TTIC 31210:
Advanced Natural Language Processing

Kevin Gimpel

Spring 2019

Lecture 14:
Inference in Bayesian NLP

Roadmap

- intro (1 lecture)
- deep learning for NLP (5 lectures)
- structured prediction (4.5 lectures)
- generative models, latent variables, unsupervised learning, variational autoencoders (1.5 lectures)
- **Bayesian methods in NLP (2 lectures)**
- Bayesian nonparametrics in NLP (1.5 lectures)
- research tips & other topics (0.5 lectures)

Assignments

- we’ll go over Assignment 3 today

- Assignment 4 has been posted; due in 2 weeks

Motivation

- in neural NLP, we typically assume parameters and architectures are fixed
- 1-layer MLP:

$$p(Y = y \mid \boldsymbol{x}) = \frac{\exp\{\mathbf{w}_y^\top \tanh\left(\mathbf{W} g(\boldsymbol{x})\right)\}}{Z}$$

- now, include parameters as random variables and condition on them:

$$p(Y = y \mid \boldsymbol{x}, \Theta = \{\mathbf{w}, \mathbf{W}\}) = \frac{\exp\{\mathbf{w}_y^\top \tanh\left(\mathbf{W} g(\boldsymbol{x})\right)\}}{Z}$$

Motivation

$$p(Y = y \mid \boldsymbol{x}, \Theta = \{\mathbf{w}, \mathbf{W}\}) = \frac{\exp\{\mathbf{w}_y^\top \tanh\left(\mathbf{W} g(\boldsymbol{x})\right)\}}{Z}$$

- how do we get back to $p(Y = y \mid \boldsymbol{x})$?
- marginalize over new random variables:

$$p(Y = y \mid \boldsymbol{x}) = \int_{\Theta} p(Y = y, \Theta = \{\mathbf{w}, \mathbf{W}\} \mid \boldsymbol{x}) d\Theta$$

- intuitively: don't commit to a single set of parameter values; use them all (with a suitable prior distribution)

Going Further…

- marginalize over architectures & parameters?

$$p(Y = y \mid \boldsymbol{x}, \Lambda = \mathrm{MLP}(\mathbf{w}, \mathbf{W})) = \frac{\exp\{\mathbf{w}_y^\top \tanh\left(\mathbf{W} g(\boldsymbol{x})\right)\}}{Z}$$

$$p(Y = y \mid \boldsymbol{x}) = \int_\Lambda p(Y = y, \Lambda = \mathrm{MLP}(\mathbf{w}, \mathbf{W}) \mid \boldsymbol{x}) d\Lambda$$

Generative Story Template

1: Draw a set of parameters θ from $p(\Theta)$

2: Draw a latent structure z from $p(Z \mid \theta)$

3: Draw the observed data x from $p(X \mid z, \theta)$

the above generative story implies the following factorization of the joint distribution:

$$p(x, z, \theta) = p(\theta)p(z \mid \theta)p(x \mid z, \theta)$$

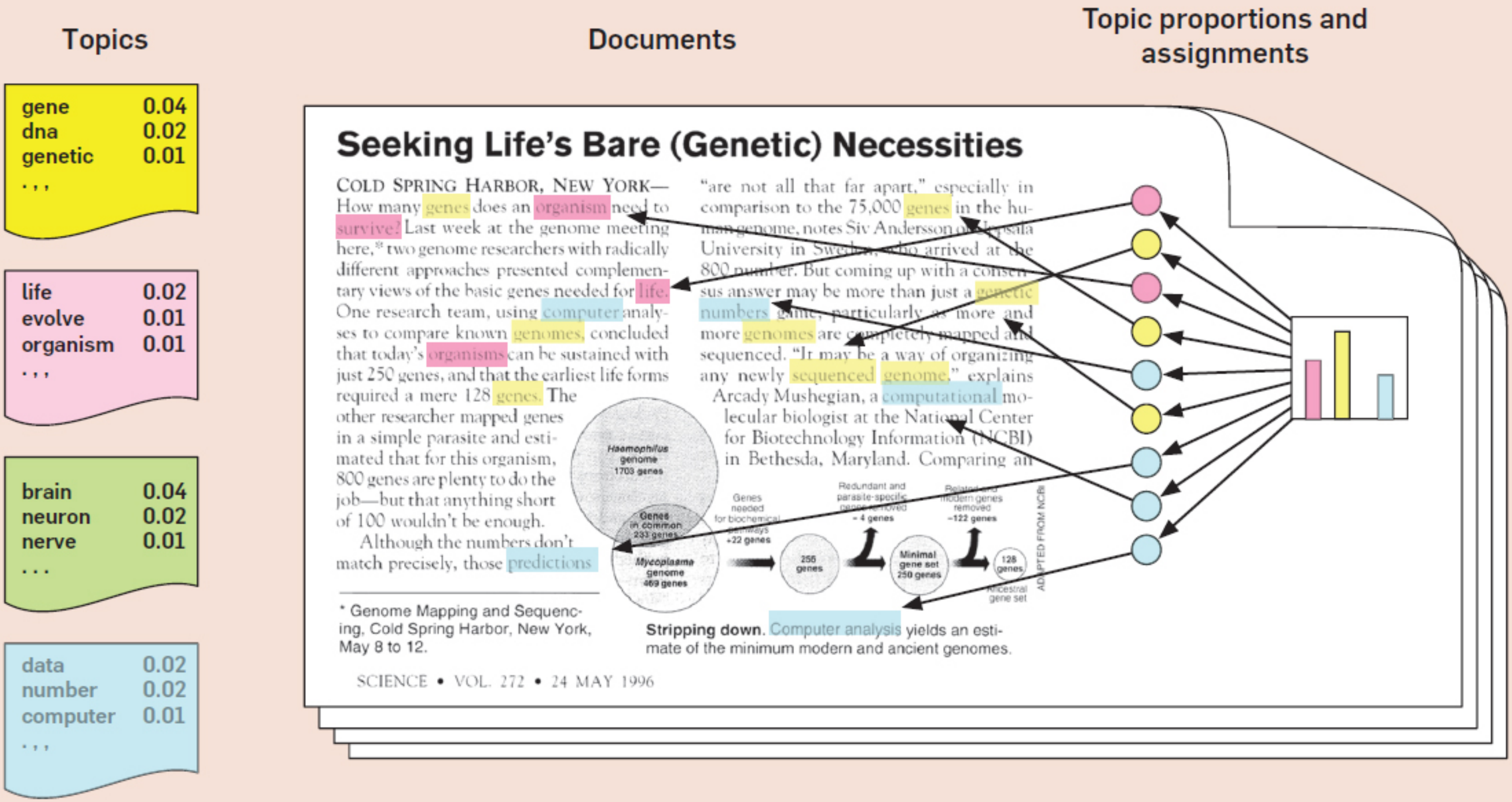
Latent Dirichlet Allocation

David M. Blei BLEI@CS.BERKELEY.EDU
Computer Science Division
University of California
Berkeley, CA 94720, USA

Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department
Stanford University
Stanford, CA 94305, USA

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
Computer Science Division and Department of Statistics
University of California
Berkeley, CA 94720, USA

- generative model for document collections using latent variables that can be interpreted as "topics"
- learns a multinomial distribution over words for each topic



Blei (2012): *Probabilistic Topic Models*

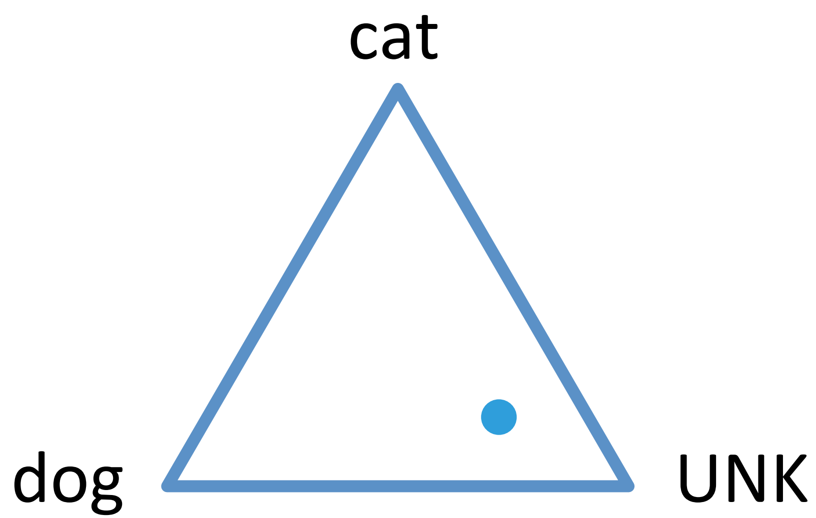
Dirichlet Distribution

- parameterized by a positive vector α

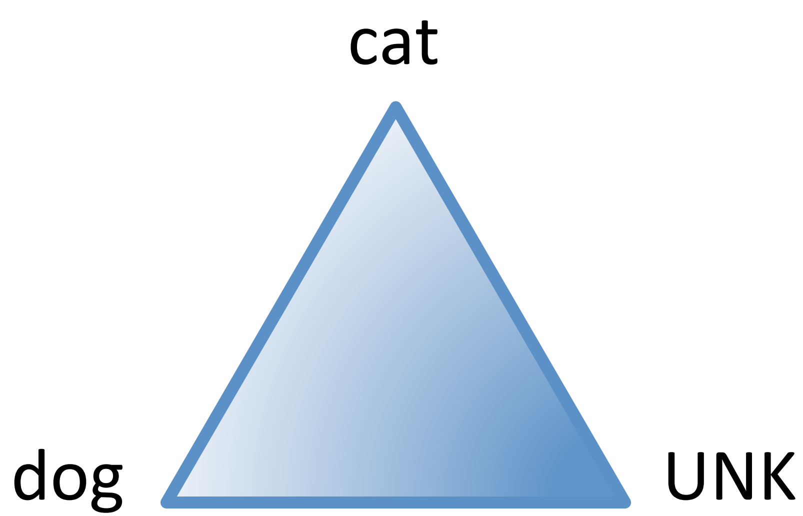
$$\theta \sim \mathrm{Dirichlet}(\alpha)$$

$$p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i}^{K} \theta_i^{\alpha_i - 1}$$

- categorical = point on the simplex



- Dirichlet = distribution over the simplex



Compare Categorical and Dirichlet

categorical:

$$x \sim \text{Categorical}(\theta)$$

$$p(X = x_i \mid \theta) = \theta_i \qquad i \in \{1, \ldots, K\}$$

vector form of categorical:

$$Y_i = \mathbb{I}[X = i] \qquad Y \in \{0, 1\}^K$$

$$p(Y = y \mid \theta) = \prod_{i=1}^{K} \theta_i^{y_i}$$

Dirichlet:

$$\theta \sim \text{Dirichlet}(\alpha)$$

$$p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

Posterior over Categorical Parameters?

$$p(Y = y \mid \theta) = \prod_{i=1}^{K} \theta_i^{y_i}$$

$$p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

posterior (given a single observation *y*):

$$p(\theta \mid y, \alpha) \propto$$

Posterior over Categorical Parameters?

$$p(Y = y \mid \theta) = \prod_{i=1}^{K} \theta_i^{y_i}$$

$$p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

posterior:

$$p(\theta \mid y, \alpha) \propto p(\theta \mid \alpha) p(y \mid \theta) \propto$$

Posterior over Categorical Parameters?

$$p(Y = y \mid \theta) = \prod_{i=1}^{K} \theta_i^{y_i}$$

$$p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

posterior:

$$p(\theta \mid y, \alpha) \propto p(\theta \mid \alpha) p(y \mid \theta) \propto \left(\prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \right) \times \left(\prod_{i=1}^{K} \theta_i^{y_i} \right)$$

Posterior over Categorical Parameters?

$$p(Y = y \mid \theta) = \prod_{i=1}^{K} \theta_i^{y_i}$$

$$p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$$

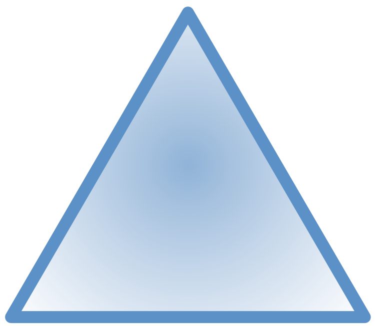
posterior:

$$p(\theta \mid y, \alpha) \propto p(\theta \mid \alpha) p(y \mid \theta) \propto \left(\prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \right) \times \left(\prod_{i=1}^{K} \theta_i^{y_i} \right)$$

