Training and Inference Methods over Translation Forests

Zhifei Li

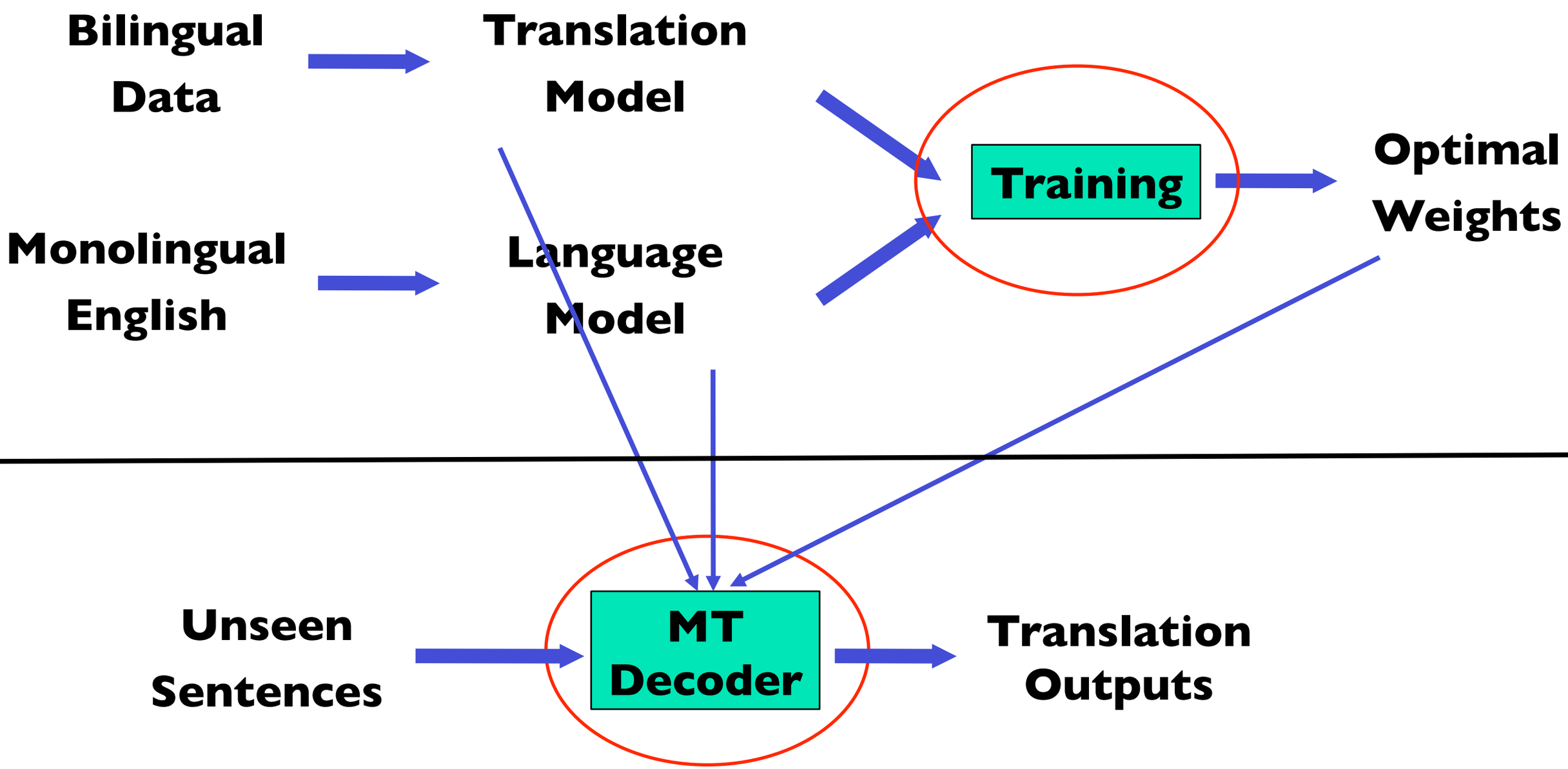
Center for Language and Speech Processing

Computer Science Department

Johns Hopkins University

Joint work with Sanjeev Khudanpur, Jason Eisner, Chris Callison-Burch, and many other team members in the Joshua project

Statistical Machine Translation Pipeline



垫子 上 的 猫

dianzi shang de mao

a cat on the mat

$\mathbf{X} \rightarrow \langle \mathbf{X_0}\ \mathbf{de}\ \mathbf{X_1}, \mathbf{X_1}\ \text{on}\ \mathbf{X_0} \rangle$

---

zhongguo de shoudu

capital of China

$\mathbf{X} \rightarrow \langle \mathbf{X_0}\ \mathbf{de}\ \mathbf{X_1}, \mathbf{X_1}\ \text{of}\ \mathbf{X_0} \rangle$

---

wo de mao

my cat

$\mathbf{X} \rightarrow \langle \mathbf{X_0}\ \mathbf{de}\ \mathbf{X_1}, \mathbf{X_0}\ \mathbf{X_1} \rangle$

---

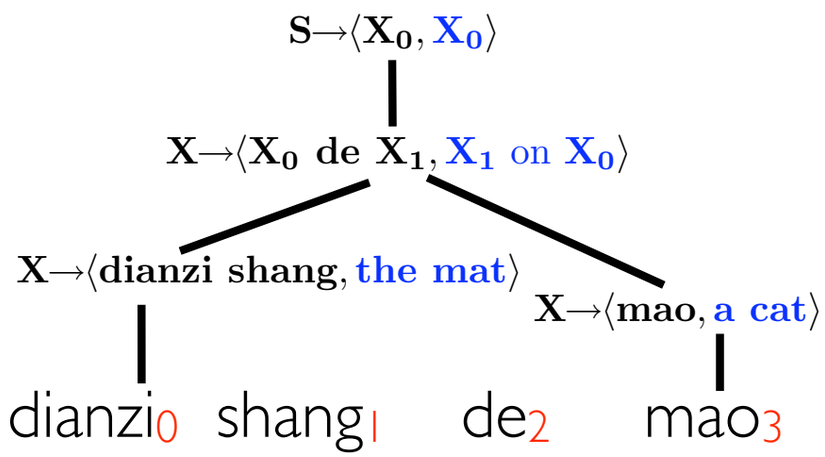
zhifei de mao

zhifei 's cat

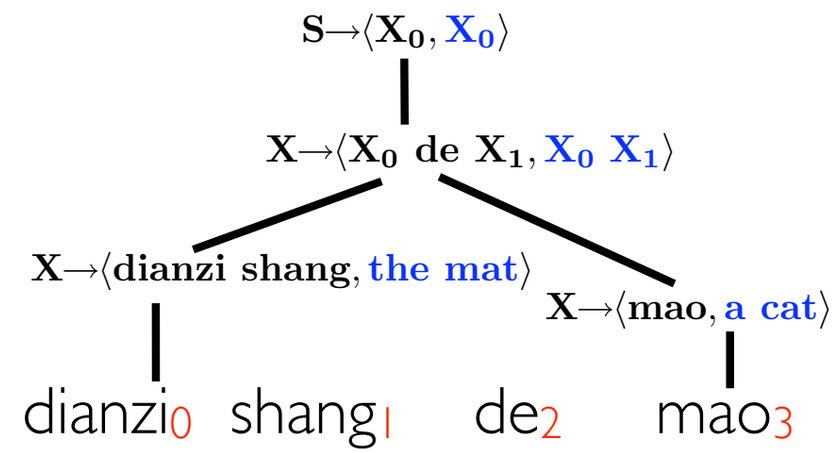
$\mathbf{X} \rightarrow \langle \mathbf{X_0}\ \mathbf{de}\ \mathbf{X_1}, \mathbf{X_0}\ \text{'s}\ \mathbf{X_1} \rangle$

Joshua
(chart parser)

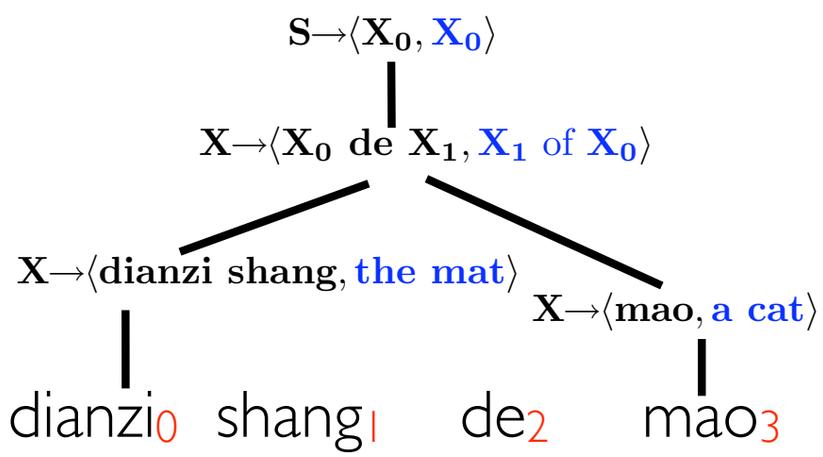
dianzi shang de mao



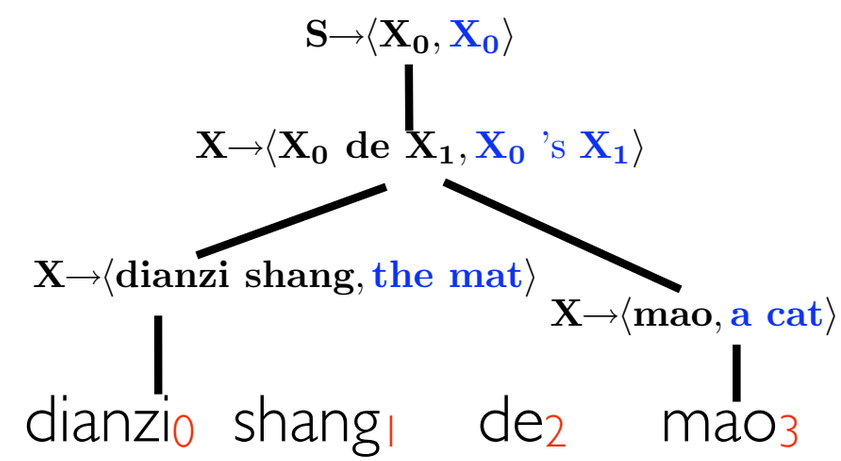
a cat on the mat



the mat a cat



a cat of the mat

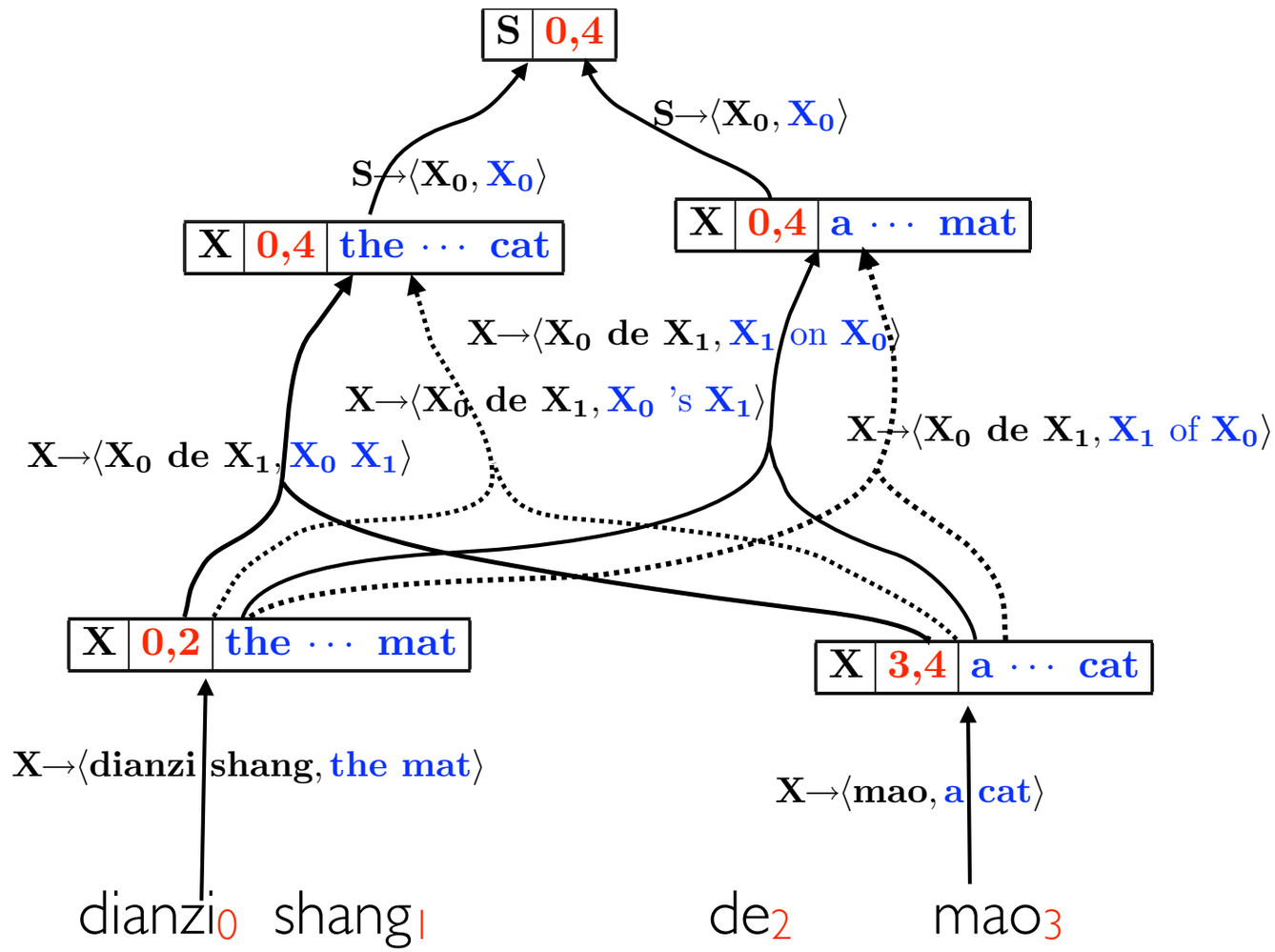


the mat 's a cat

Joshua
(chart parser)

dianzi shang de mao

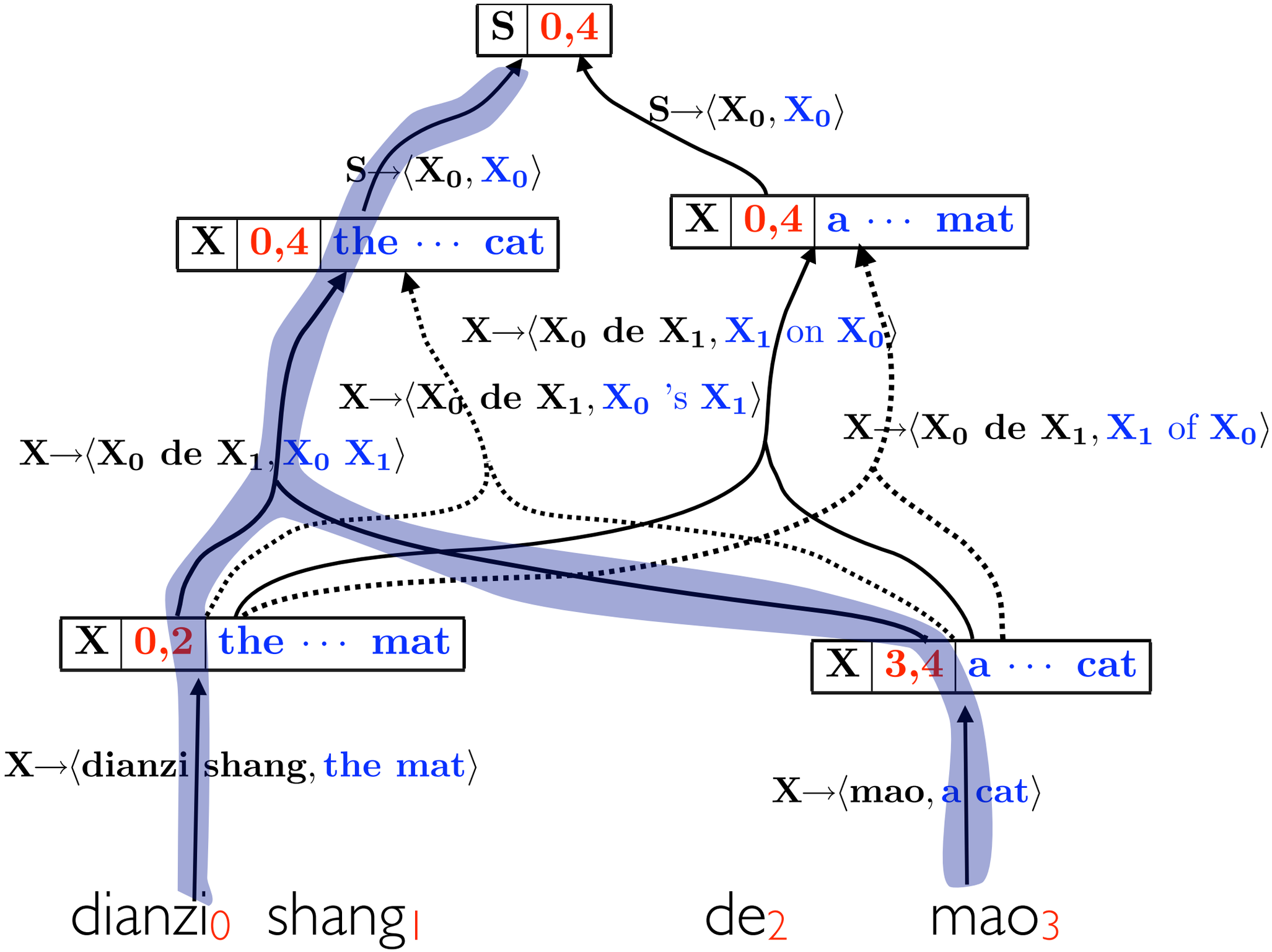
hypergraph



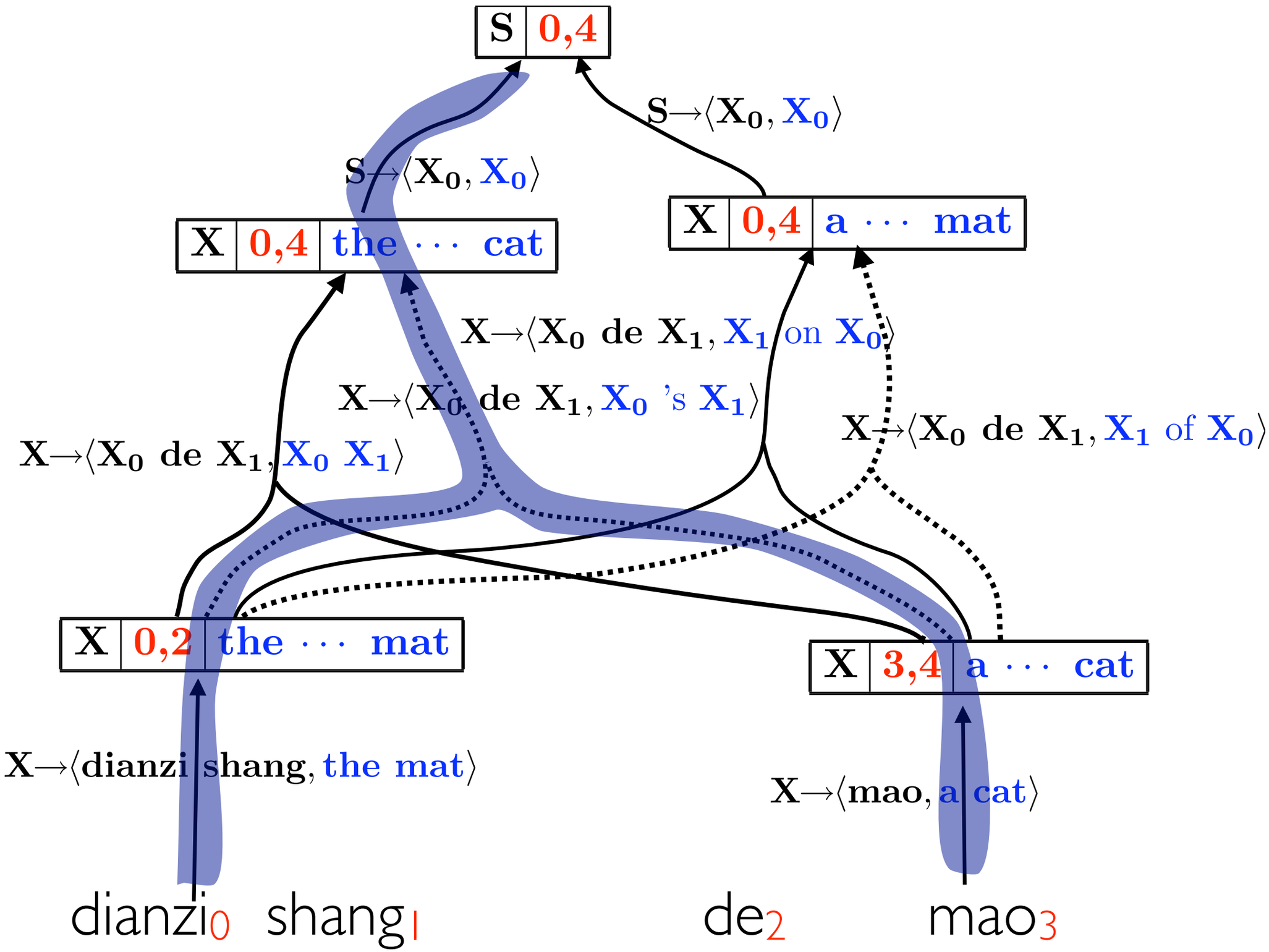
Joshua
(chart parser)

