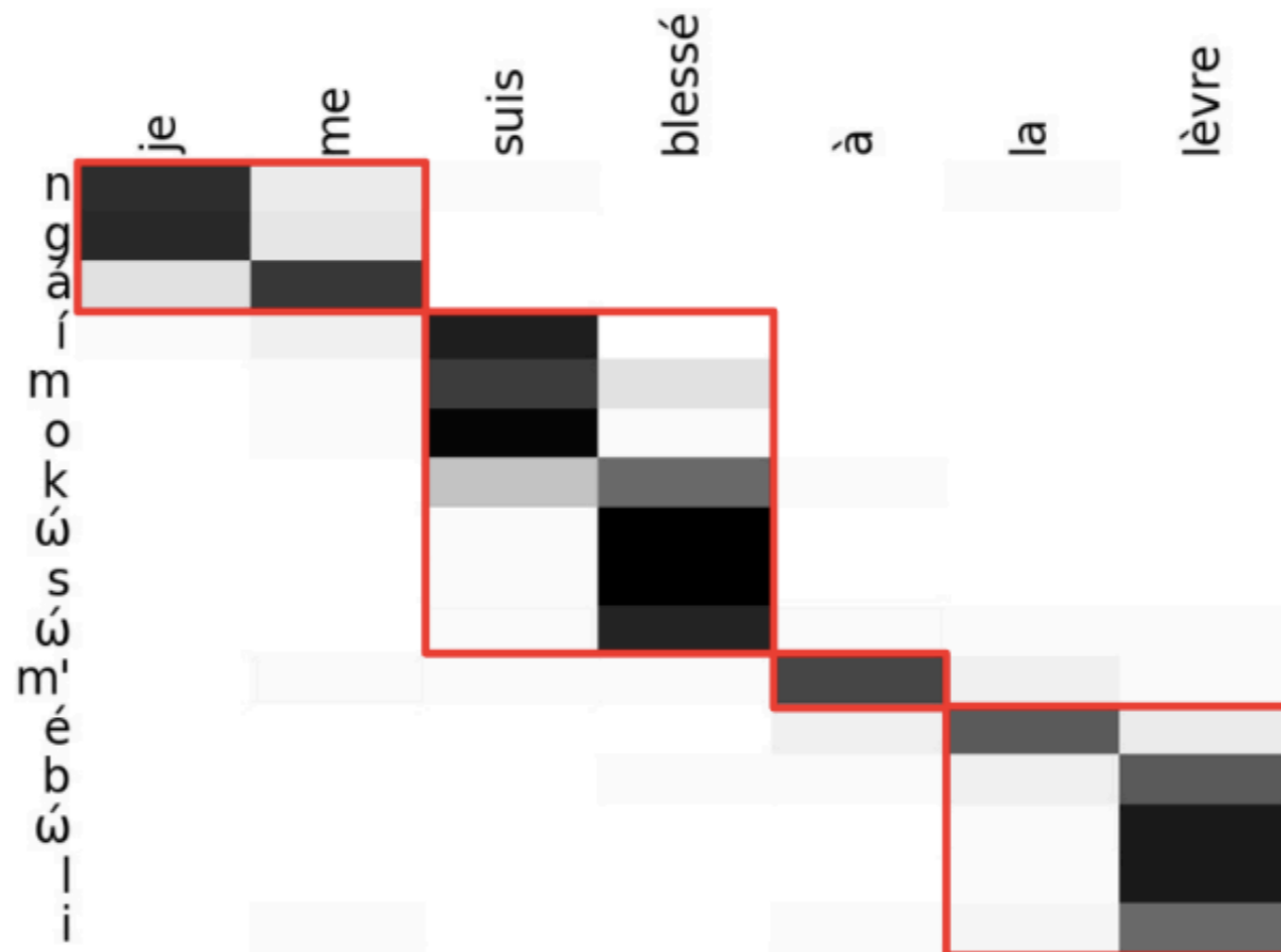
### Prepositional phrases

ADVP (0.14) <b>on (0.72)</b> NP (0.14) ADVP (0.05) <b>for (0.54)</b> NP (0.40) ADVP (0.02) <b>because (0.73)</b> of (0.18) NP (0.07) such (0.31) <b>as (0.65)</b> NP (0.04) from (0.39) <b>NP (0.49)</b> PP (0.12)
<b>of (0.97)</b> NP (0.03) <b>in (0.93)</b> NP (0.07) <b>by (0.96)</b> S (0.04) <b>at (0.99)</b> NP (0.01) NP (0.1) <b>after (0.83)</b> NP (0.06)