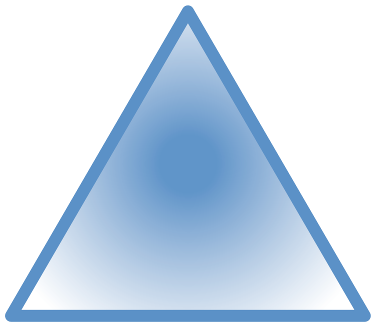
$$= \prod_{i=1}^{K} \theta_i^{\alpha_i + y_i - 1}$$

Posterior over Categorical Parameters?

prior: $p(\Theta = \theta \mid \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}$



posterior: $p(\theta \mid y, \alpha) \propto \prod_{i=1}^{K} \theta_i^{\alpha_i + y_i - 1}$



posterior has form of another Dirichlet distribution!

posterior parameters: $\alpha' = \alpha + y$

Conjugate Priors

- Dirichlet is (simplest) conjugate prior to multinomial
  - Dirichlet parameters are like "pseudo-observations"
- definition: "posterior obtained from a given prior in the prior family and a given likelihood function belongs to the same prior family"
- result of "algebraic similarity" between prior family and likelihood
- often leads to tractability & closed-form analytic solutions for posterior

Generative Story Template

1: Draw a set of parameters θ from $p(\Theta)$

2: Draw a latent structure z from $p(Z \mid \theta)$

3: Draw the observed data x from $p(X \mid z, \theta)$

the above generative story implies the following factorization of the joint distribution:

$$p(x, z, \theta) = p(\theta)p(z \mid \theta)p(x \mid z, \theta)$$

less Bayesian

more Bayesian

- using a prior distribution over parameters (not even really Bayesian)

1: Draw a set of parameters θ from $p(\theta \mid \alpha)$

2: Draw a latent structure z from $p(z \mid \theta)$

3: Draw the observed data x from $p(x \mid z, \theta)$

$$p(x, z, \theta \mid \alpha) = p(\theta \mid \alpha)\, p(z \mid \theta)\, p(x \mid z, \theta)$$

less Bayesian

- using a prior distribution over parameters (not even really Bayesian)
- computing posterior over parameters instead of using a point estimate

more Bayesian

- computing posterior over parameters instead of using a point estimate

$$p(x, z, \theta \mid \alpha) = p(\theta \mid \alpha)\, p(z \mid \theta)\, p(x \mid z, \theta)$$

data is a set of samples: $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$

joint: $p(x^{(1)}, \ldots, x^{(n)}, z^{(1)}, \ldots, z^{(n)}, \theta \mid \alpha)$

posterior with point estimate: $p(z^{(1)}, \ldots, z^{(n)} \mid x^{(1)}, \ldots, x^{(n)}, \hat{\theta}, \alpha)$

posterior: $p(z^{(1)}, \ldots, z^{(n)}, \theta \mid x^{(1)}, \ldots, x^{(n)}, \alpha)$

less Bayesian

more Bayesian

- using a prior distribution over parameters (not even really Bayesian)
- computing posterior over parameters instead of using a point estimate
- integrating out parameters (with fixed parameters for prior)

- integrating out parameters (with fixed parameters for prior)

$$p(x, z, \theta \mid \alpha) = p(\theta \mid \alpha)\, p(z \mid \theta)\, p(x \mid z, \theta)$$

data is a set of samples: $x^{(1)}, x^{(2)}, ..., x^{(n)}$

joint: $p(x^{(1)}, ..., x^{(n)}, z^{(1)}, ..., z^{(n)}, \theta \mid \alpha)$

posterior: $p(z^{(1)}, ..., z^{(n)}, \theta \mid x^{(1)}, ..., x^{(n)}, \alpha)$

collapsed posterior: $p(z^{(1)}, ..., z^{(n)} \mid x^{(1)}, ..., x^{(n)}, \alpha)$

less Bayesian

more Bayesian

- using a prior distribution over parameters (not even really Bayesian)
- computing posterior over parameters instead of using a point estimate
- integrating out parameters (with fixed parameters for prior)
- integrating out parameters while estimating parameters of prior ("Empirical Bayes")
- integrating out parameters *and* prior parameters (using a "hyperprior")
- …

Inference

- inference roughly means "calculate statistical quantities of interest"
- examples:
  - compute the mode of some random variables when conditioning on some and marginalizing out others
  - compute marginals of some random variables (variable posteriors when marginalizing out everything else)
  - compute posterior distribution over some subset of random variables

Learning?

- in Bayesian NLP, there's often no "learning"
- there is only "inference"
- just define model and do inference to calculate what we want to calculate
  - no parameters are being estimated from data*
  - we are not optimizing any loss function*
  - there is no gradient descent*
- but sometimes we do learn some latent variables (certain parameters or hyperparameters), and infer or marginalize over others

* typically

Markov Chain Monte Carlo (MCMC)

- MCMC algorithms are widely used in Bayesian modeling but also useful more generally
- can be used to generate samples from distributions that are hard to sample from
- samples can be used to estimate quantities of interest
- these estimates are unbiased

Gibbs Sampling

- Gibbs sampling is the simplest and most widely-used MCMC algorithm (at least in NLP)

Gibbs Sampling Template

$U_1, ..., U_p =$ latent variables

$U_{-i} =$ all latent variables other than U_i

$\boldsymbol{X} =$ all observed data and hyperparameters

Gibbs sampling:

initialize all U_i to values u_i

repeat until convergence:

sample u from $p(U_i \mid u_{-i}, \boldsymbol{X})$

set $U_i \leftarrow u$

Gibbs Sampling Template

Gibbs sampling:

  initialize all U_i to values u_i

  repeat until convergence:

    sample u from $p(U_i \mid u_{-i}, \boldsymbol{X})$

    set $U_i \leftarrow u$

At convergence, each time we update any value of any random variable in $U_1, \ldots, U_p$, we have another sample from the posterior

these samples can be used to estimate any quantity of interest while offering some nice theoretical properties

Disadvantages of Gibbs Sampling?