dianzi shang de mao

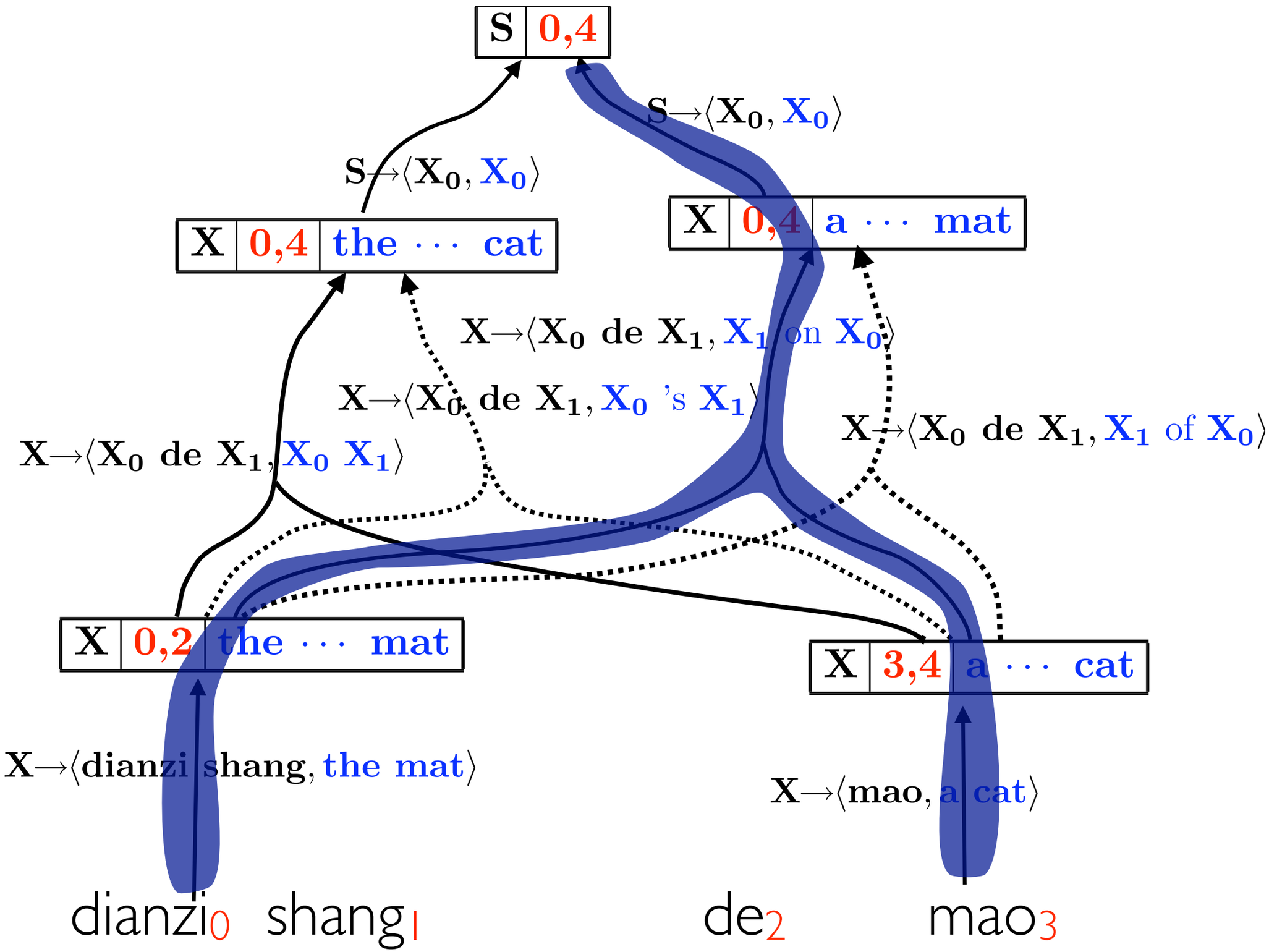
A hypergraph is a compact data structure to encode **exponentially many trees**.



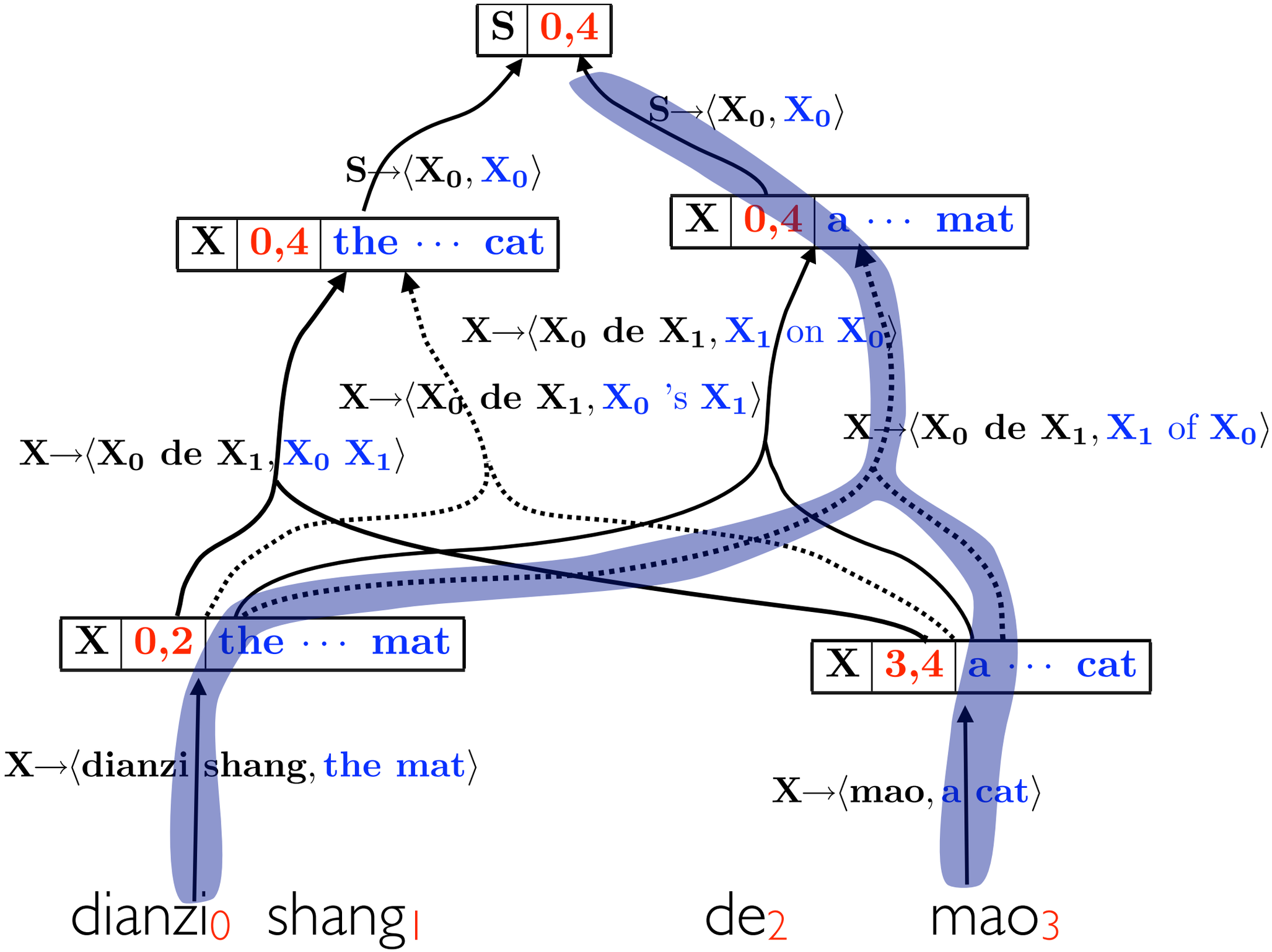
A hypergraph is a compact data structure to encode **exponentially many trees**.



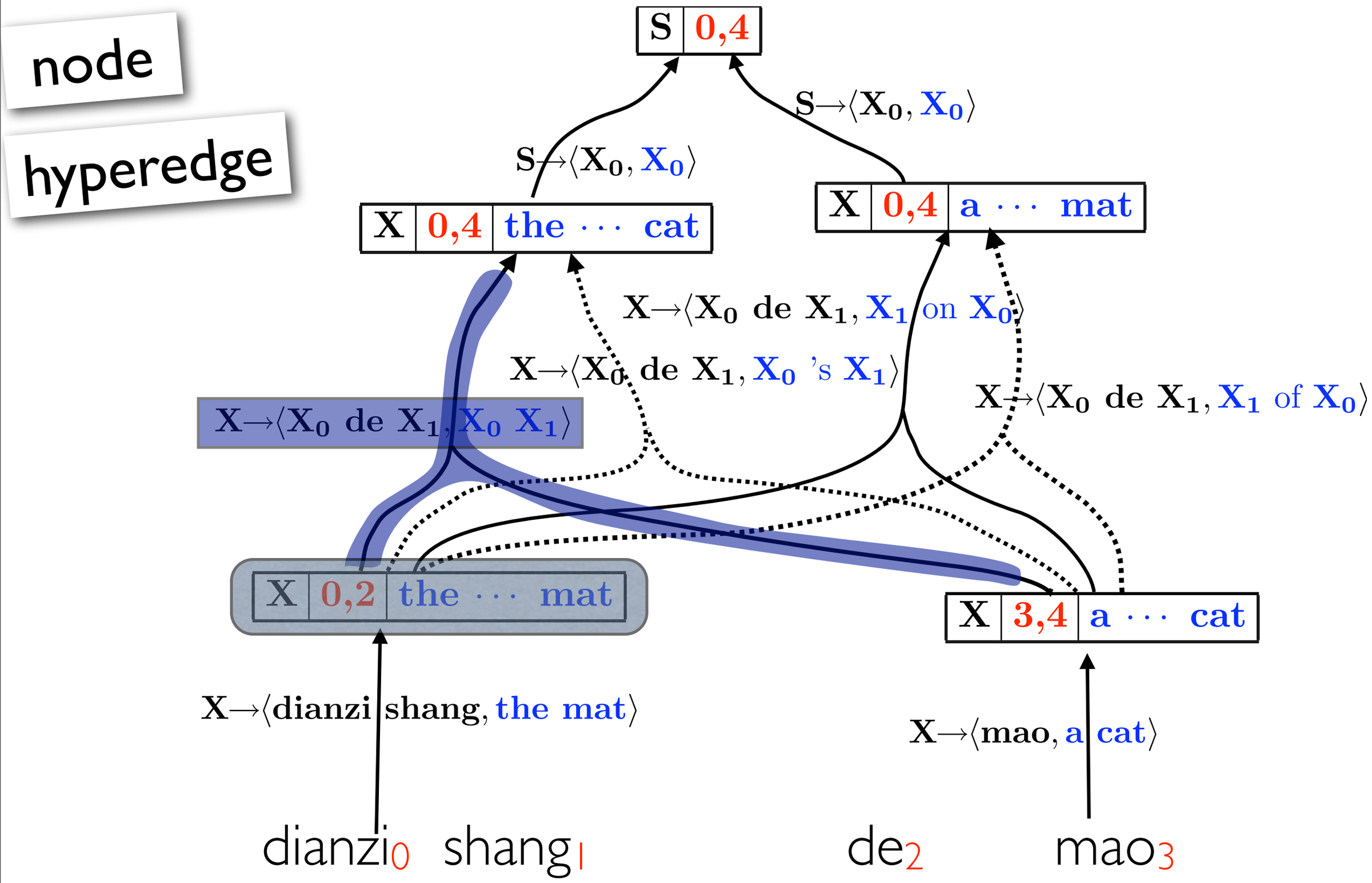
A hypergraph is a compact data structure to encode **exponentially many trees**.



A hypergraph is a compact data structure to encode **exponentially many trees**.



A hypergraph is a compact data structure to encode **exponentially many trees**.



Why hypergraphs?

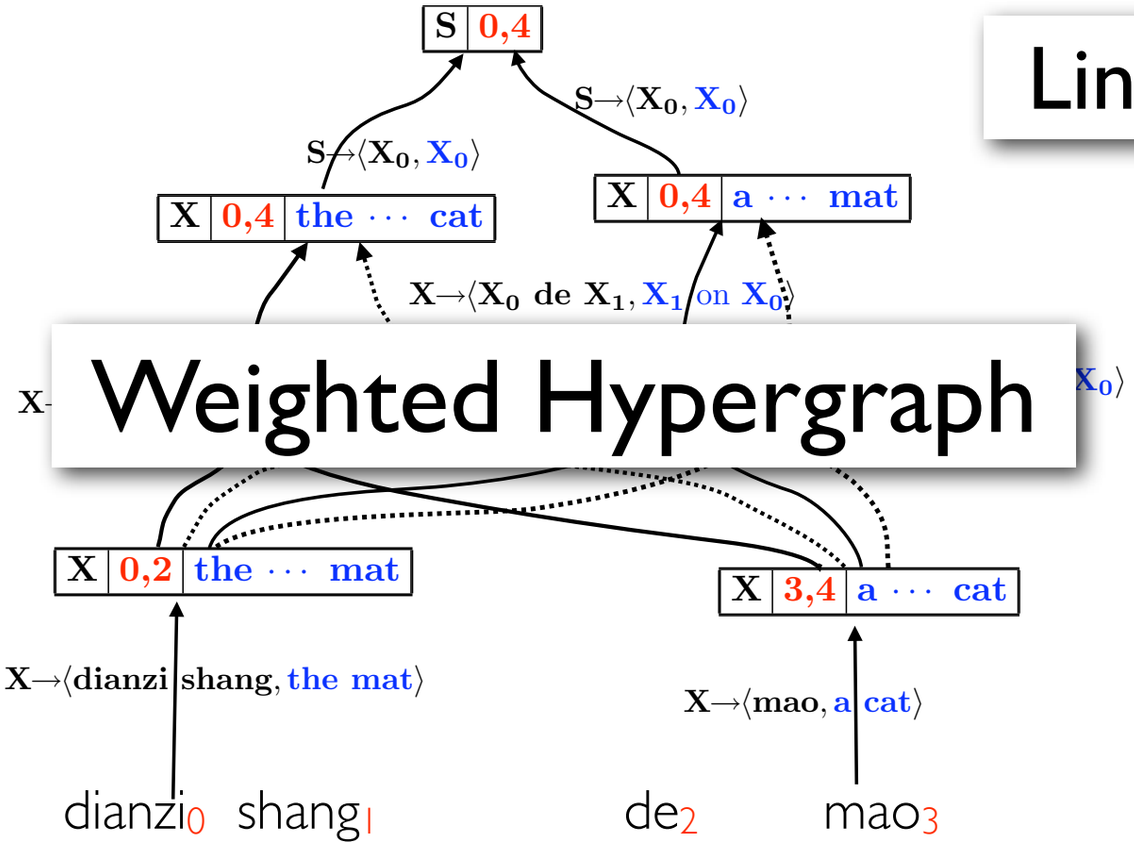
- Contains a much larger hypothesis space than an n-best list
- General compact data structure
  - special cases include lattice, finite state machine, and packed forest
    - can be used for speech, monolingual parsing, tree-based MT systems, and many more