# Other Examples

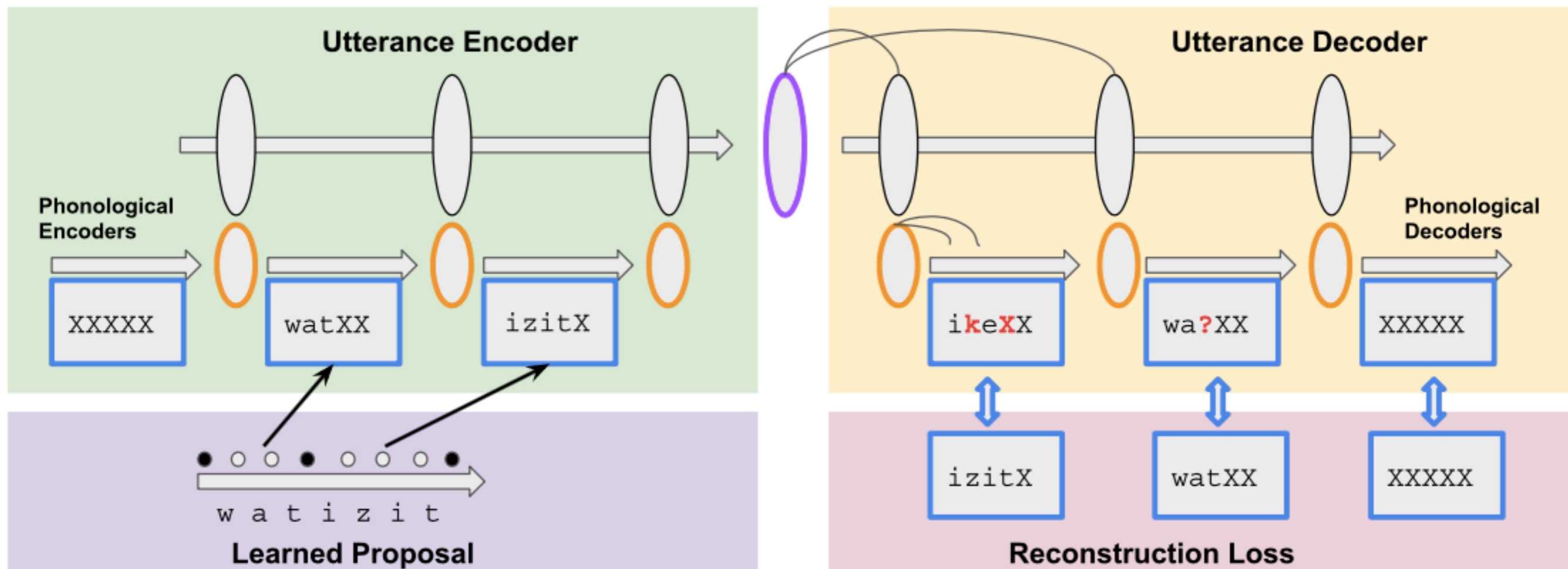
# Learning about Word Segmentation from Attention (Boito et al. 2017)

- We want to learn word segmentation in an unsegmented language
- Simple idea: we can inspect the attention matrices from a neural MT system to extract words



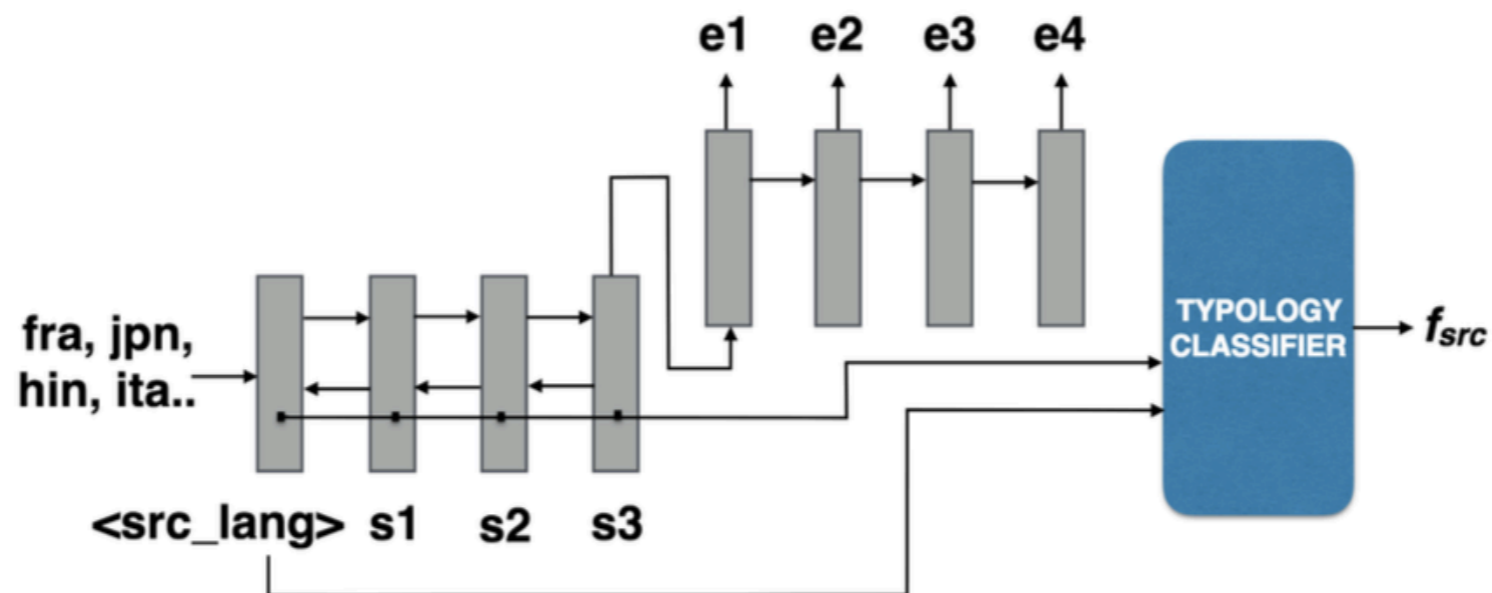
# Learning Segmentations w/ Reconstruction Loss (Elsner and Shain 2017)

- Learn segmentations of speech/text that allow for easy reconstruction of the original
- Idea: consistent segmentation should result in easier-to-reconstruct segments
- Train segmentation using policy gradient



# Learning Language-level Features (Malaviya et al. 2017)

- All previous work learned features of a single sentence
- Can we learn *features of the whole language*? e.g. Typology: what is the canonical word order, etc.
- A simple method: train a neural MT system on 1017 languages, and extract its representations



Questions?