Gibbs sampling:

initialize all U_i to values u_i

repeat until convergence:

sample u from $p(U_i \mid u_{-i}, \boldsymbol{X})$

set $U_i \leftarrow u$

nearby samples are not necessarily uncorrelated, so it can take many samples for good estimates, especially of rare events

guarantees are at convergence

"burn-in" time can be hard to estimate & depends on initialization

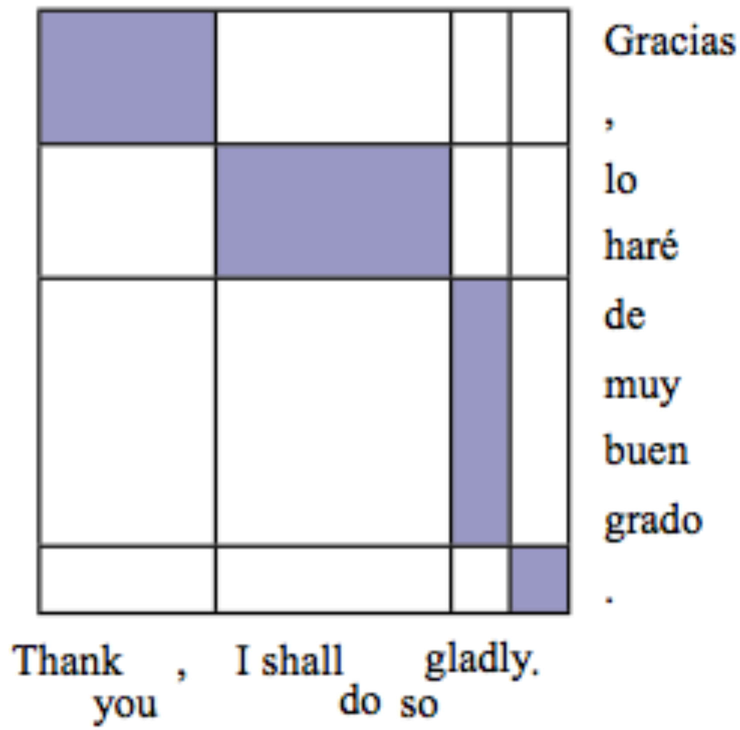
- Gibbs sampling is simple and has nice guarantees, but it can be tricky to derive for NLP models

- why? we just need to sample each random variable conditioned on all the others

- in certain kinds of NLP models, hard to define the random variables!

- even when we can do this, the sampler might be very slow to converge ("mix slowly")

Example: Phrase Alignments in Machine Translation



(b) example phrase alignment

DeNero et al. (2008): *Sampling Alignment Structure under a Bayesian Translation Model*

Example: Phrase Alignments in Machine Translation

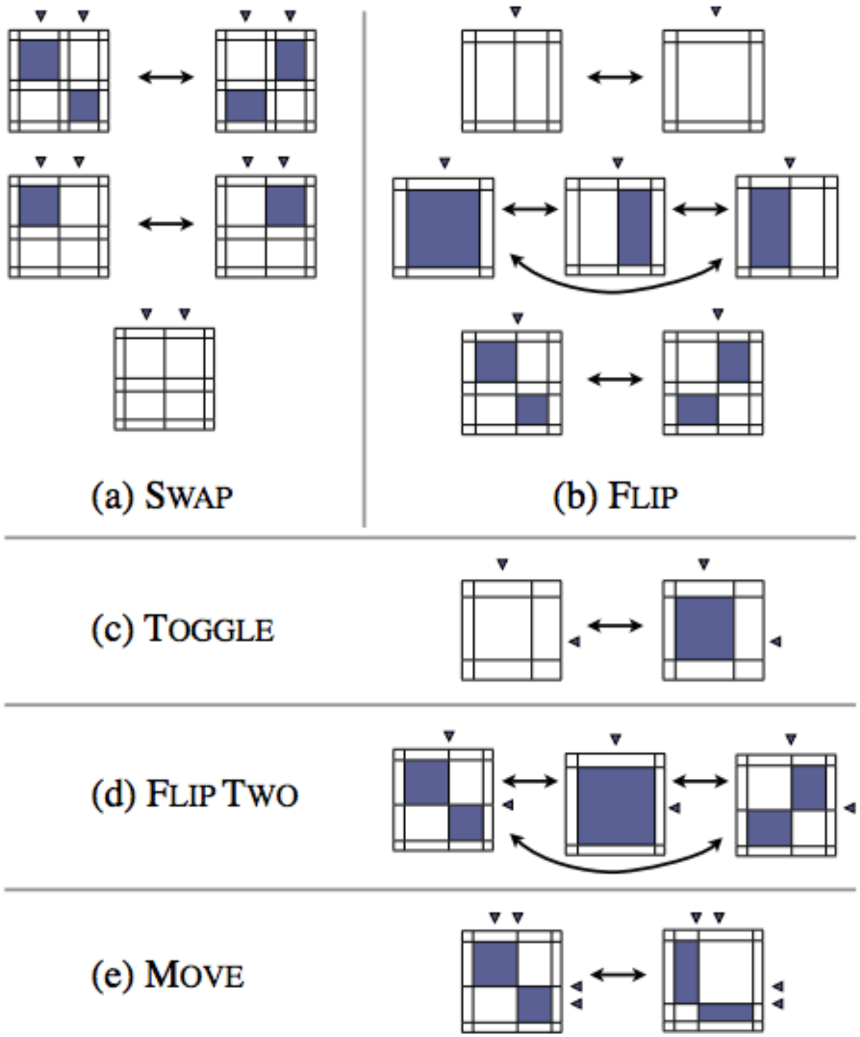


Figure 2: Each local operator manipulates a small portion of a single alignment. Relevant phrases are exaggerated for clarity. The outcome sets (depicted by arrows) of each possible configuration are fully connected. Certain con-