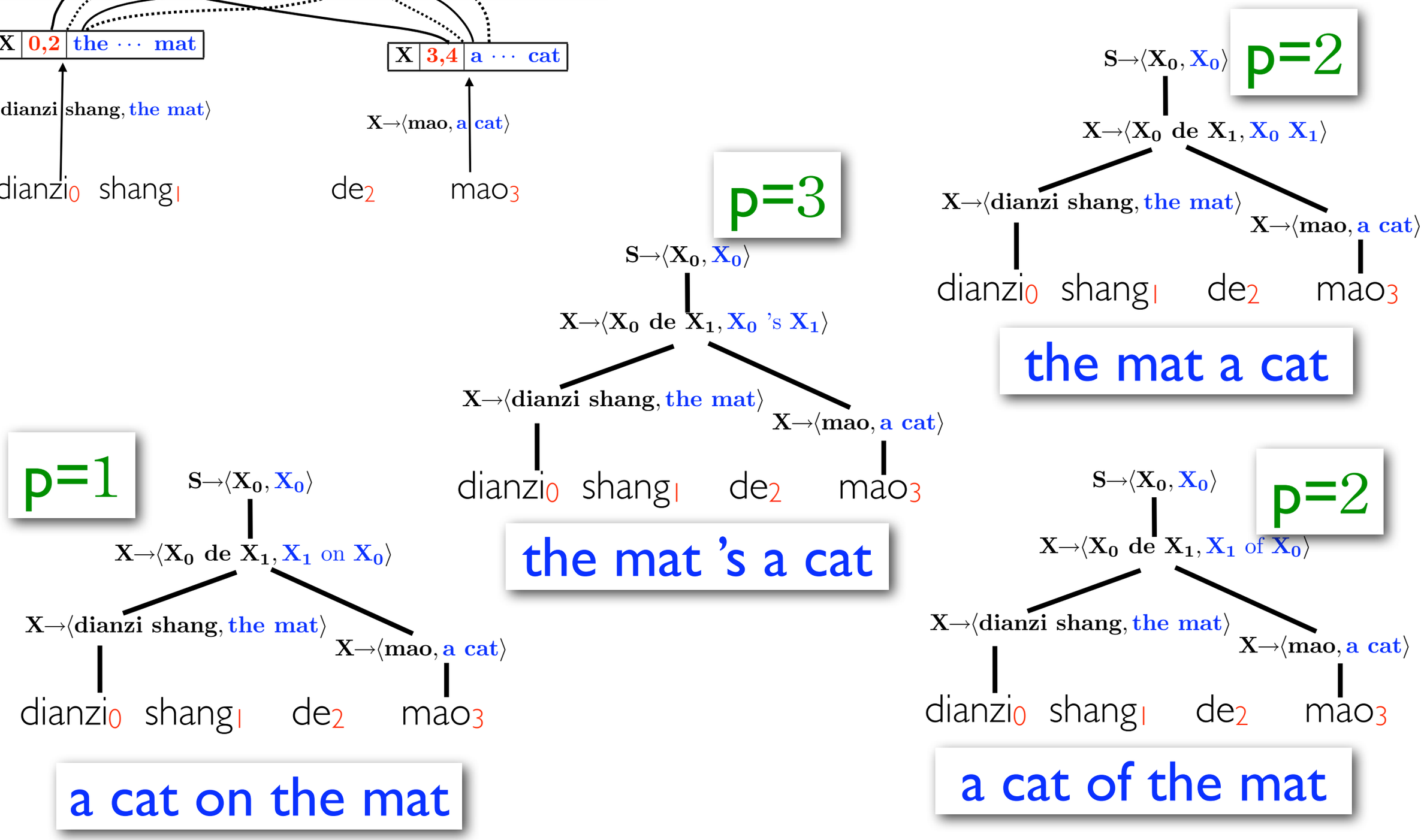
Linear model:

$$p(d \mid x) = \theta \cdot \Phi(d, x)$$

weights

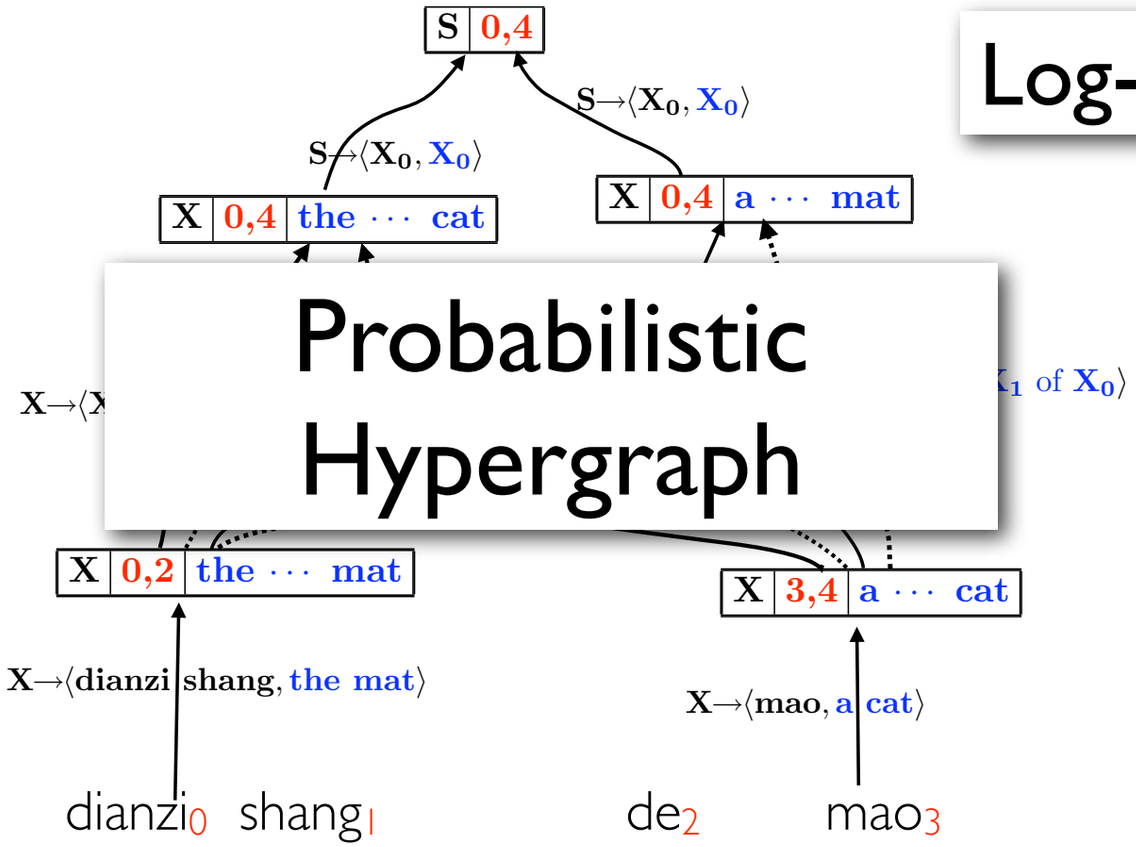
features





Log-linear model:

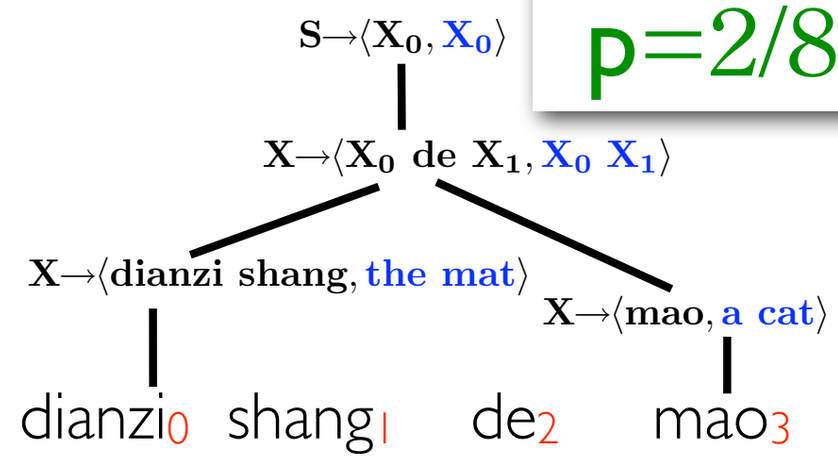
$$p(d \mid x) = \frac{e^{\theta \cdot \Phi(d,x)}}{Z(x)}$$



Probabilistic Hypergraph

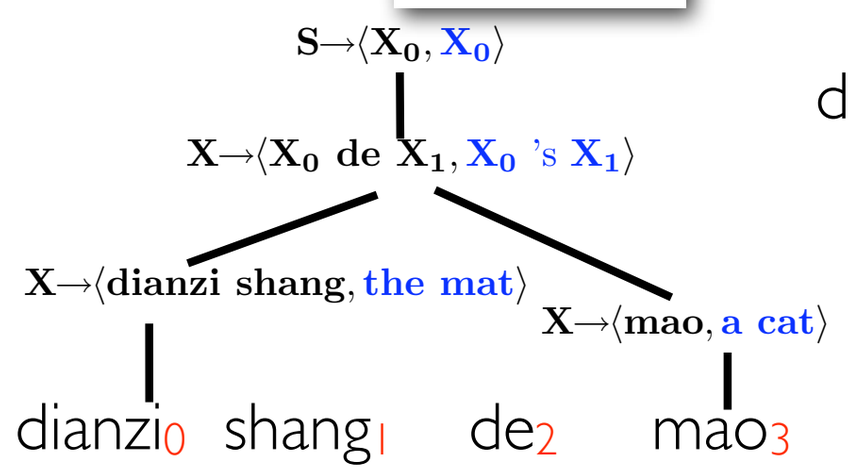
Z=2+1+3+2=8

p=2/8



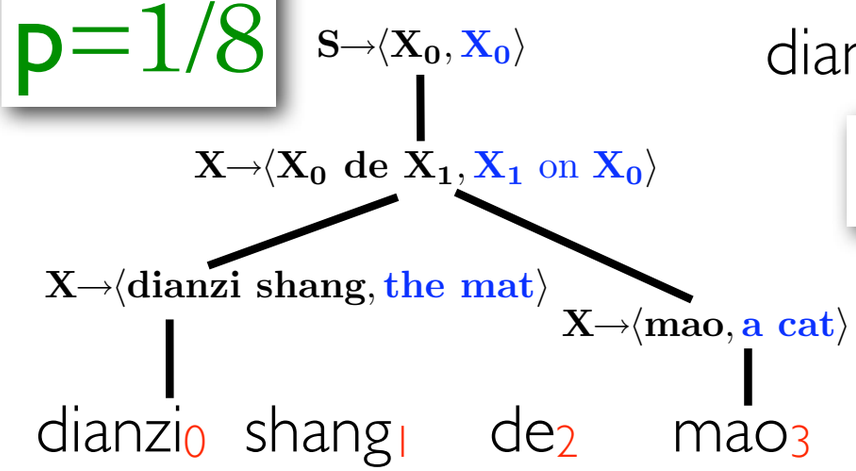
the mat a cat

p=3/8



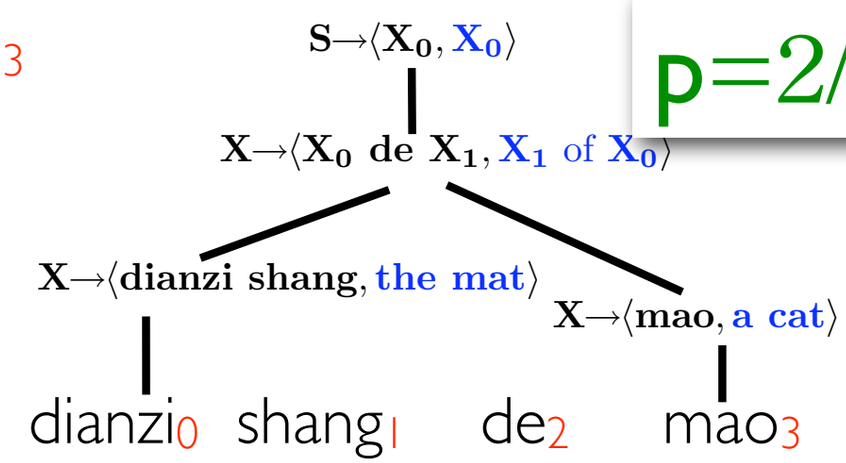
the mat 's a cat

p=1/8



a cat on the mat

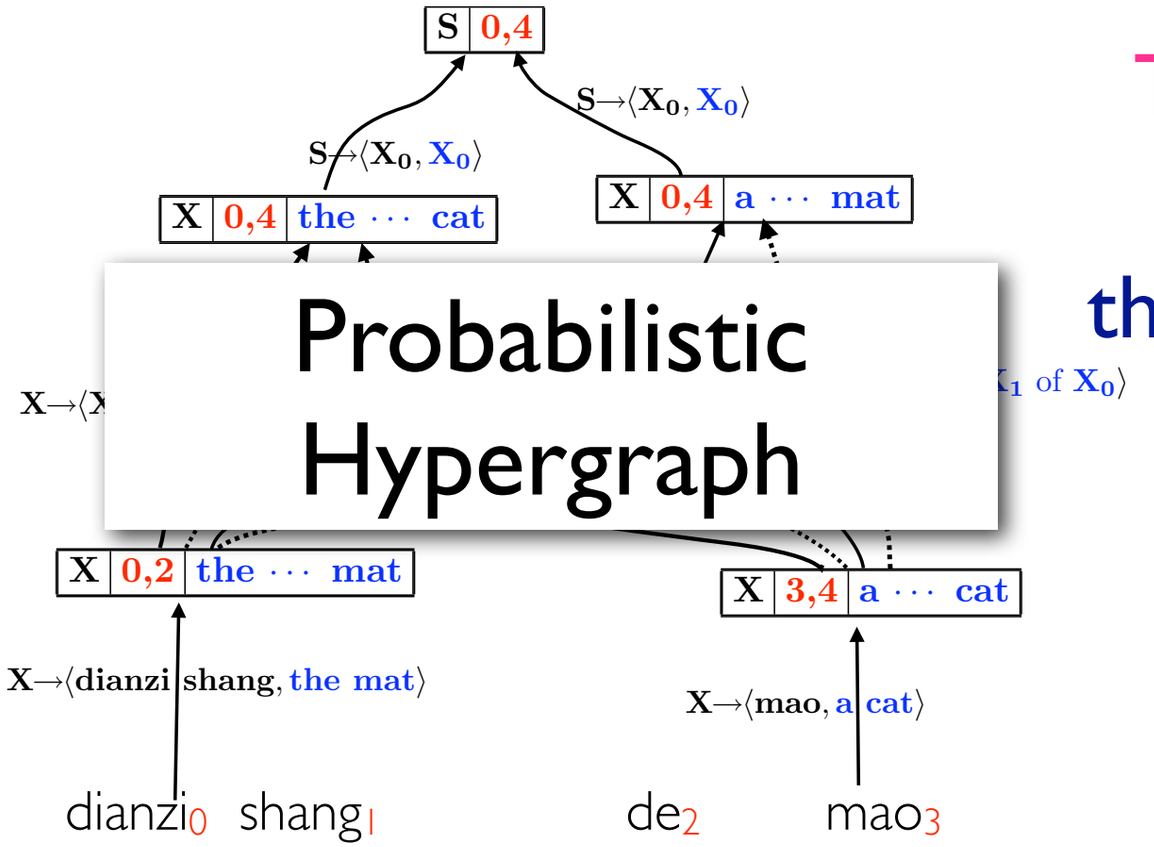
p=2/8

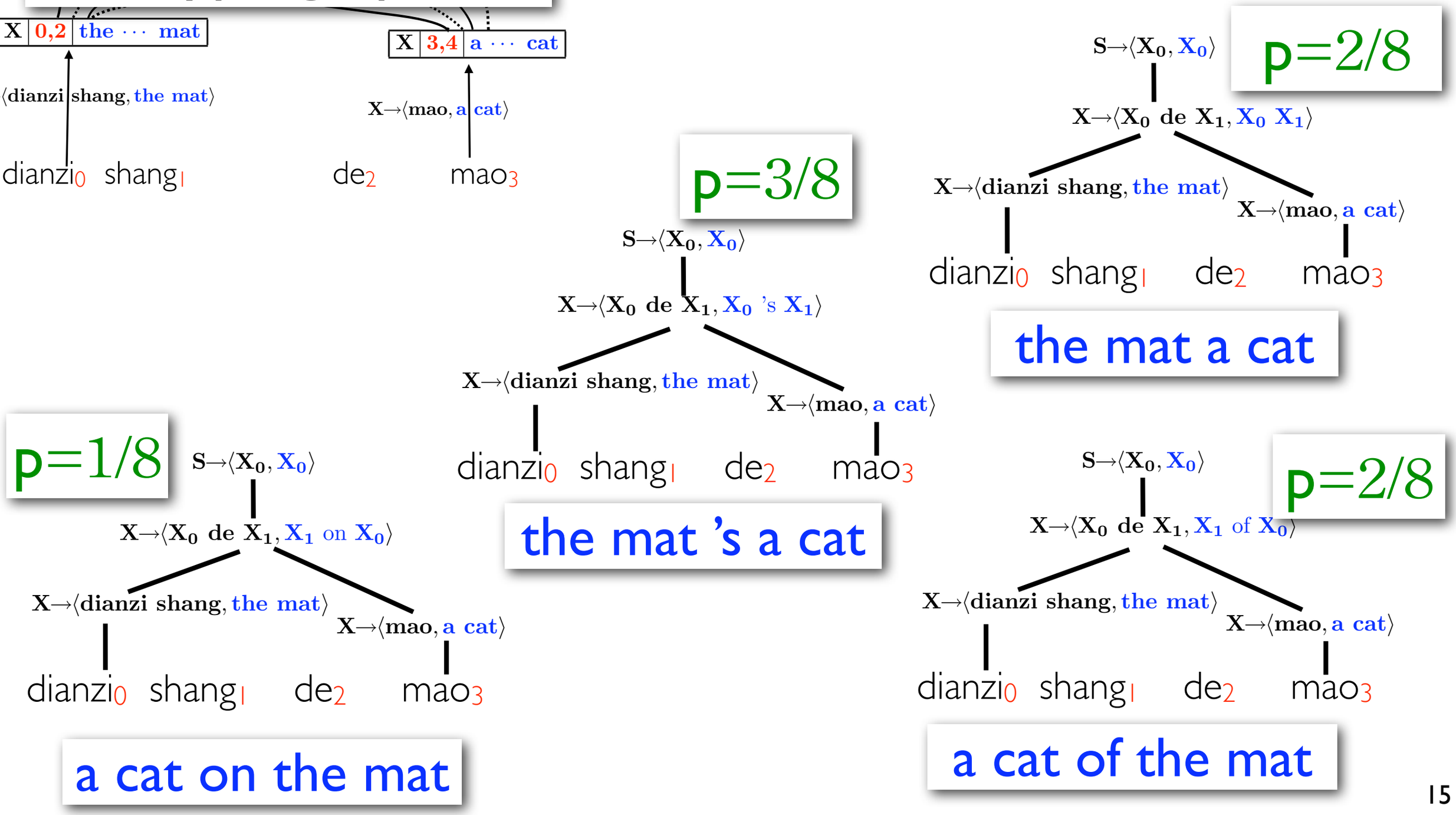


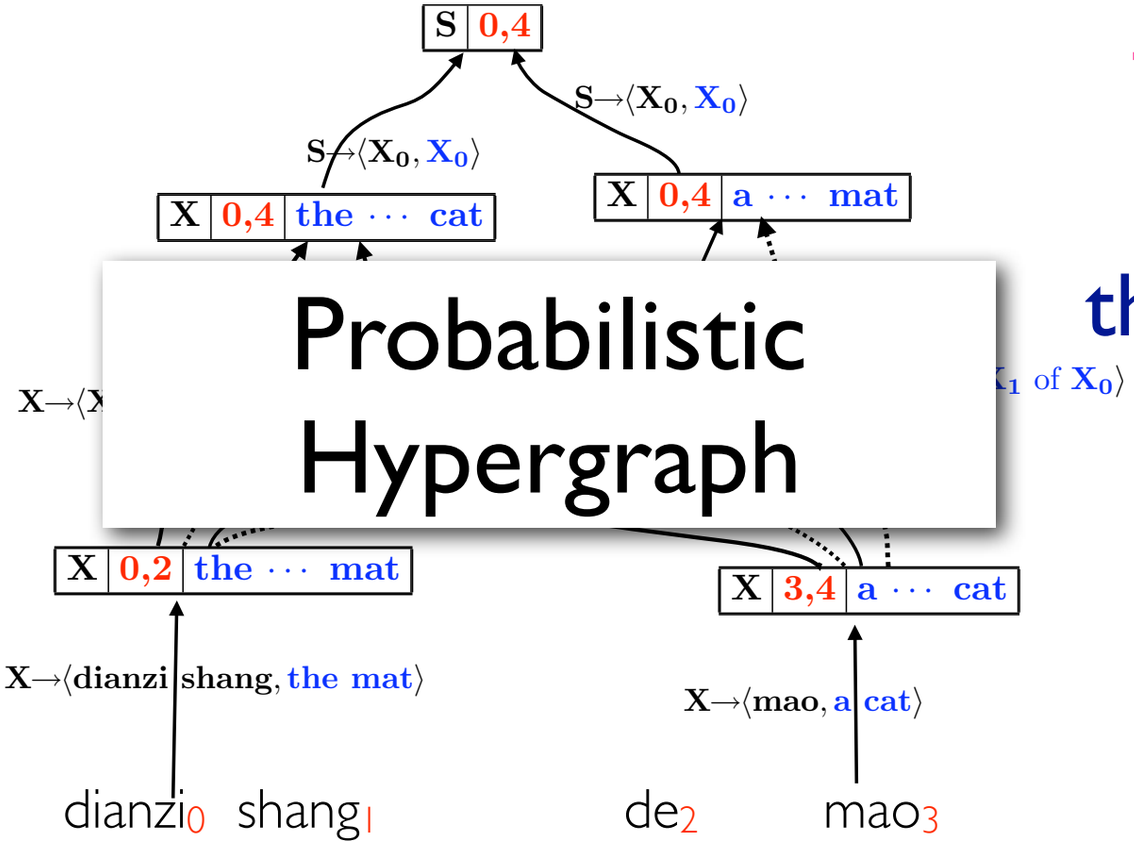
a cat of the mat

The hypergraph defines a probability distribution over **trees**!

the distribution is parameterized by Θ







Probabilistic Hypergraph

The hypergraph defines a probability distribution over **trees**!

the distribution is parameterized by Θ

What criterion do we use to set the parameters Θ? Training

Given a fixed Θ and a hypergraph, Inference

- how to compute the posterior of a hyperedge?
- how to choose a desired translation output?