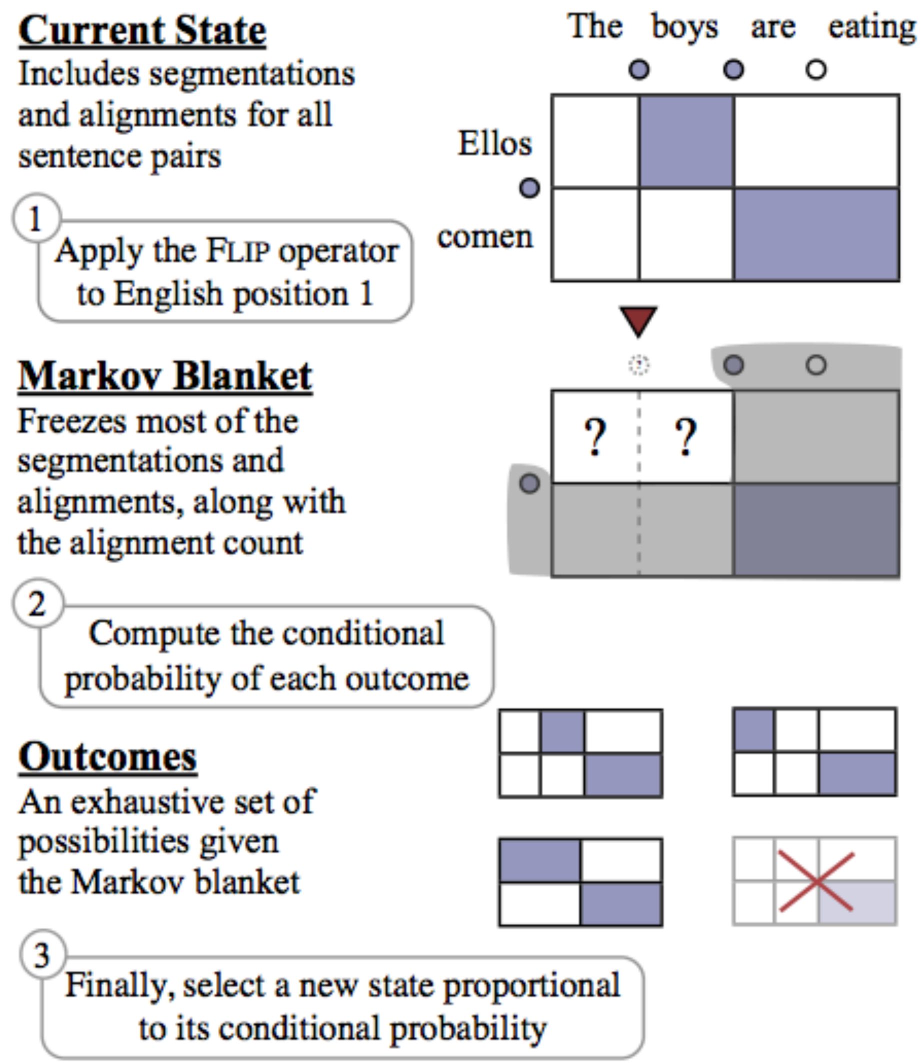
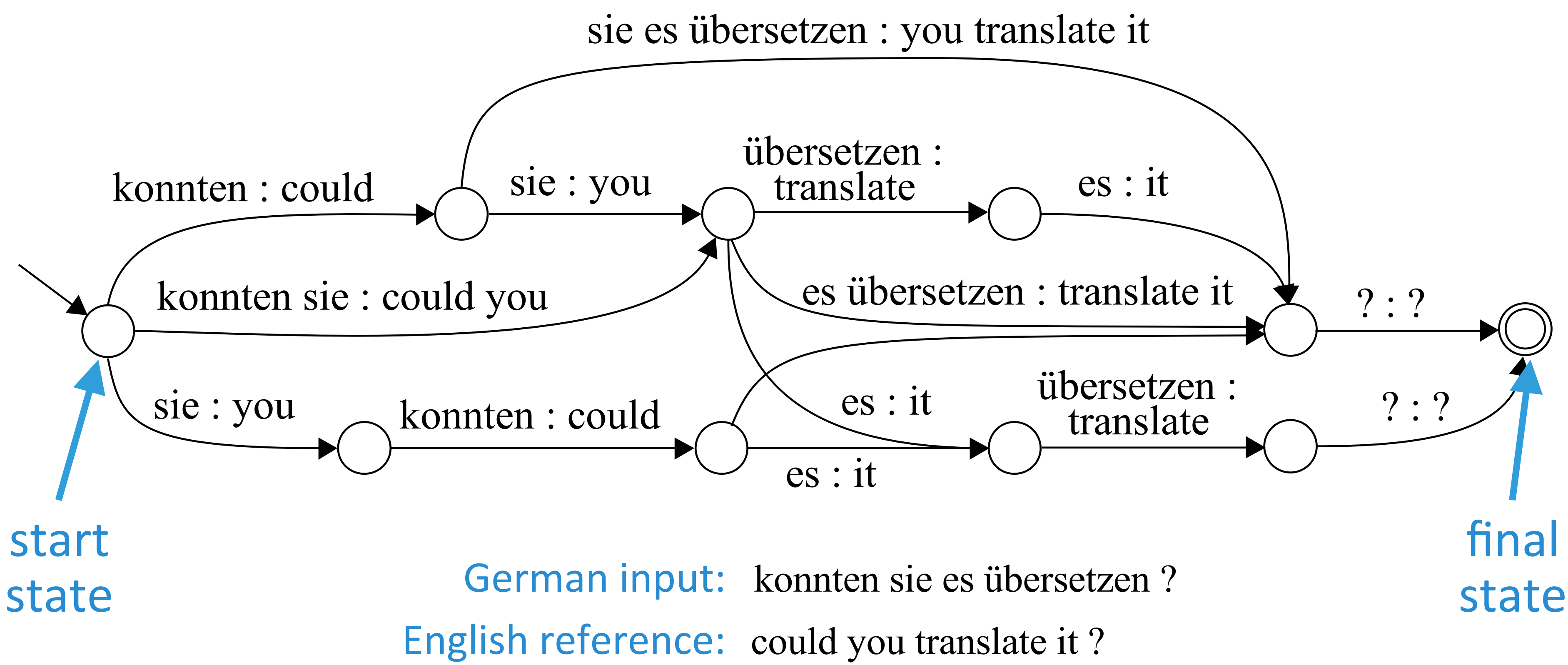


Figure 3: The three steps involved in applying the FLIP operator. The Markov blanket freezes all segmentations

Graphical Models in NLP?

- Gibbs sampling is easy to apply to graphical models, but graphical models are not a good fit for certain tasks/models in NLP:
  - segmentation
  - context-free grammars (see case-factor diagrams; McAllester et al., 2007)
  - finite-state automata
  - models over paths in graphs
  - models over hyperpaths in hypergraphs

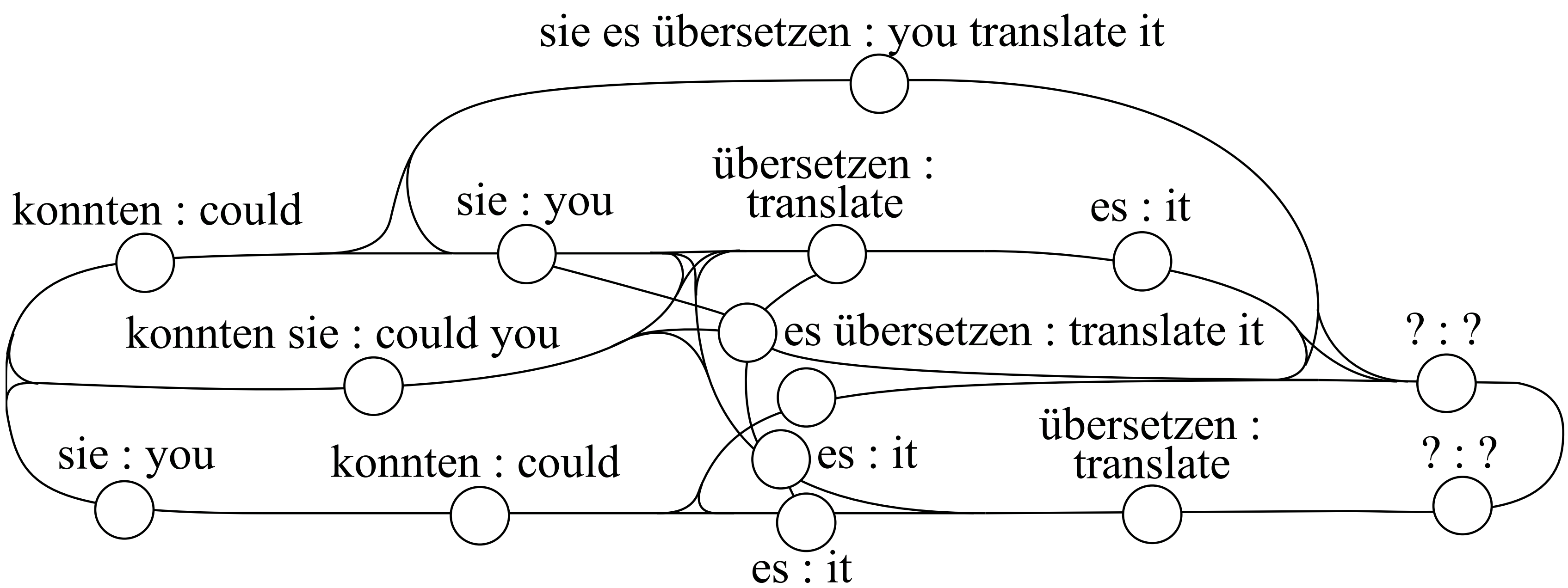
Example: Lattice for Phrase-Based Machine Translation



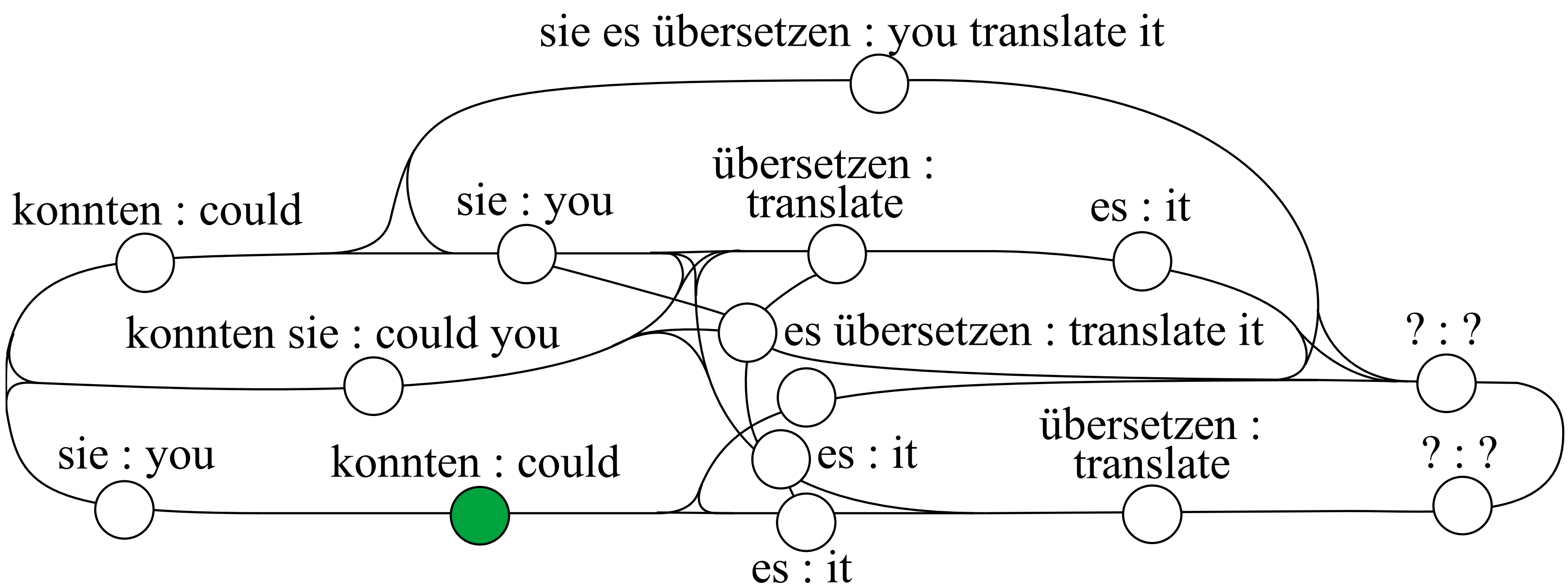
- this is a finite-state transducer: a directed graph where each edge consumes part of the input and outputs a string
- each edge has a score (not shown)
- a translation is a path from the start state to a final state