Why are the problems difficult?

- brute-force will be too slow as there are exponentially many trees, so require sophisticated dynamic programs
- sometimes intractable, require approximations

Outline

- Hypergraph as hypothesis space
- Training methods (Li and Eisner, 2009)
- Exact and approximate inference
  - exact inference
    - ★ semiring framework (Li and Eisner, 2009)
  - approximate inference
    - ★ "principled" ones
      - ✓ variational decoding (Li et. al, 2009)
      - ✓ sampling and message passing methods
    - ★ heuristic-based ones
      - ✓ e.g., cube-pruning (Chiang, 2007)
- Joshua project (Li et. al, 2009)

Motivation for Training

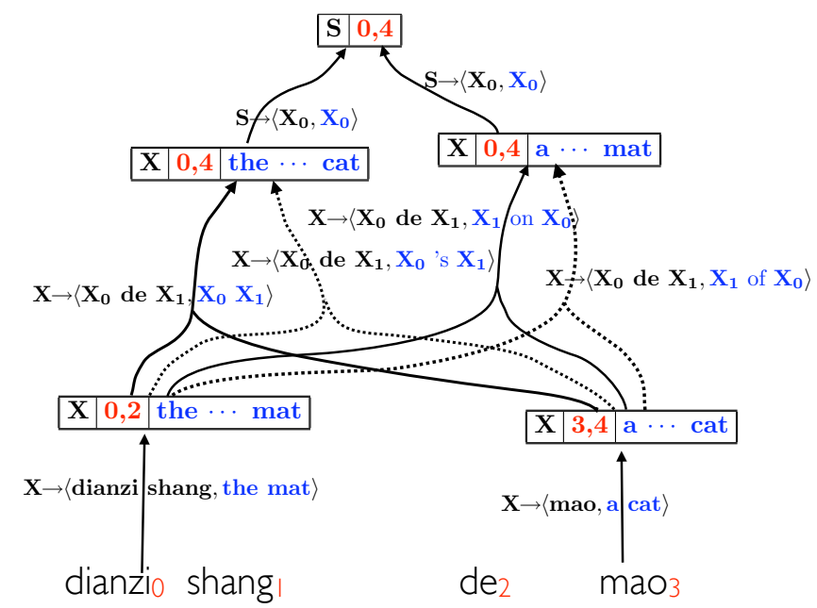
- Towards a training method that can
  - dealing with millions of features
  - incorporate linguistics or semantic information
  - incorporate non-local dependency
  - perform reliable training

Training Setup

- Each **training example** consists of
  - a hypergraph representing hypothesized translations
  - a reference translation

x: dianzi shang de mao

y*: a cat on the mat



- **Training**
  - adjust the parameters Θ so that the reference translation is preferred by the model

Training objectives

- Minimum error rate training (MERT)
- Minimum risk training
- **Minimum risk with deterministic annealing**
- Conditional random field (CRF)
- Perceptron
- MIRA

Minimum Error Rate Training

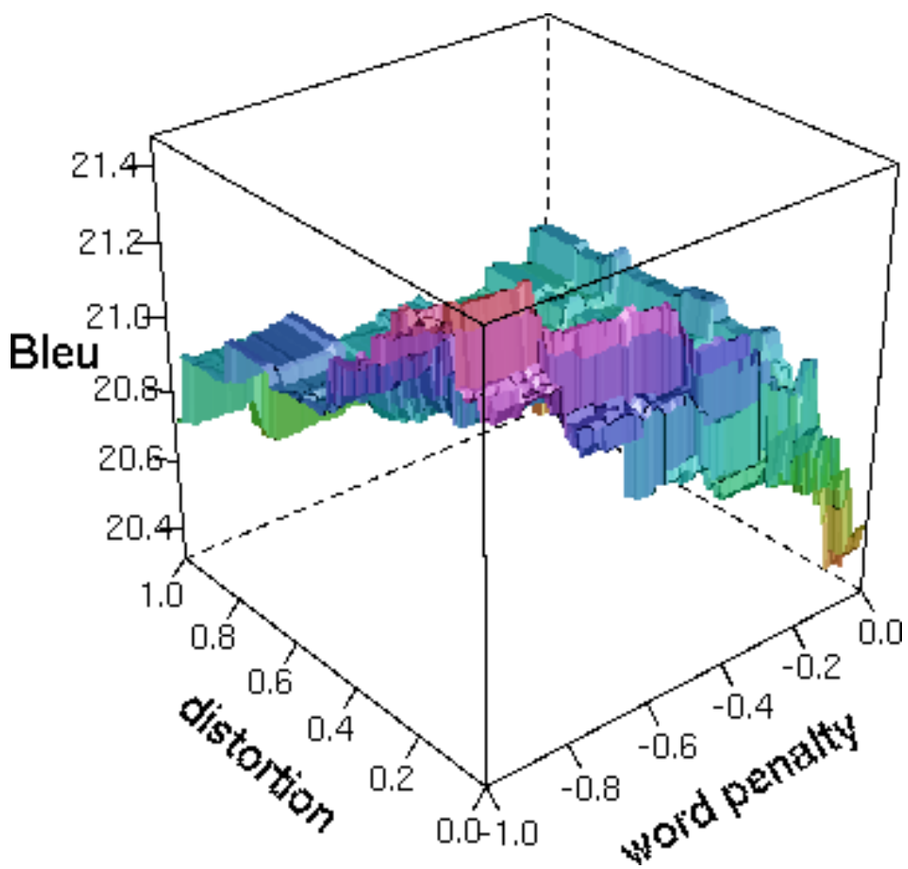
- Minimum Error Rate Training (MERT) (Och, 2003)

$$\theta^* = \arg\min_{\theta} \ \mathrm{Loss}(\hat{y},\ y^*)$$

$$-\ \mathrm{BLEU}(y, y^*)$$

(Papineni et al., 2001)

$$\hat{y} = \arg\max_{y} \ p_\theta(y \mid x)$$



(Smith and Eisner, 2006)

piecewise constant (not smoothed), not amenable to gradient descent, so cannot be scaled up to a large number of features

Minimum Risk

- Minimum Error Rate Training (MERT) (Och, 2003)

$$\theta^* = \arg\min_{\theta} \ \mathrm{Loss}(\hat{y},\ y^*)$$

$$-\ \mathrm{BLEU}(y, y^*)$$

(Papineni et al., 2001)

$$\hat{y} = \arg\max_{y} \ p_\theta(y \mid x)$$

- Minimum Risk Training

$$\theta^* = \arg\min_{\theta} \ \mathrm{Risk}(\theta, y^*)$$

$$= \arg\min_{\theta} \sum_{y \in \mathrm{HG}} p_\theta(y|x) \times \mathrm{Loss}(y, y^*)$$

risk = expected loss = expected error rate

Minimum Risk with Deterministic Annealing

- Minimum risk objective

$$\theta^* = \arg\min_{\theta} \mathrm{Risk}(\theta, y^*)$$

suffer from local-minimum problem

- Minimum risk with deterministic annealing

$$\theta^* = \arg\min_{\theta} \mathrm{Risk}(\theta, y^*) - \mathrm{Temperature} \times \mathrm{Entropy}(p_\theta)$$

Smith and Eisner (2006) tried this on an **n-best list**

Conditional Random Field (CRF)

- Minimum risk objective

$$\theta^* = \arg\min_{\theta} \sum_{y \in \mathrm{HG}} p_\theta(y|x) \times \mathrm{Loss}(y, y^*)$$

- Conditional Random Field (CRF)
  - or maximum conditional likelihood (MLE)

$$\theta^* = \arg\max_{\theta} p_\theta(y^*|x)$$

CRF is minimum-risk training with a **zero-one** loss function!

$$\mathrm{Loss}(y, y^*) = \begin{cases} 0 & \text{if } y = y^* \\ 1 & \text{otherwise.} \end{cases}$$

no partial credit ☹

$$\mathrm{Risk}(\theta, y^*) = \sum_{y \in \mathrm{HG} \ \& \ y != y*} p_\theta(y|x) = 1 - p_\theta(y^*|x)$$

Perceptron

- Not explicitly optimize a certain objective function, rather it is a simple procedure

Perceptron$(x, \mathrm{GEN}(x), y)$

$\theta \leftarrow \vec{0}$ $\quad\triangleright$ initialize as zero vector
for $t \leftarrow 1$ **to** T
  for $i \leftarrow 1$ **to** N
    $\hat{y} \leftarrow \arg\max_{y \in \mathrm{GEN}(x_i)} \Phi(x_i, y) \cdot \theta$
    if $(\hat{y} \neq y_i^*)$
      $\theta \leftarrow \theta + \Phi(x_i, y_i^*) - \Phi(x_i, \hat{y})$
return θ

See Li and Khudanpur (2009) for its application in MT

Why Minimum Risk Training?

	Min-Risk	MERT	CRF	Perceptron	MIRA
Scalability	☺	☹	☺	☺	☺
BLEU	☺	☺	☹	☹	☺
Latent variable	☺	☹	☺	☹	☹
Oracle translation	☺	☺	☹	☹	☹
Model regularization	☺	☹	☺	☺	☺

See Zens et al. (2007) for experimental comparisons.

Experimental Results

- Data set: IWSLT CN-EN 2005

	Training scheme	BLEU
(Och, 2003)	MERT (Nbest, small)	47.7
(Smith and Eisner, 2006)	MR (Nbest, small)	47.7
New!	MR (hypergraph, small)	48.4
New!	MR (hypergraph, large)	**48.7**

Small: discriminatively tune 5 features

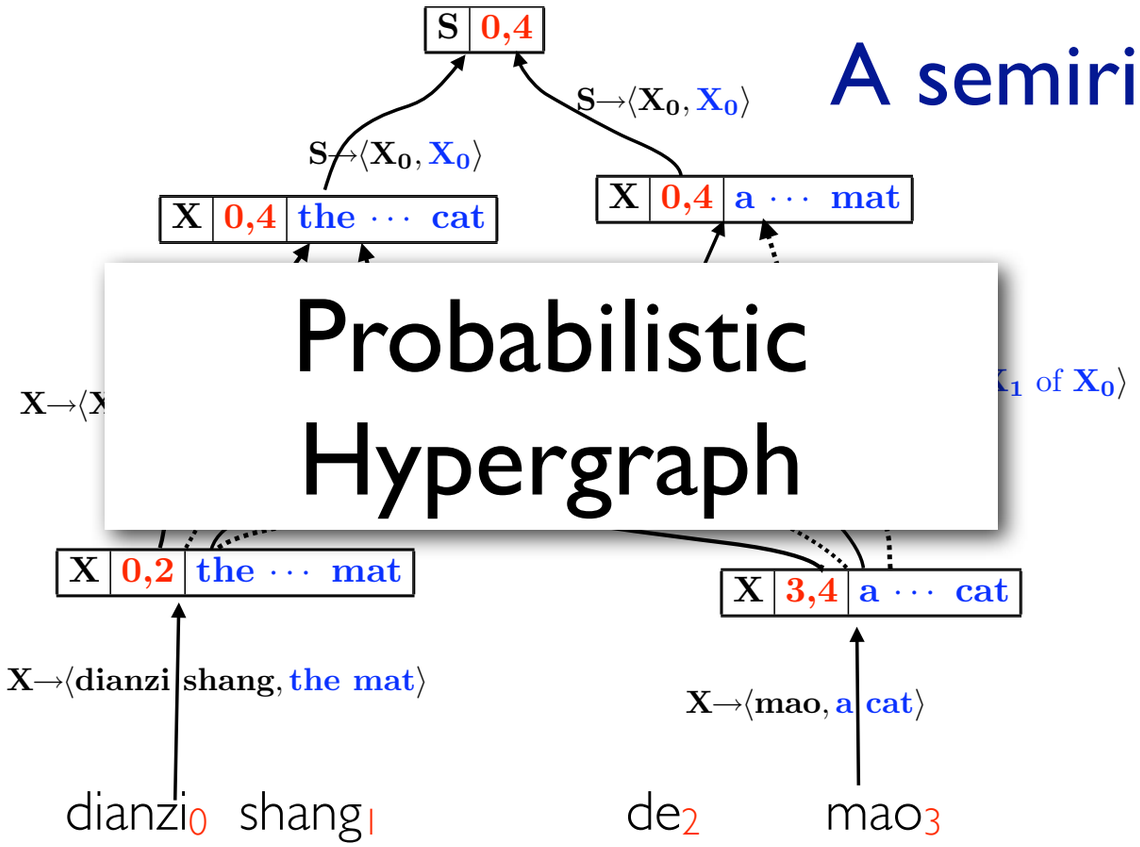
Large: also discriminatively tune the language model (**21k** additional features)

See Zens et al. (2007) for experimental comparisons on nbest for small number of features.

Outline

- Hypergraph as hypothesis space
- Training methods (Li and Eisner, 2009)
- Exact and approximate inference
  - exact inference
    - ★ semiring framework (Li and Eisner, 2009)
  - approximate inference
    - ★ "principled" ones
      - ✓ variational decoding (Li et. al, 2009)
      - ✓ sampling and message passing methods
    - ★ heuristic-based ones
      - ✓ cube-pruning (Chiang, 2007)
- Joshua project (Li et. al, 2009)

A semiring framework to compute all of these



Probabilistic Hypergraph

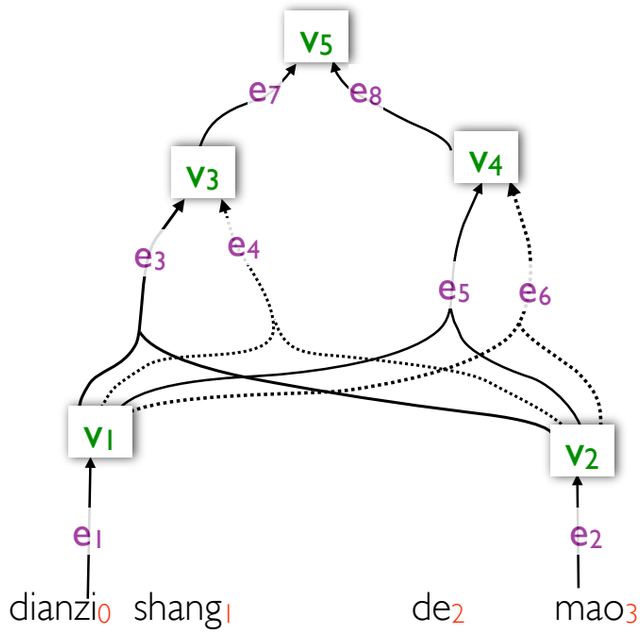
- "decoding" quantities:
  - Viterbi
  - K-best
  - Counting
  - .....