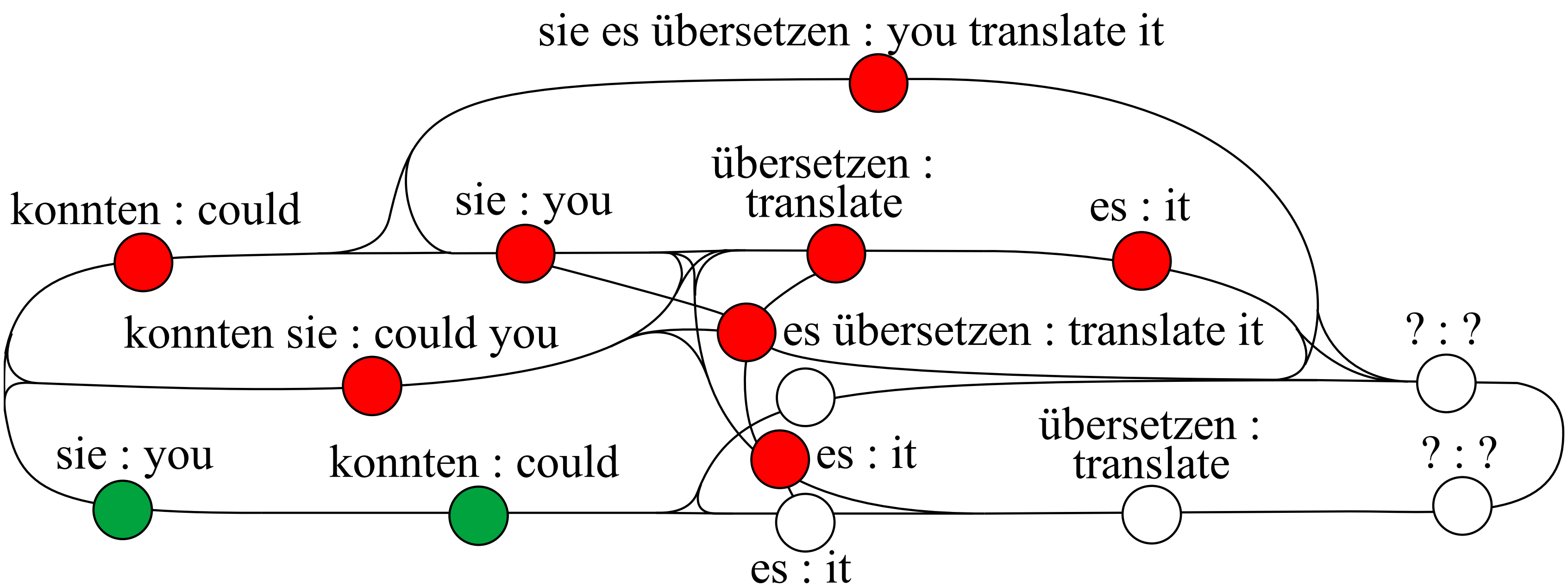
Phrase Lattice → Graphical Model

- each edge in the phrase lattice is a binary random variable in the graphical model



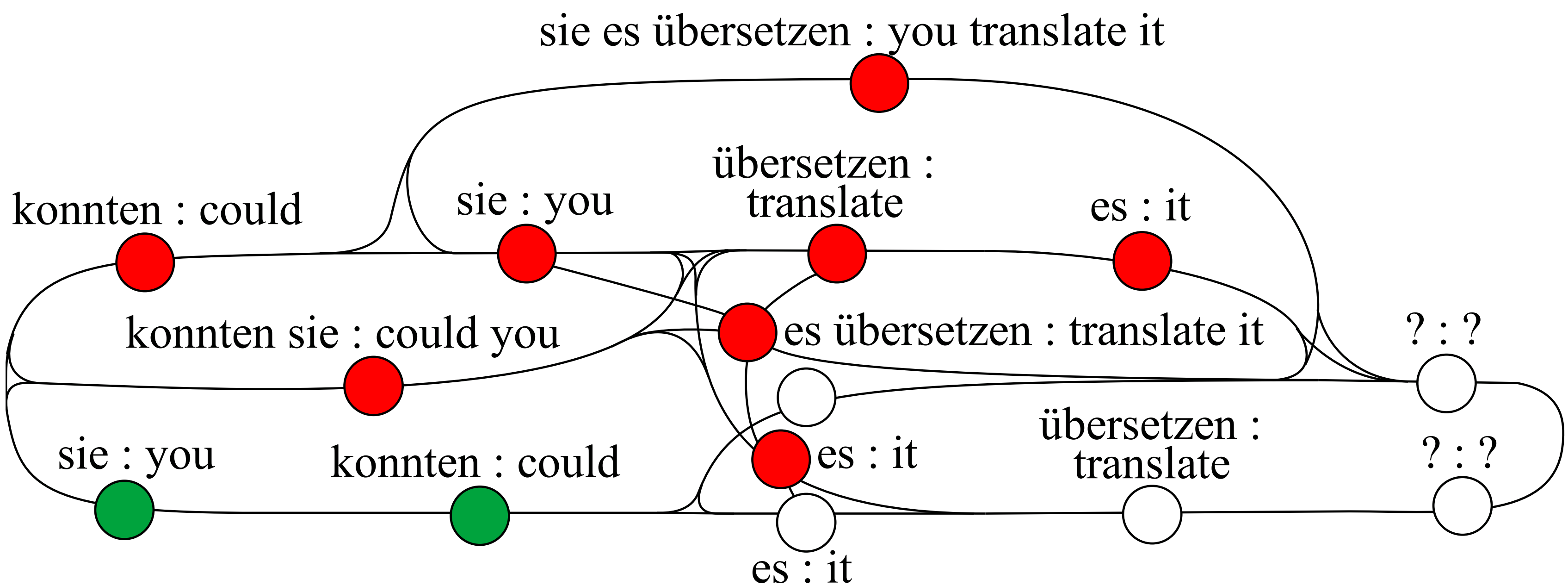
- consider what happens when we set a variable to 1 (green)



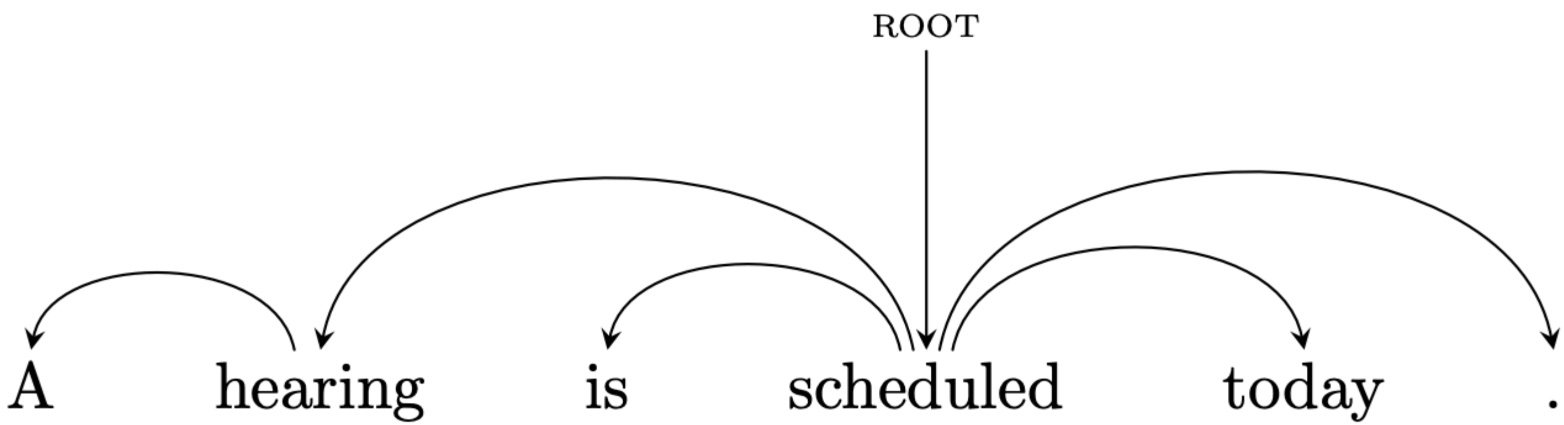


- just by setting one variable to 1, many other variables are forced to be 0 or 1 to obey path legality constraints

- long-distance, deterministic dependencies among variables
- known to be problematic for certain inference algorithms (Gibbs sampling and belief propagation)



Graphical Models for Dependency Parsing



- define a binary random variable for each pair of words in the sentence
- global tree constraint among all random variables (special handling for this constraint)

Smith & Eisner (2008): *Dependency Parsing by Belief Propagation*

Summary: Graphical Models in NLP

- we can often come up with a way to define random variables for structured NLP tasks
- downside: every variable may have an edge to all others! (global constraints)
- global, deterministic potentials can cause issues with certain general-purpose inference algorithms in graphical models
- it's better to use specialized algorithms designed for the global constraints

LDA Generative Story

1: For each topic $k = 1...K$, draw multinomial word distribution $\beta_k \sim \text{Dirichlet}(\psi)$

2: For each document i:

  a: Draw a multinomial topic distribution $\theta^{(i)} \sim \text{Dirichlet}(\alpha)$

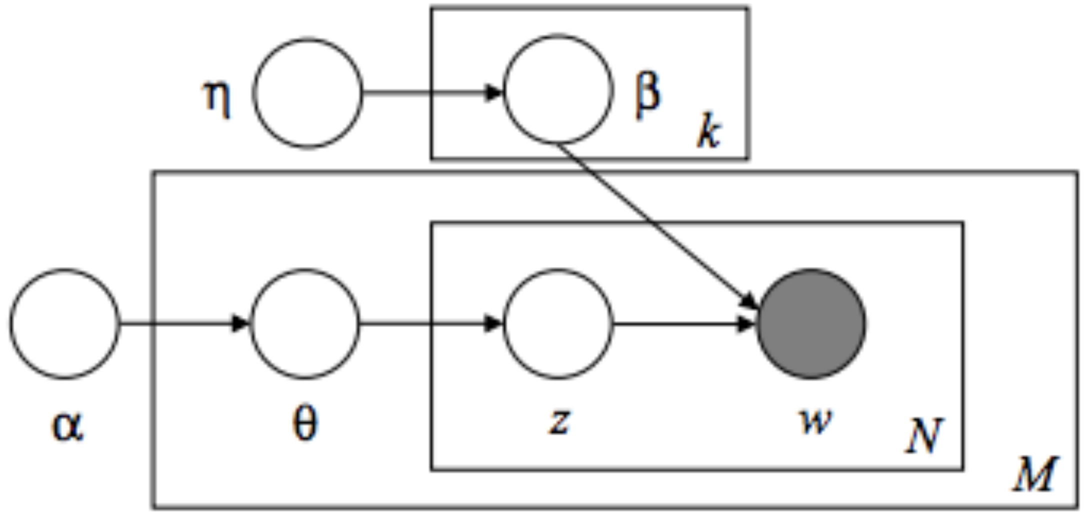
  b: For each position j in document i:

   i: Draw a topic $z^{(i,j)} \sim \text{Multinomial}(\theta^{(i)})$

   ii: Draw a word $w^{(i,j)} \sim \text{Multinomial}(\beta_{z^{(i,j)}})$

K = # topics
N = # documents
M = # words in each document
V = # words in vocabulary

Graphical Model for LDA



Gibbs Sampling for LDA

$$Z^{(i,j)} \mid \text{everything else} \sim \text{Multinomial}(\theta^{(i)} \odot \beta_{\cdot,w^{(i,j)}})$$

$$\theta^{(i)} \in \mathbb{R}^K$$

$$\beta \in \mathbb{R}^{K \times V}$$

Gibbs Sampling for LDA

$$Z^{(i,j)} \mid \text{everything else} \sim \text{Multinomial}(\theta^{(i)} \odot \beta_{\cdot,w^{(i,j)}})$$

$$\theta^{(i)} \mid \text{everything else} \sim \text{Dirichlet}(\alpha + m^{(i)})$$

$$\beta_k \mid \text{everything else} \sim \text{Dirichlet}(\psi + n_k)$$

$\theta^{(i)} \in \mathbb{R}^K$

$\beta \in \mathbb{R}^{K \times V}$

$m_k^{(i)} =$ # words in doc i from topic k

$n_{k,v} =$ # of times word v appears with topic k in any document

- we now have a way to generate samples from the posterior for the LDA model
- how should we do the following?
  - get topic assignments for each word in the document collection?
  - get topic distribution for a document?
  - get estimates of topic-word distributions for each topic?