- First-order quantities:
  - expectation
    - entropy
    - Bayes risk
    - cross-entropy
    - KL divergence
    - feature expectations
  - first-order gradient of Z

- Second-order quantities:
  - Expectation over product
    - interaction between features
  - Hessian matrix of Z
    - second-order gradient descent
  - gradient of expectation
    - gradient of entropy or Bayes risk

Compute Quantities on a Hypergraph: a Recipe

- Semiring-weighted inside algorithm
  - three steps:
    - choose a semiring



$$\langle K, \oplus, \otimes \rangle$$

a **set** with **plus** and **times** operations

e.g., integer numbers with regular + and ×

    - specify a weight for each hyperedge

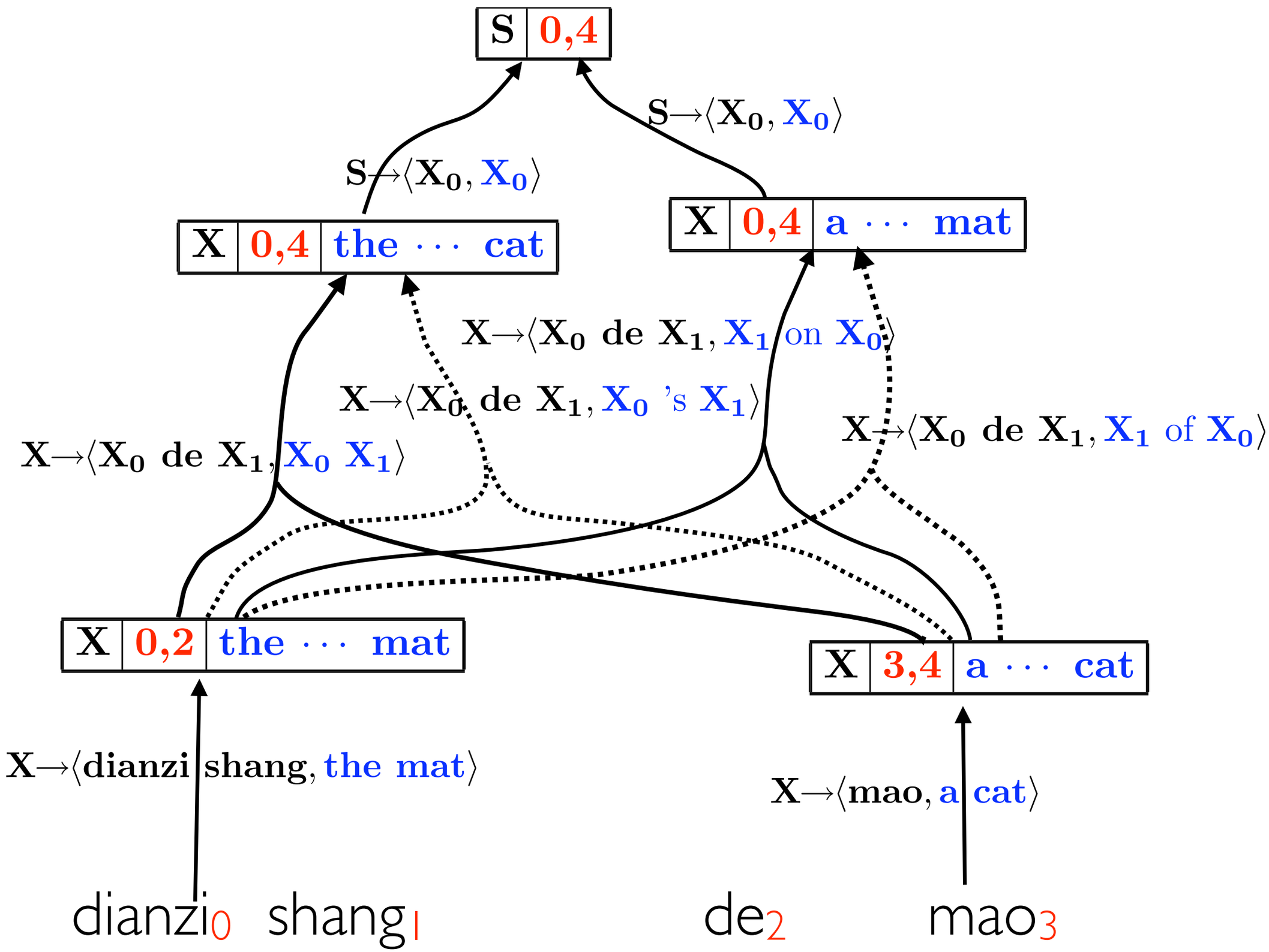
      each weight is a semiring member

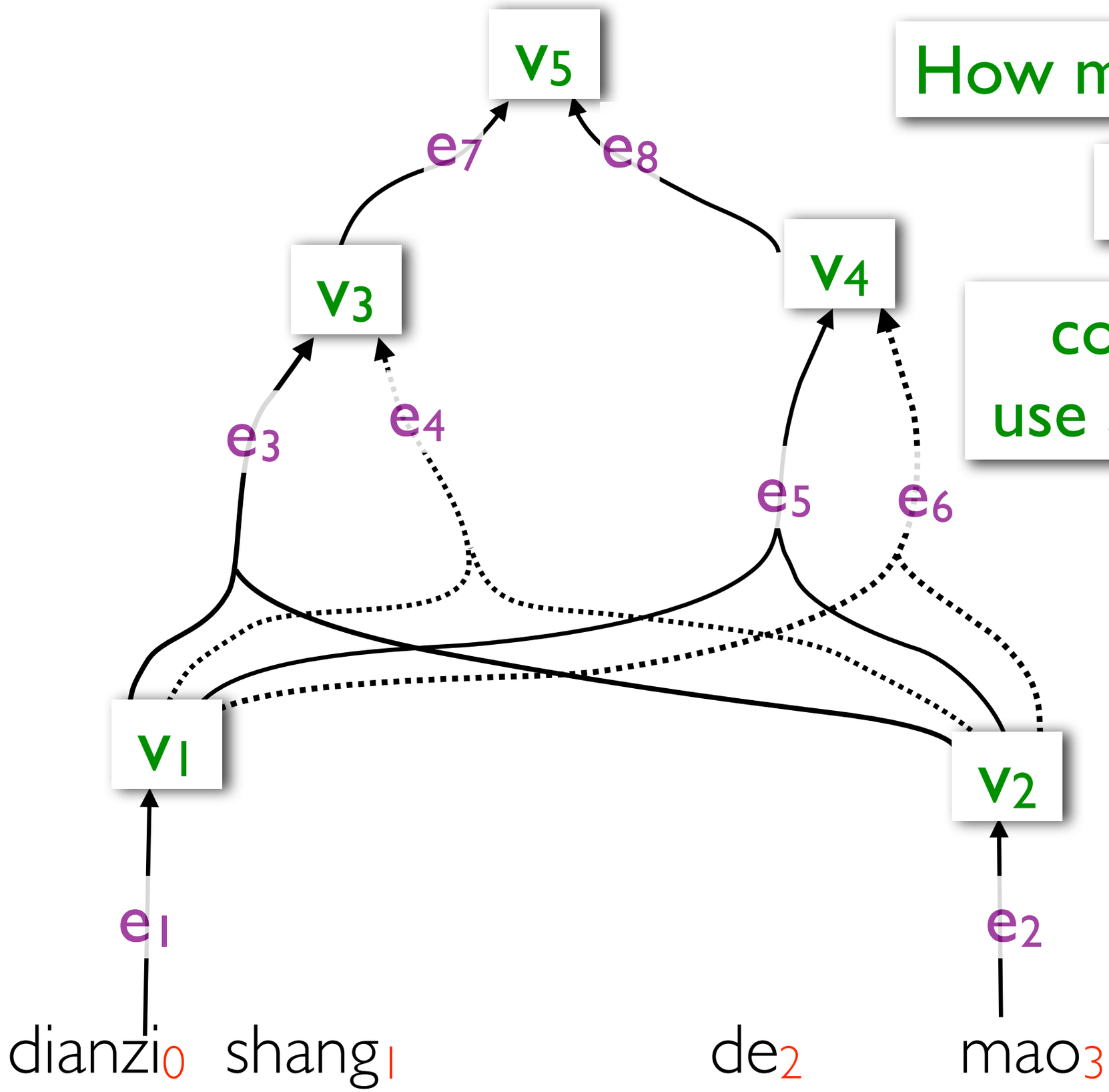
    - run the inside algorithm

      complexity is O(hypergraph size)

Semirings

- "Decoding" time semirings (Goodman, 1999)
  - counting, Viterbi, K-best, etc.
- "Training" time semirings
  - first-order expectation semirings (Eisner, 2002)
  - second-order expectation semirings (new)
- Applications of the Semirings (new)
  - entropy, risk, gradient of them, and many more


S 0,4
S→⟨X0, X0⟩
S→⟨X0, X0⟩
X 0,4 the ··· cat
X 0,4 a ··· mat
X→⟨X0 de X1, X1 on X0⟩
X→⟨X0 de X1, X0 's X1⟩
X→⟨X0 de X1, X1 of X0⟩
X→⟨X0 de X1, X0 X1⟩
X 0,2 the ··· mat
X 3,4 a ··· cat
X→⟨dianzi shang, the mat⟩
X→⟨mao, a cat⟩
dianzi0 shang1 de2 mao3



How many trees?

four ☺

compute it
use a semiring?

Compute the Number of Derivation Trees

Three steps:

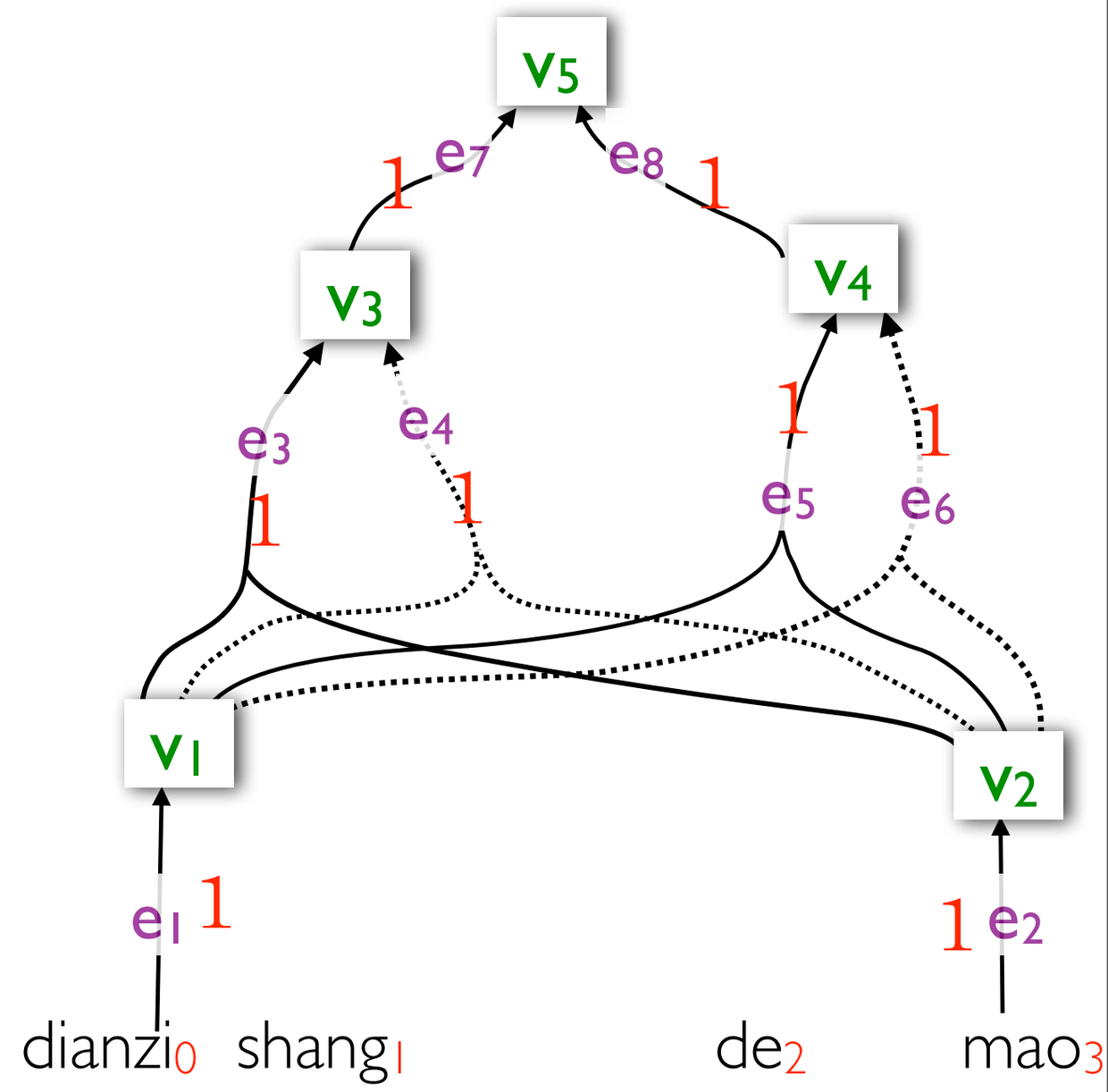
- choose a semiring

  counting semiring:

  ordinary integers with regular + and x

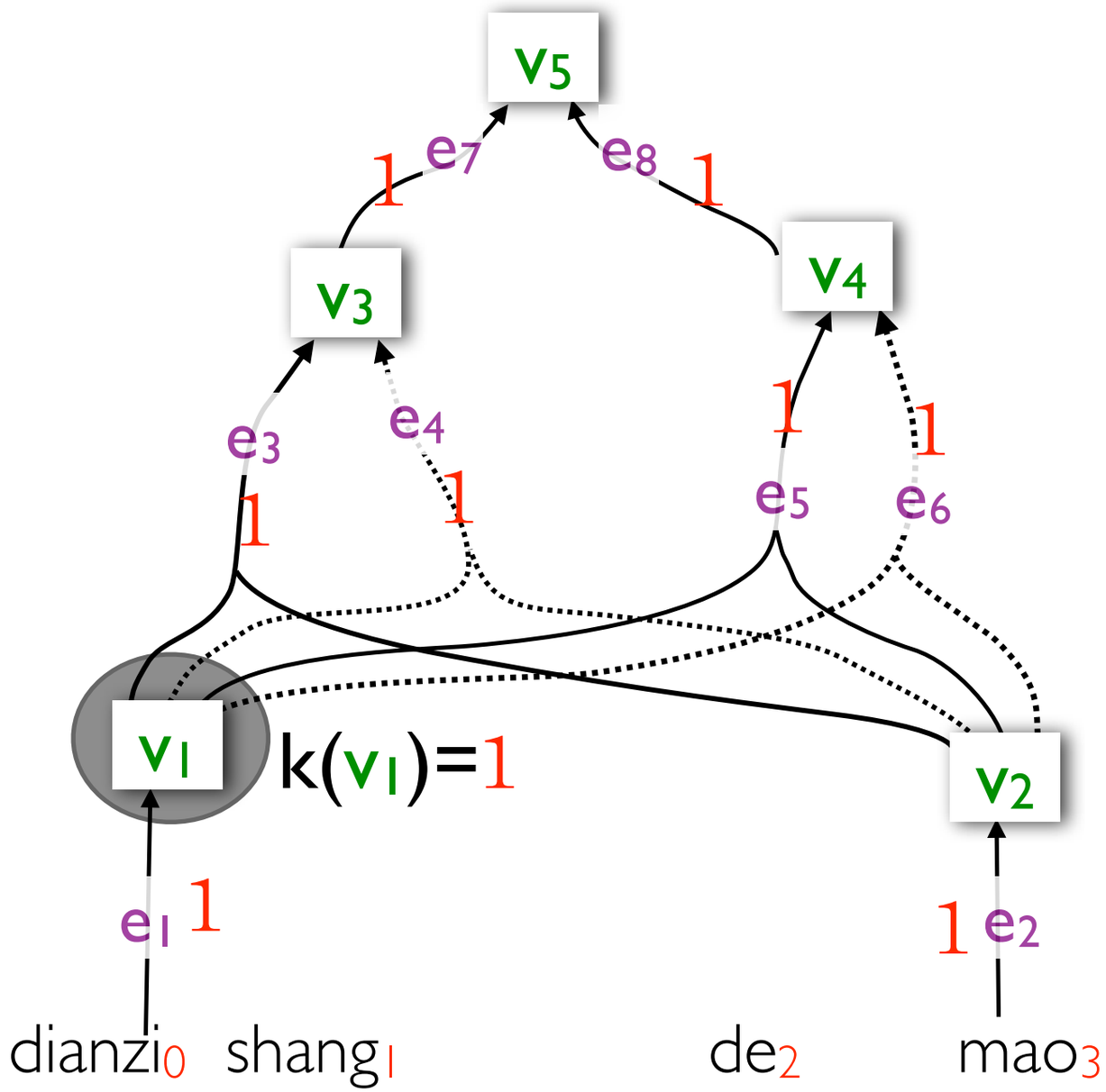
- specify a weight for each hyperedge

- run the inside algorithm



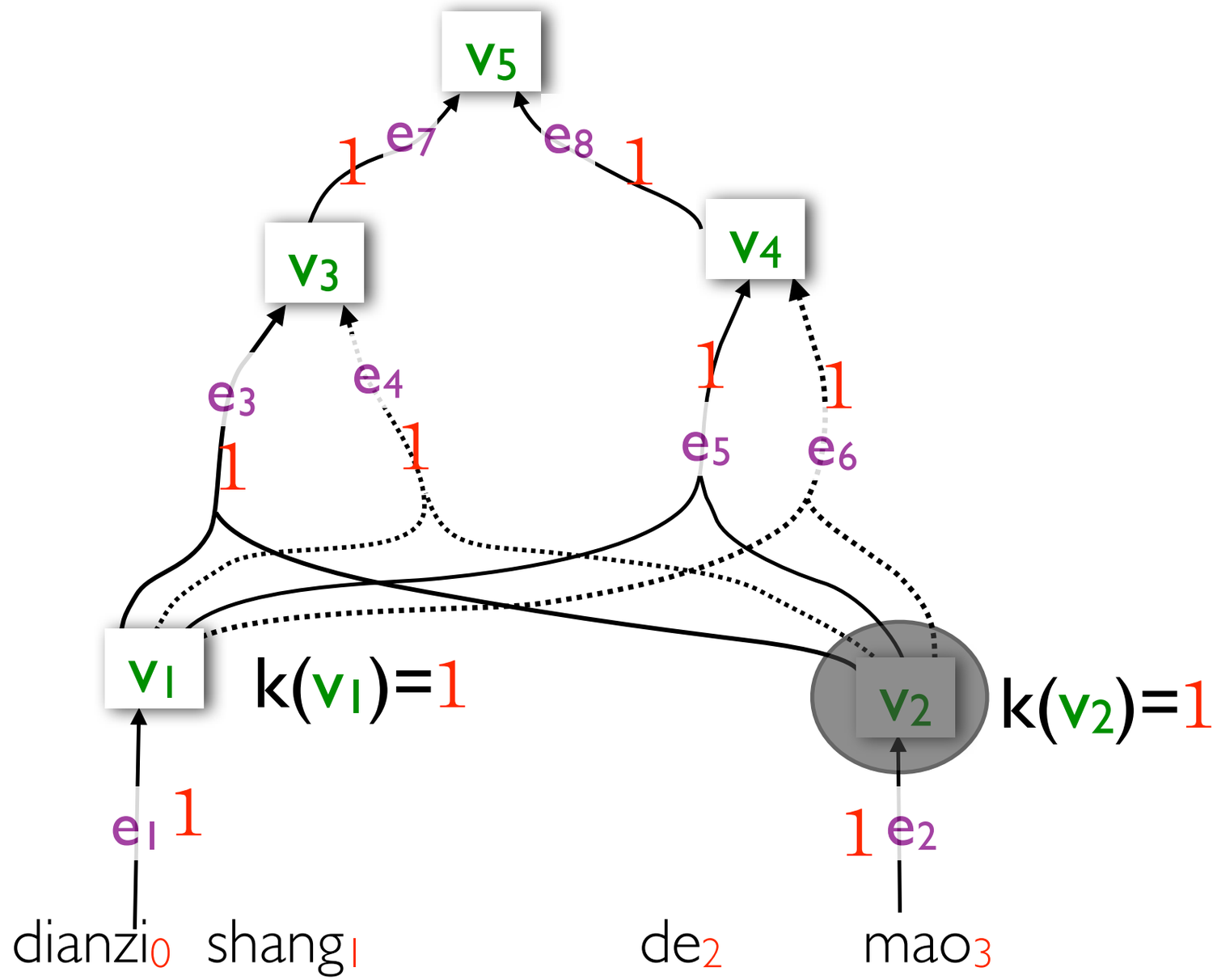
$k(v_1) = k(e_1)$

Bottom-up process in computing the number of trees



$k(v_1) = k(e_1)$ $k(v_2) = k(e_2)$

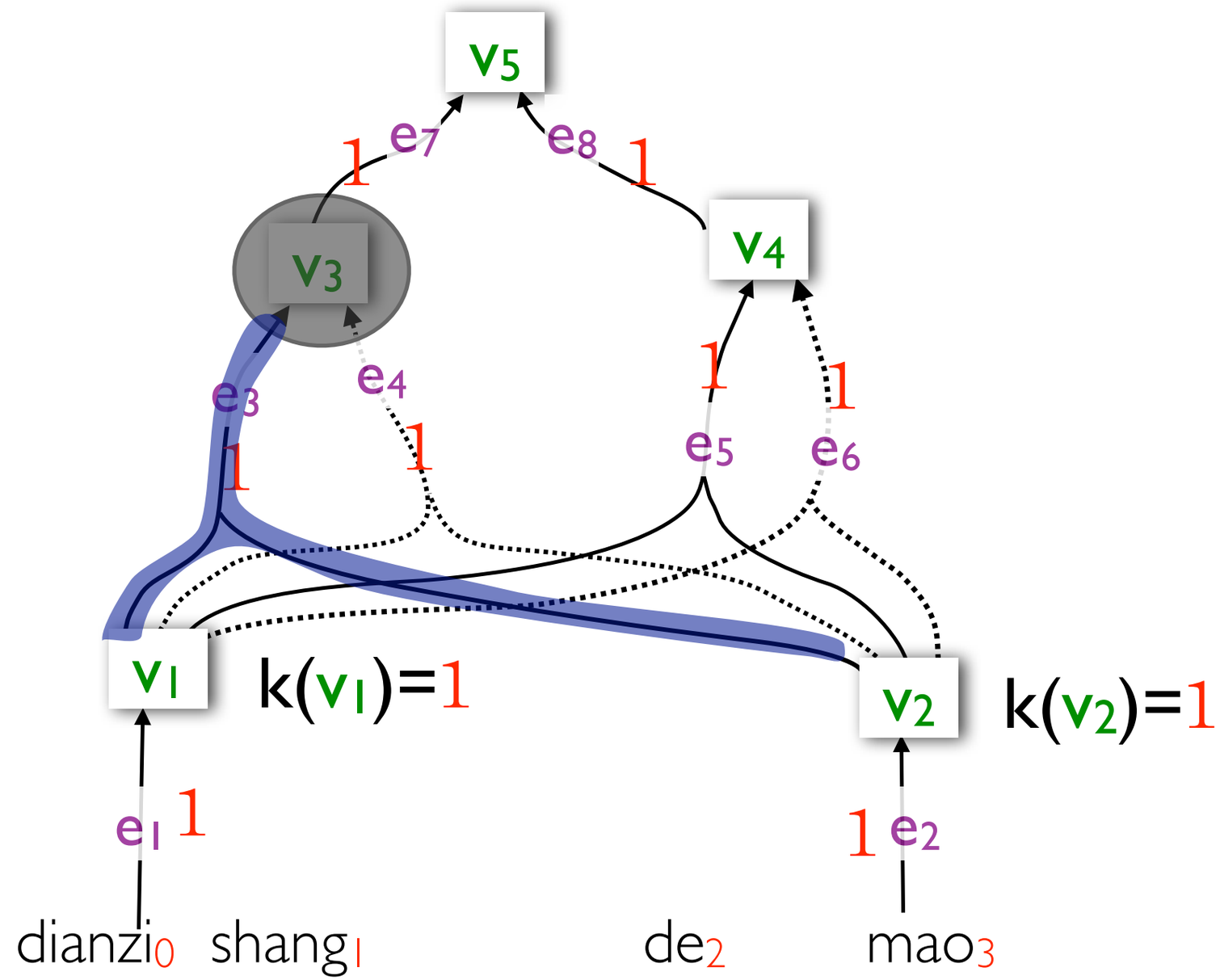
Bottom-up
process in computing the number of trees



Compute $k(v_3)$: the weight at node v_3

Hyperedge e_3: $k(e_3) \otimes k(v_1) \otimes k(v_2) = 1 \otimes 1 \otimes 1 = 1$

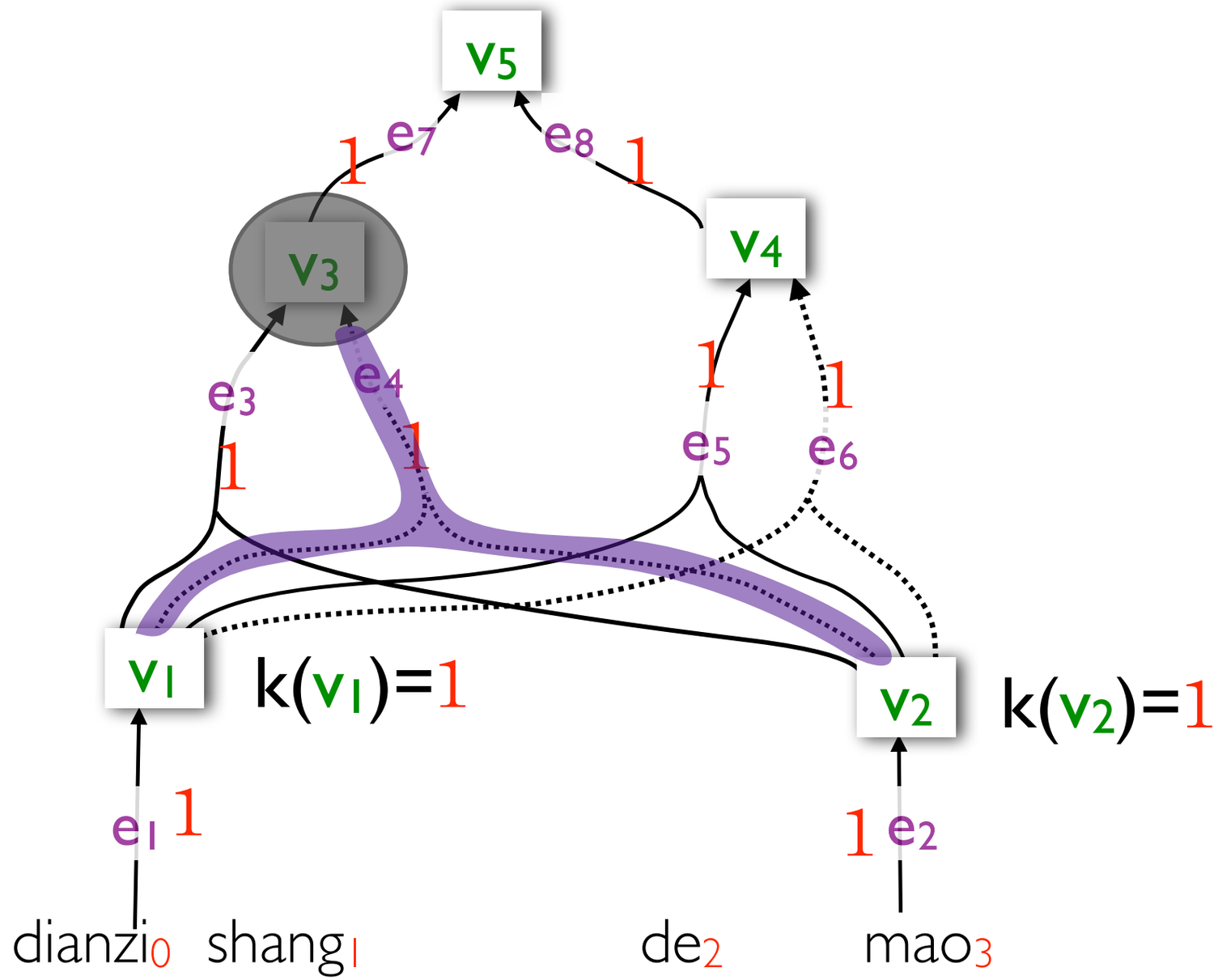
Bottom-up process in computing the number of trees



Compute $k(v_3)$: the weight at node v_3

Hyperedge e_4: $k(e_4) \otimes k(v_1) \otimes k(v_2) = 1 \otimes 1 \otimes 1 = 1$

Bottom-up process in computing the number of trees

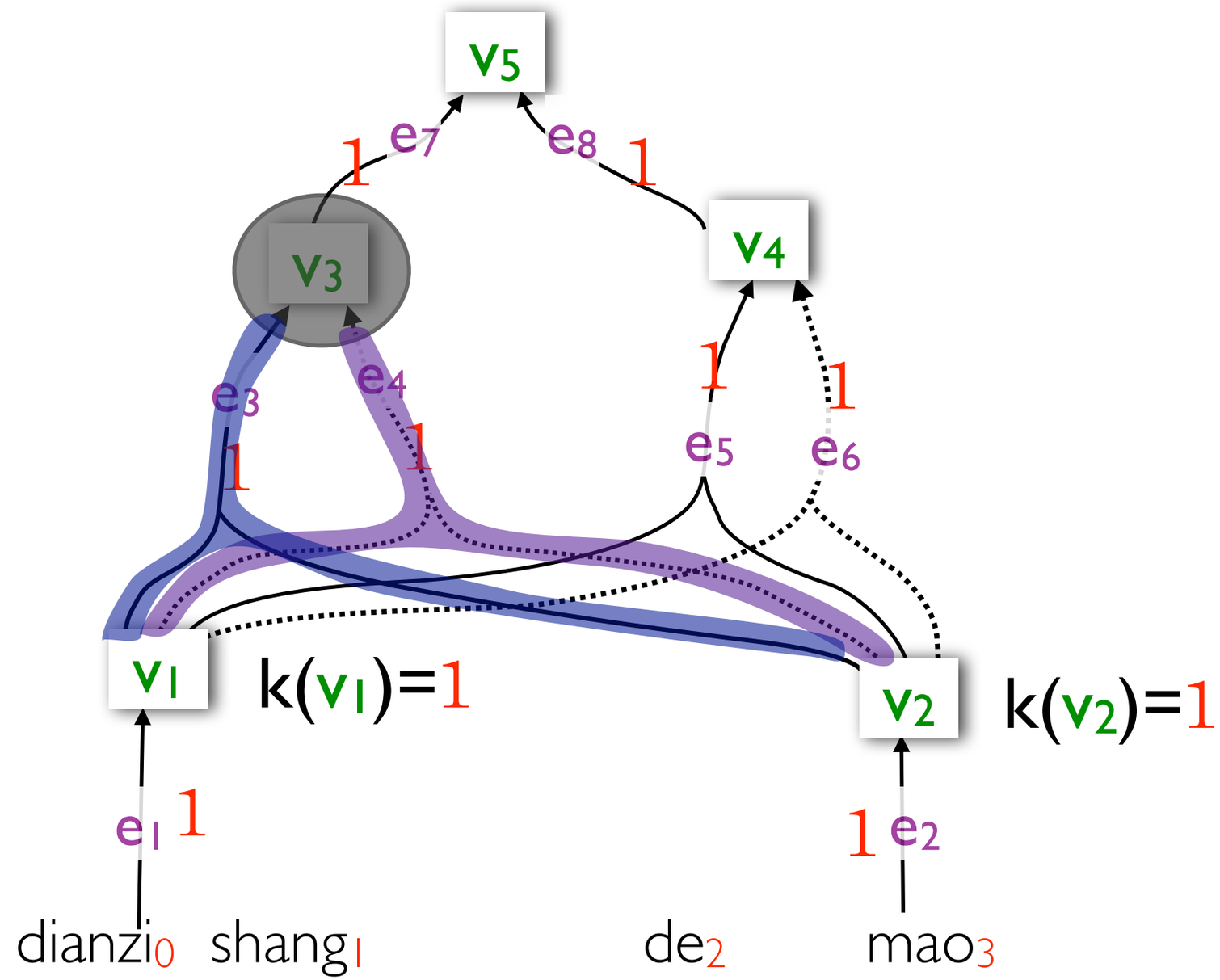


Compute $k(v_3)$: the weight at node v_3

Hyperedge e_3: $k(e_3) \otimes k(v_1) \otimes k(v_2) = 1$

Hyperedge e_4: $k(e_4) \otimes k(v_1) \otimes k(v_2) = 1$

Bottom-up process in computing the number of trees

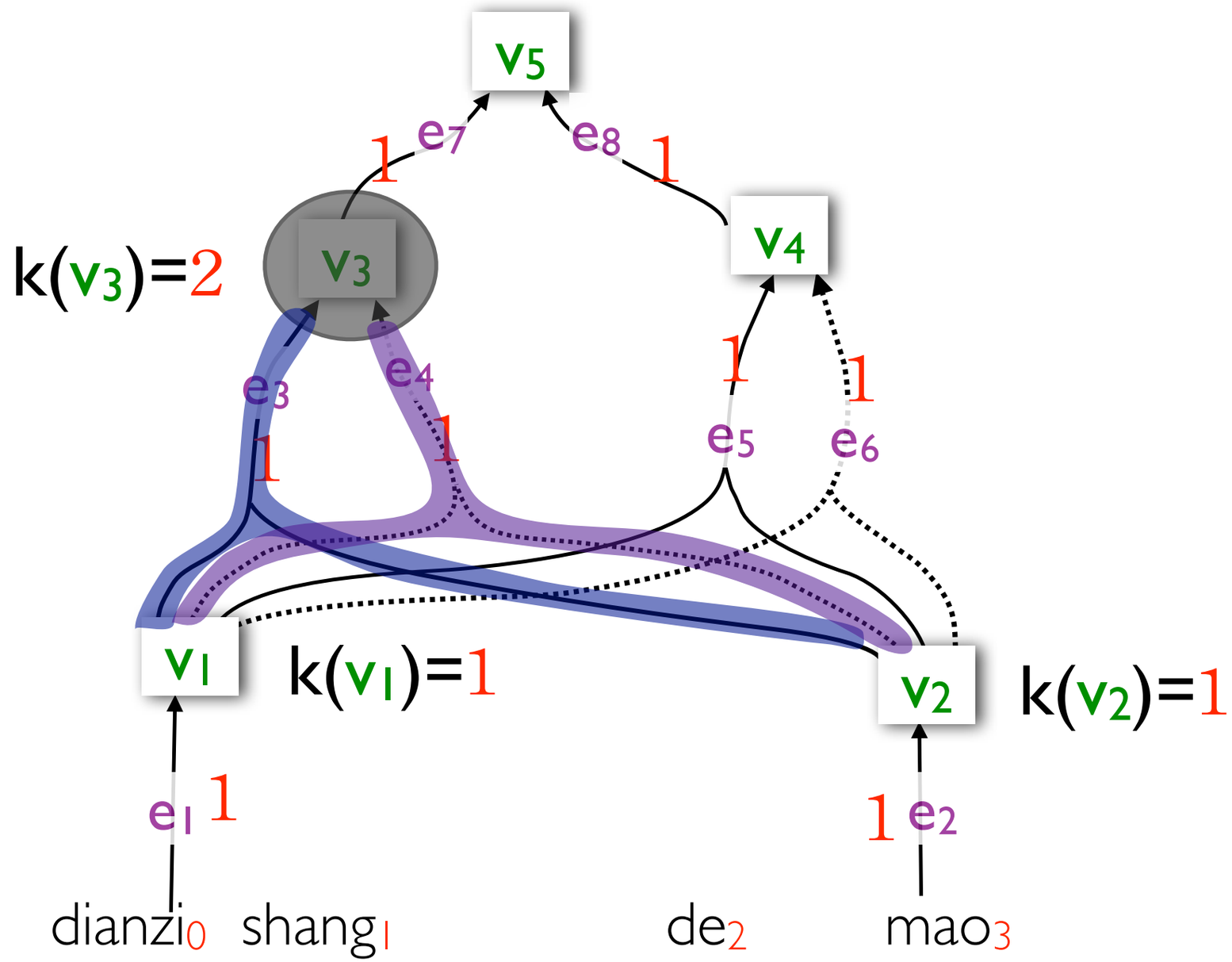


Compute $k(v_3)$: the weight at node v_3

$$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$$

$$1 \oplus 1 = 2$$

Bottom-up process in computing the number of trees

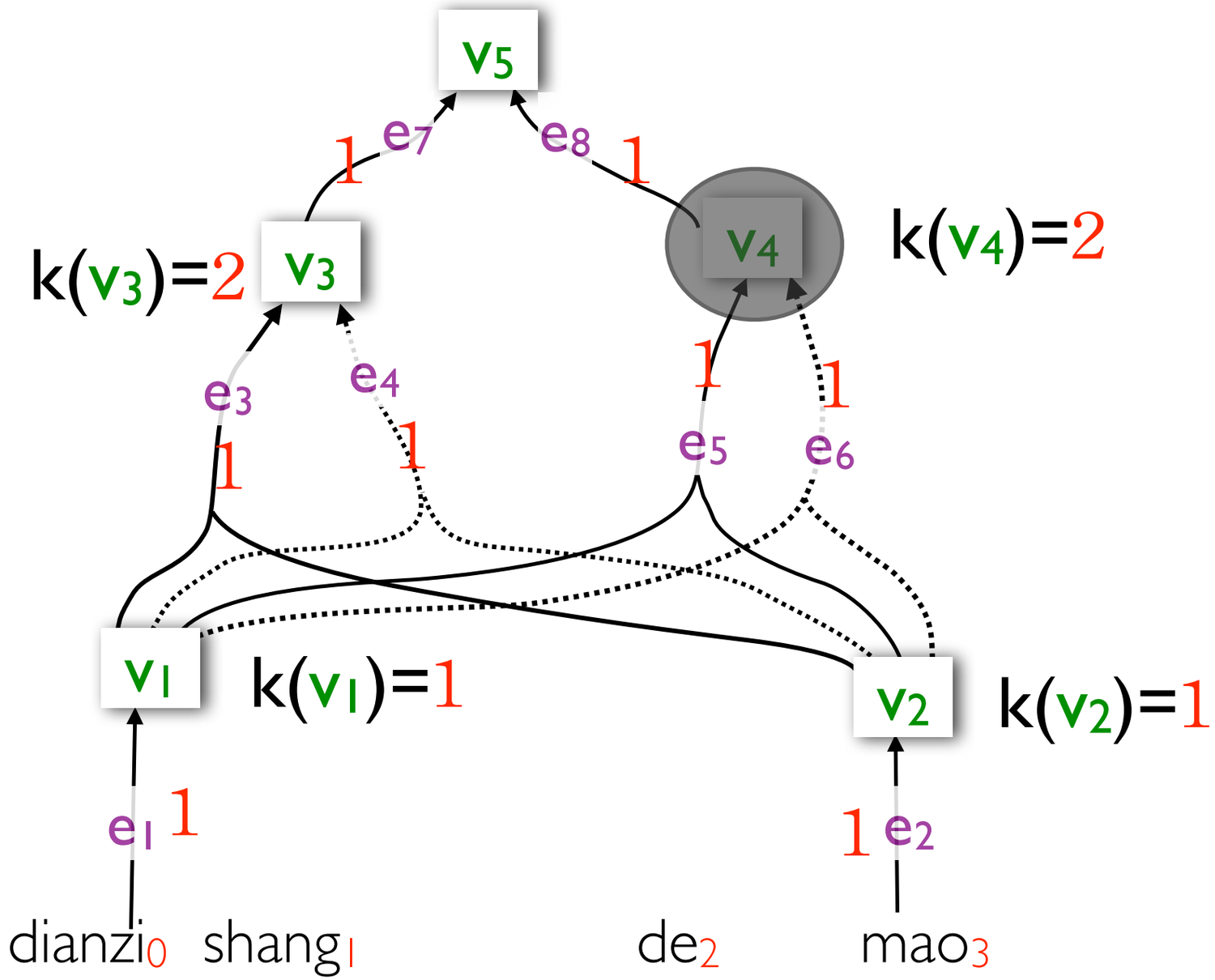


$k(v_1) = k(e_1)$ $\qquad$ $k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

Bottom-up process in computing the number of trees



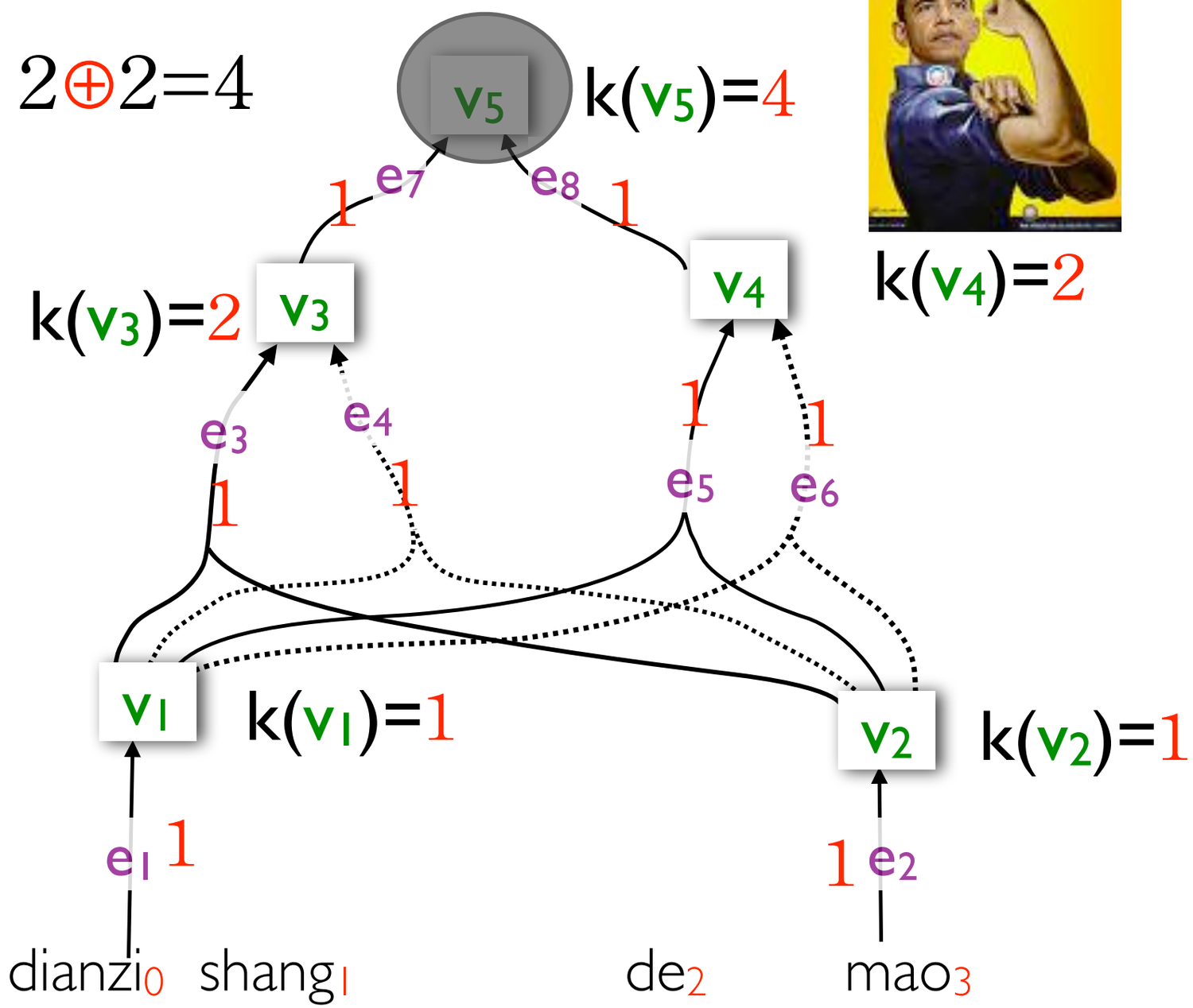
$k(v_1) = k(e_1)$ $\qquad$ $k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$

$2 \oplus 2 = 4$



Bottom-up process in computing the number of trees

$k(v_1) = k(e_1)$ $\qquad$ $k(v_2) = k(e_2)$

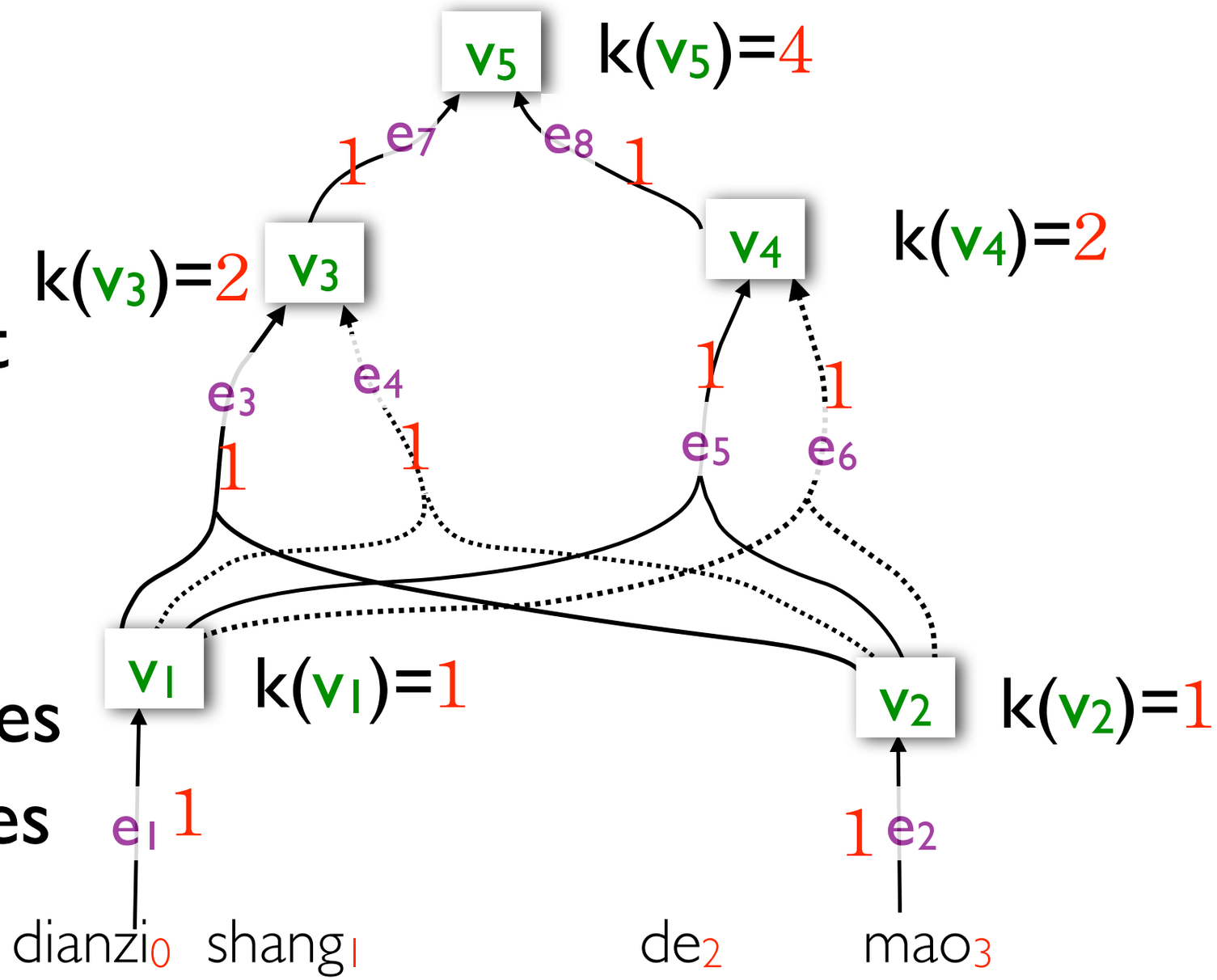
$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

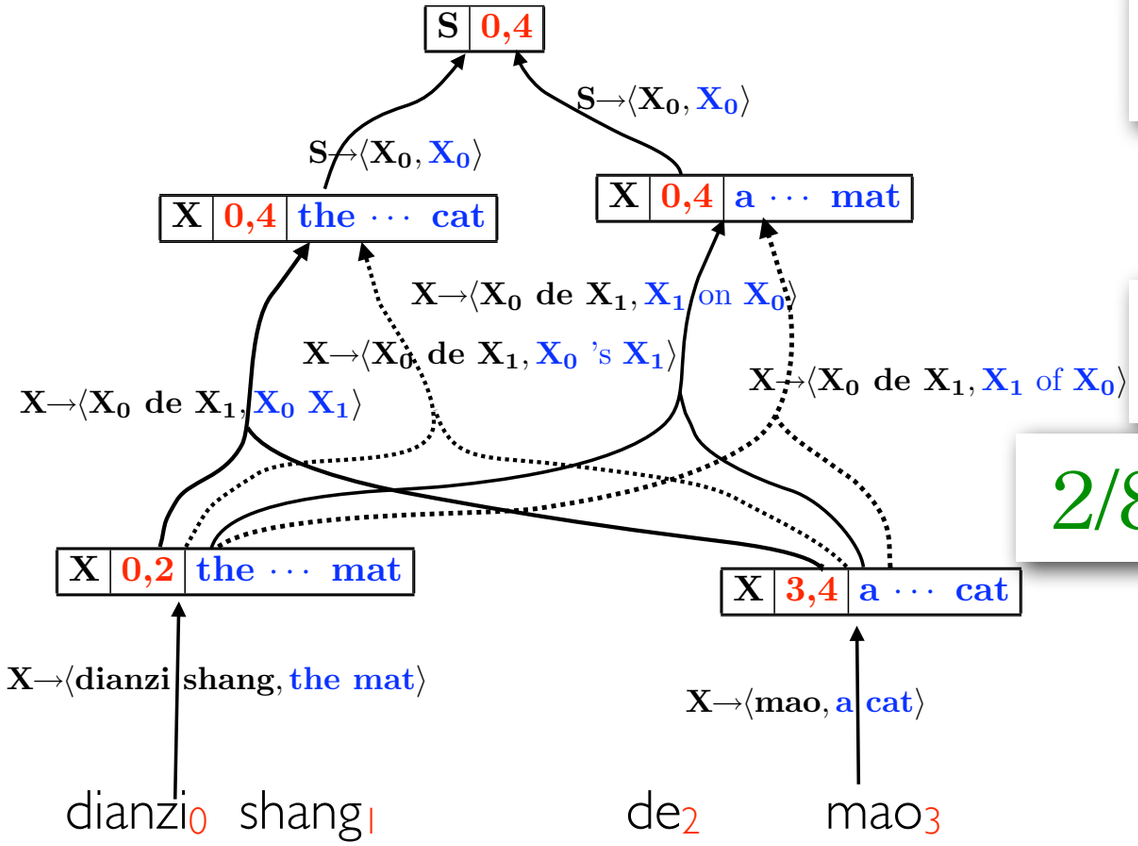
$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$

Summary:

- input: a weight at each edge
- output: a weight at each node
- $\oplus$ is used at nodes
- $\otimes$ is used at edges





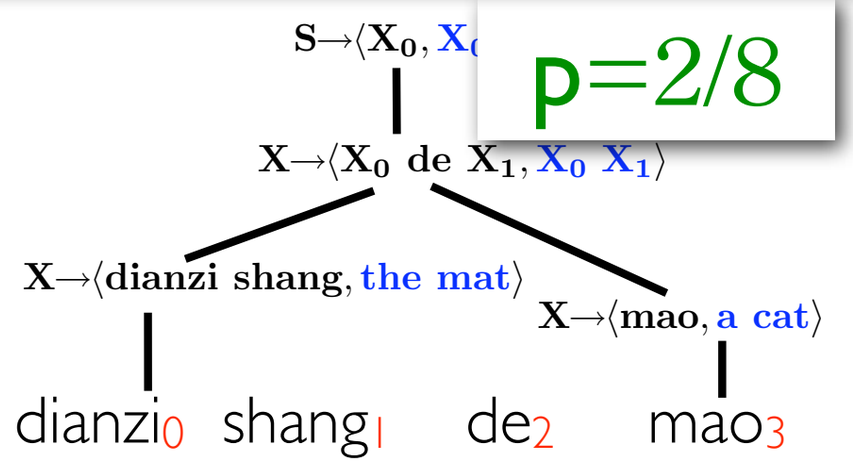
expected translation length?

$2/8 \times 4 + 6/8 \times 5 = 4.75$

variance?

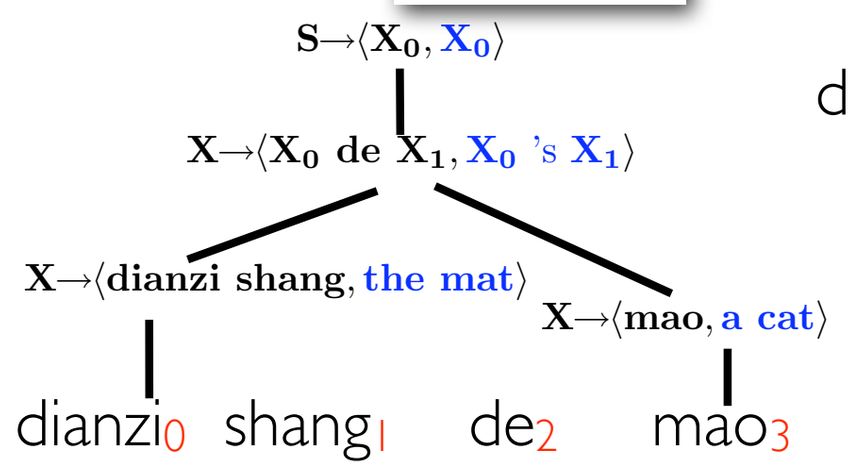
$2/8 \times (4-4.75)^2 + 6/8 \times (5-4.75)^2 \approx 0.19$

p=2/8



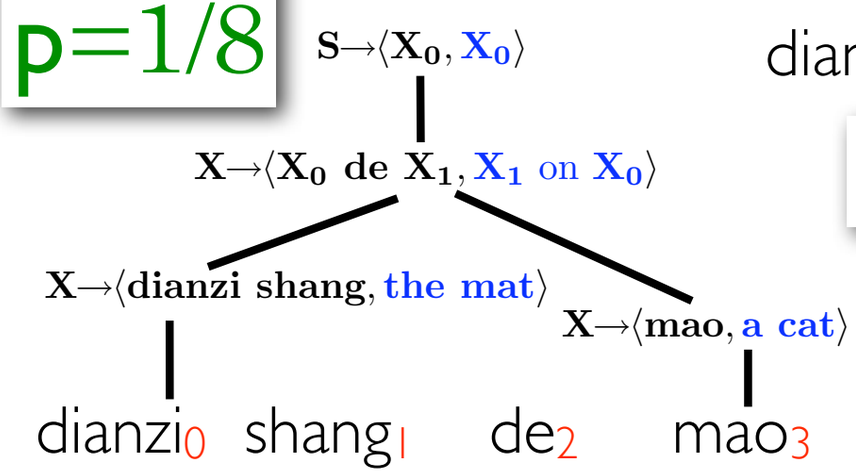
the mat a cat 4

p=3/8



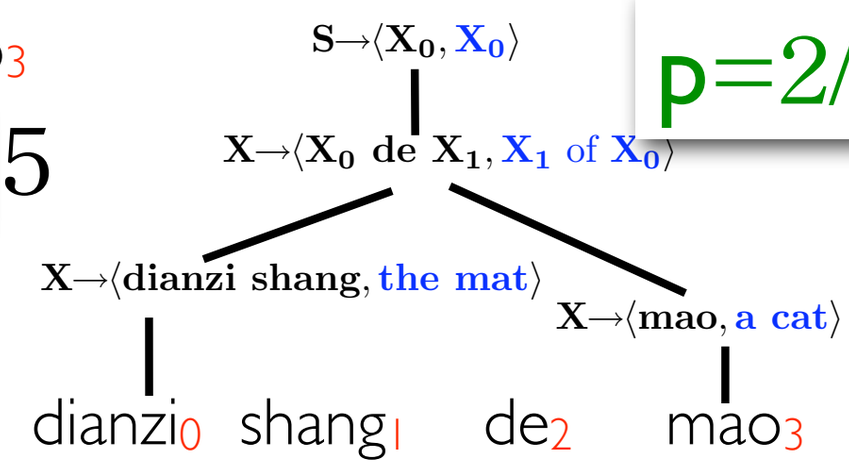
the mat 's a cat 5

p=1/8



a cat on the mat 5

p=2/8



a cat of the mat 5

First- and Second-order Expectation Semirings

First-order:

(Eisner, 2002)

- each member is a 2-tuple: $\langle p, r \rangle$

$\langle p_1, r_1 \rangle \otimes \langle p_2, r_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1 \rangle$
$\langle p_1, r_1 \rangle \oplus \langle p_2, r_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2 \rangle$

Second-order:

- each member is a 4-tuple: $\langle p, r, s, t \rangle$

$\langle p_1, r_1, s_1, t_1 \rangle \otimes \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 p_2,\ p_1 r_2 + p_2 r_1,\ p_1 s_2 + p_2 s_1,$ $p_1 t_2 + p_2 t_1 + r_1 s_2 + r_2 s_1 \rangle$
$\langle p_1, r_1, s_1, t_1 \rangle \oplus \langle p_2, r_2, s_2, t_2 \rangle$	$\langle p_1 + p_2,\ r_1 + r_2,\ s_1 + s_2,\ t_1 + t_2 \rangle$

Compute Expected Translation Length

- choose a semiring

  first-order expectation semiring

- specify a weight for each hyperedge

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e: transition probability or log-linear score at edge e

r_e: number of English words generated at edge e

- run the inside algorithm

Compute Variance in Translation Length

- choose a semiring

  second-order expectation semiring

- specify a weight for each edge

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$$

p_e: transition probability or log-linear score at edge e

$r_e = s_e$: number of English words generated at edge e

- run the inside algorithm

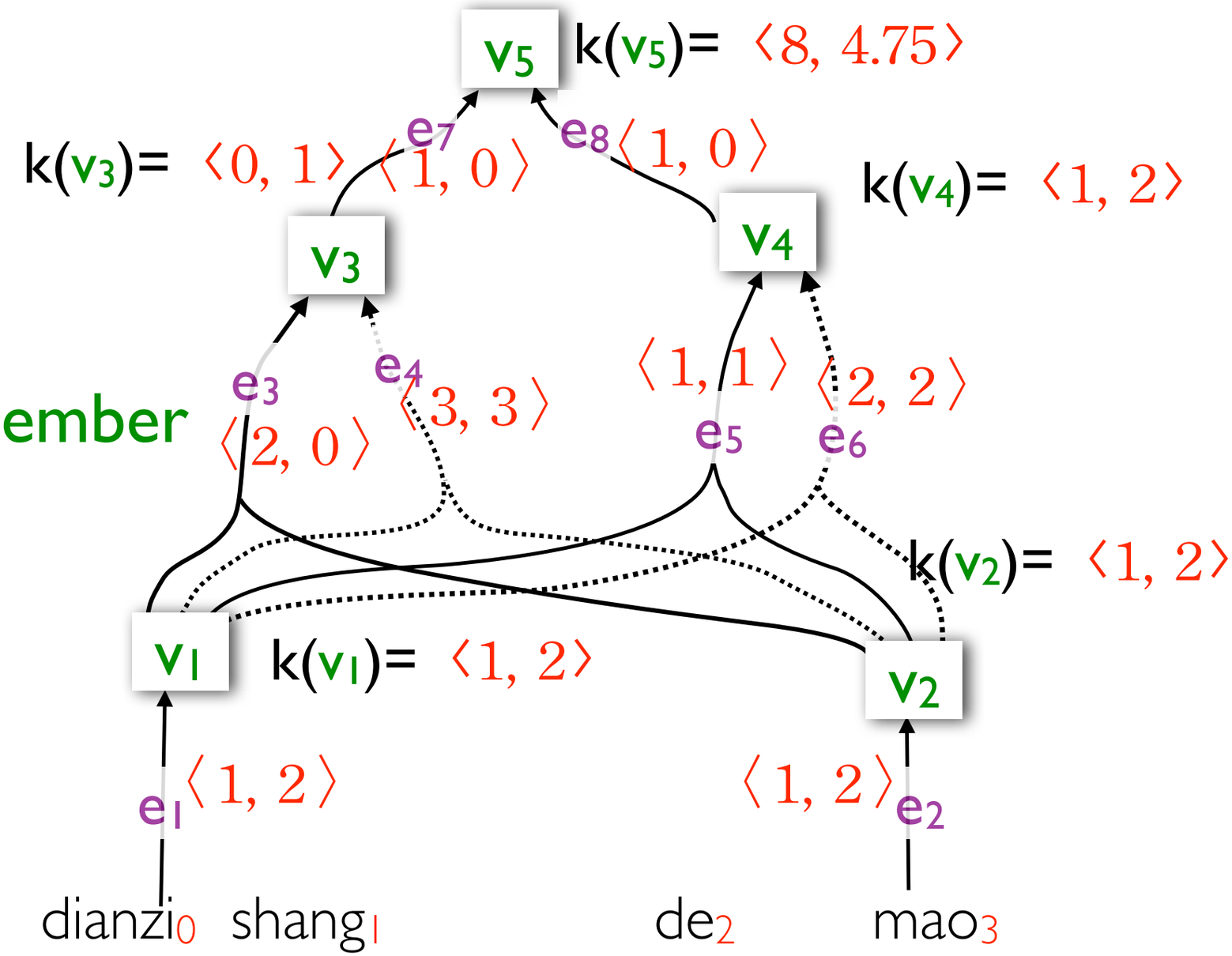
$k(v_1) = k(e_1)$ $\qquad$ $k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$

First-order:
each semiring member
is a **2-tuple**



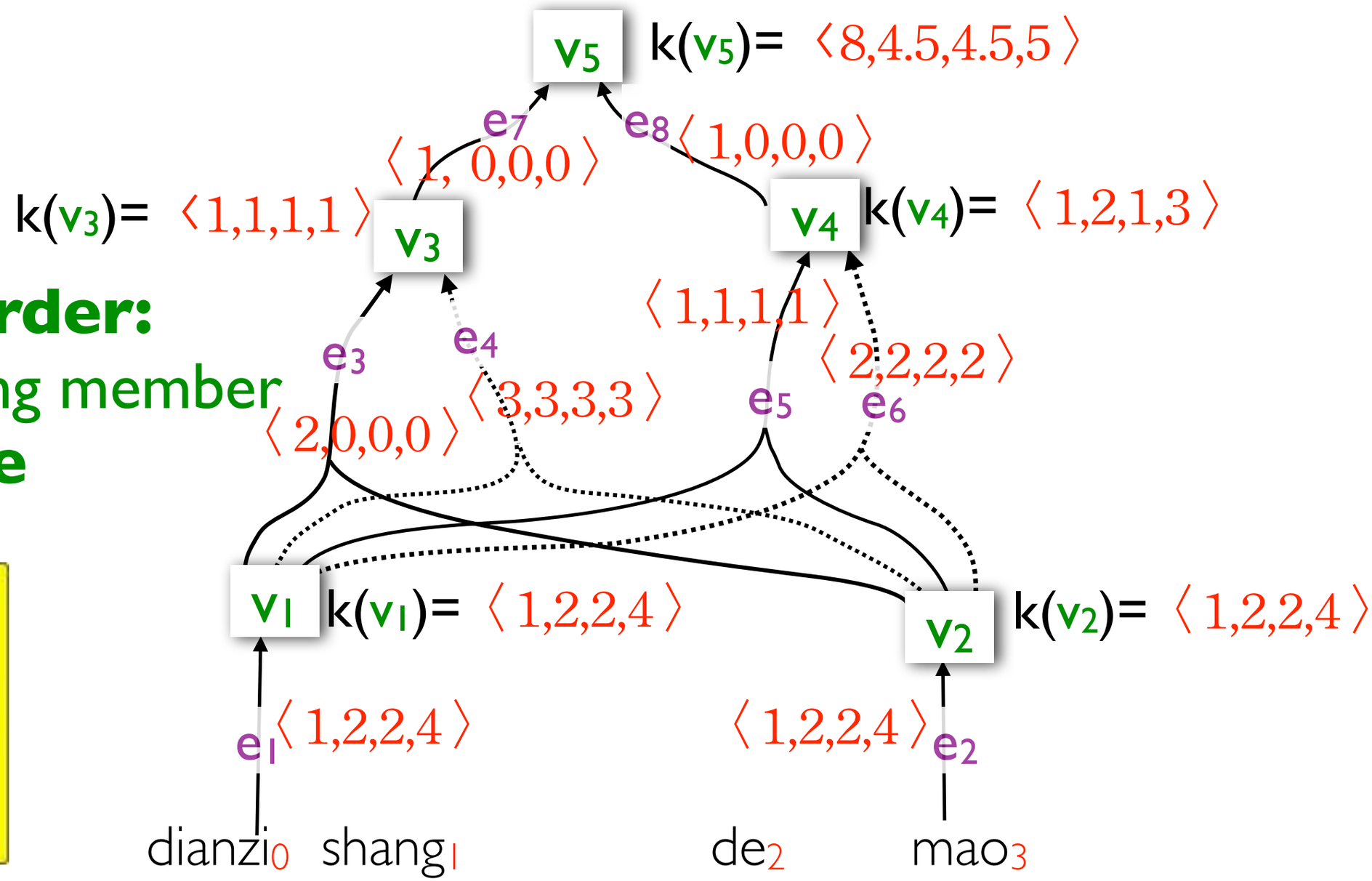
$k(v_1) = k(e_1)$ $\qquad$ $k(v_2) = k(e_2)$

$k(v_3) = k(e_3) \otimes k(v_1) \otimes k(v_2) \oplus k(e_4) \otimes k(v_1) \otimes k(v_2)$

$k(v_4) = k(e_5) \otimes k(v_1) \otimes k(v_2) \oplus k(e_6) \otimes k(v_1) \otimes k(v_2)$

$k(v_5) = k(e_7) \otimes k(v_3) \oplus k(e_8) \otimes k(v_4)$

Second-order:
each semiring member
is a **4-tuple**



Expectations on Hypergraphs

- Expectation over a hypergraph

$$\overline{r} \stackrel{\text{def}}{=} \mathbb{E}_p[r] = \sum_{d \in \mathrm{HG}} p(d) r(d)$$

exponential size

  - *r*(*d*) is a function over a derivation *d*

    e.g., the length of the translation yielded by *d*

  - *r*(*d*) is additively decomposed

$$r(d) \stackrel{\text{def}}{=} \sum_{e \in d} r_e$$

e.g., translation length is additively decomposed!

Second-order Expectations on Hypergraphs

- Expectation of **products** over a hypergraph

$$\bar{t} \stackrel{\text{def}}{=} \mathbb{E}_p[r \cdot s] = \sum_{d \in \mathrm{HG}} p(d) r(d) s(d)$$

exponential size

- r and s are additively decomposed

$$r(d) \stackrel{\text{def}}{=} \sum_{e \in d} r_e$$

$$s(d) \stackrel{\text{def}}{=} \sum_{e \in d} s_e$$

r and s can be identical or different functions.

Compute expectation using expectation semiring:

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e : transition probability or log-linear score at edge e

r_e ?

Entropy:

$$r_e \stackrel{\text{def}}{=} \log p_e$$

Why?

entropy is an **expectation**

$$\mathrm{H}(p) = \mathbb{E}_p[-\log p] = -\sum_{d \in \mathrm{HG}} p(d) \log p(d)$$

log $p(d)$ is additively decomposed!

Compute expectation using expectation semiring:

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e : transition probability or log-linear score at edge e

r_e ?

Entropy: $$r_e \stackrel{\text{def}}{=} \log p_e$$

Cross-entropy: $$r_e \stackrel{\text{def}}{=} \log q_e$$

Why?

cross-entropy is an **expectation**

$$\mathrm{H}(p, q) = \mathbb{E}_p(-\log q) = -\sum_{d \in \mathrm{HG}} p(d) \log q(d)$$

log $q(d)$ is additively decomposed!

Compute expectation using expectation semiring:

$$k_e \stackrel{\text{def}}{=} \langle p_e, p_e r_e \rangle$$

p_e : transition probability or log-linear score at edge e

r_e ?

Entropy: $r_e \stackrel{\text{def}}{=} \log p_e$

Cross-entropy: $r_e \stackrel{\text{def}}{=} \log q_e$

Bayes risk: $r_e \stackrel{\text{def}}{=} \text{loss at edge } e$

Why? Bayes risk is an **expectation**

$$\text{Risk} = \mathbb{E}_p(L) = -\sum_{d \in \text{HG}} p(d) \cdot L(Y(d))$$

L(*Y*(*d*)) is additively decomposed! (Tromble et al. 2008)

Applications of expectation semirings: a summary

First-order:

Quantity	Weight for edge e	Value at root
Expectation	$\langle p_e, p_e r_e \rangle$	$\langle Z, \overline{r} \rangle$
First-order gradient	$\langle p_e, \nabla p_e \rangle$	$\langle Z, \nabla Z \rangle$

Second-order:

Covariance matrix	$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$	$\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$
Hessian matrix	$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$	$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$
Gradient of expectation	$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e (\nabla r_e) \rangle$	$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$

$$p_e = \exp(\Phi_e \cdot \theta) \qquad \nabla p_e = p_e \Phi_e$$

- choose a semiring
- define a weight for each edge
- run inside algorithm

Applications of expectation semirings: a summary

First-order:

Quantity	Weight for edge e	Value at root
Expectation	$\langle p_e, p_e r_e \rangle$	$\langle Z, \overline{r} \rangle$
First-order gradient	$\langle p_e, \nabla p_e \rangle$	$\langle Z, \nabla Z \rangle$

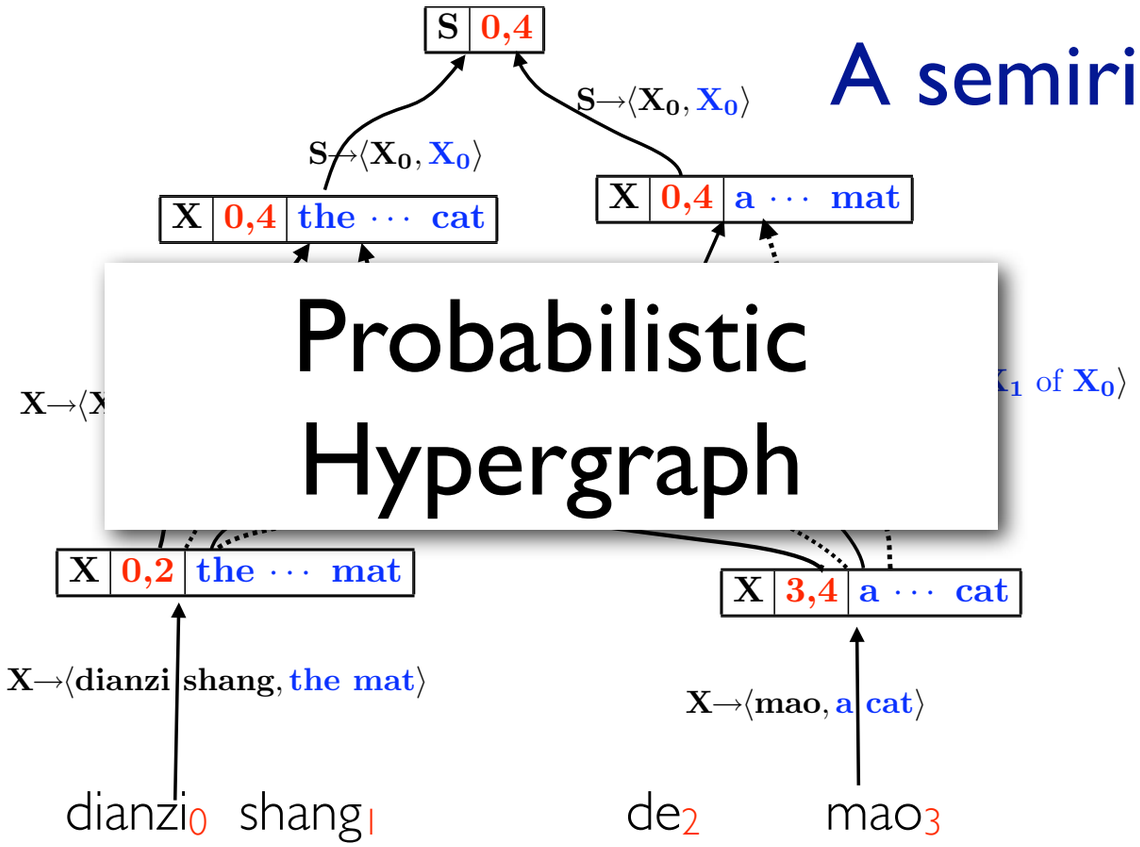
Second-order:

Covariance matrix	$\langle p_e, p_e r_e, p_e s_e, p_e r_e s_e \rangle$	$\langle Z, \overline{r}, \overline{s}, \overline{t} \rangle$
Hessian matrix	$\langle p_e, \nabla p_e, \nabla p_e, \nabla^2 p_e \rangle$	$\langle Z, \nabla Z, \nabla Z, \nabla^2 Z \rangle$
Gradient of expectation	$\langle p_e, p_e r_e, \nabla p_e, (\nabla p_e) r_e + p_e (\nabla r_e) \rangle$	$\langle Z, \overline{r}, \nabla Z, \nabla \overline{r} \rangle$

Entropy and Bayes risk are expectations.

- choose a semiring
- define a weight for each edge
- run inside algorithm

A semiring framework to compute all of these

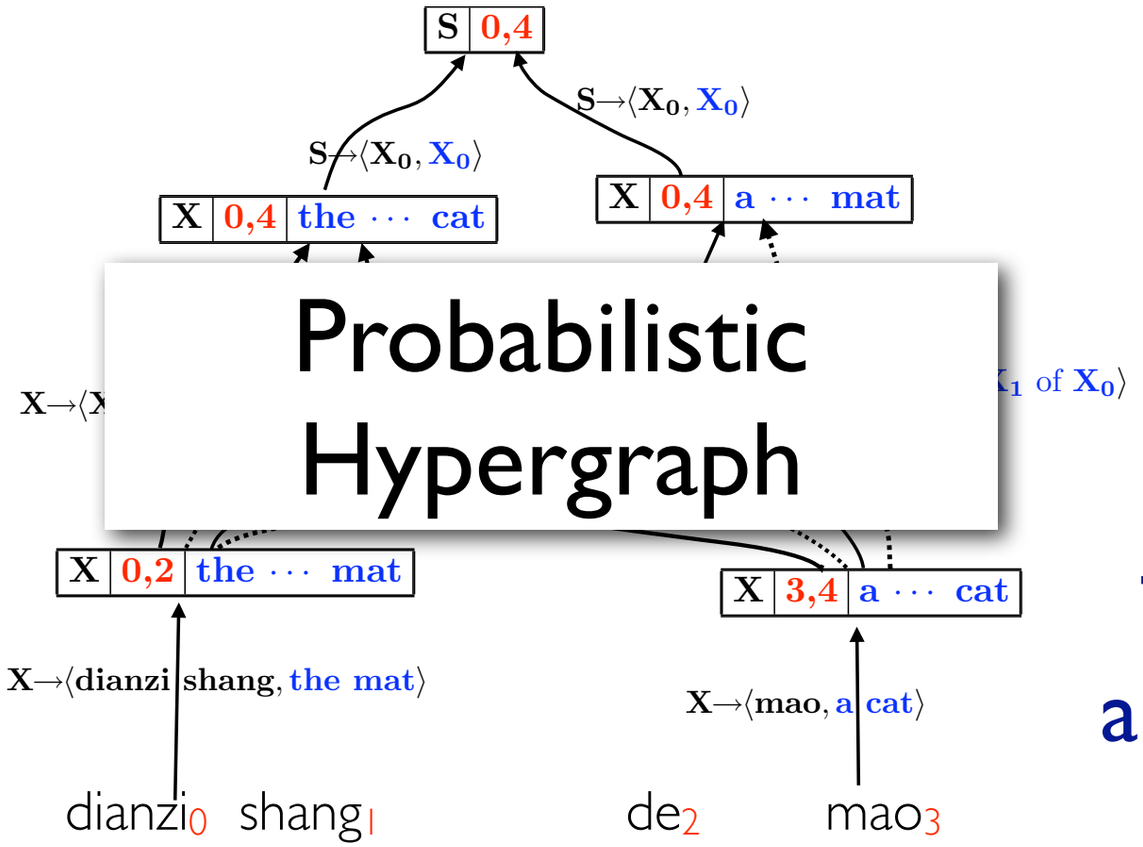


Probabilistic Hypergraph

- "decoding" quantities:
  - Viterbi
  - K-best
  - Counting
  - .....

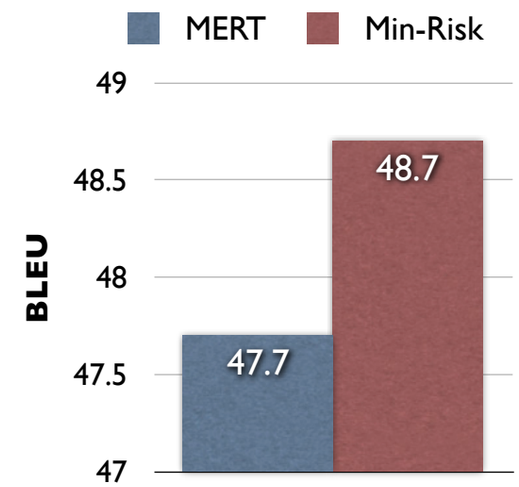
- First-order quantities:
  - expectation
    - entropy
    - Bayes risk
    - cross-entropy
    - KL divergence
    - feature expectations
  - first-order gradient of Z

- Second-order quantities:
  - Expectation over product
    - interaction between features
  - Hessian matrix of Z
    - second-order gradient descent
  - gradient of expectation
    - gradient of entropy or Bayes risk



Probabilistic Hypergraph

Improved BLEU score!



This work provides a unified, elegant, and efficient framework to compute all of these!

- First-order quantities:
  - expectation
    - entropy
    - Bayes risk
    - cross-entropy
    - KL divergence
    - feature expectations
  - first-order gradient of Z

- Second-order quantities:
  - Expectation over product
    - interaction between features
  - Hessian matrix of Z
    - second-order gradient descent
  - gradient of expectation
    - gradient of entropy or Bayes risk

Future: machine learning for MT

feature interaction

second-order gradient descent

.....

minimum risk

deterministic annealing

active learning

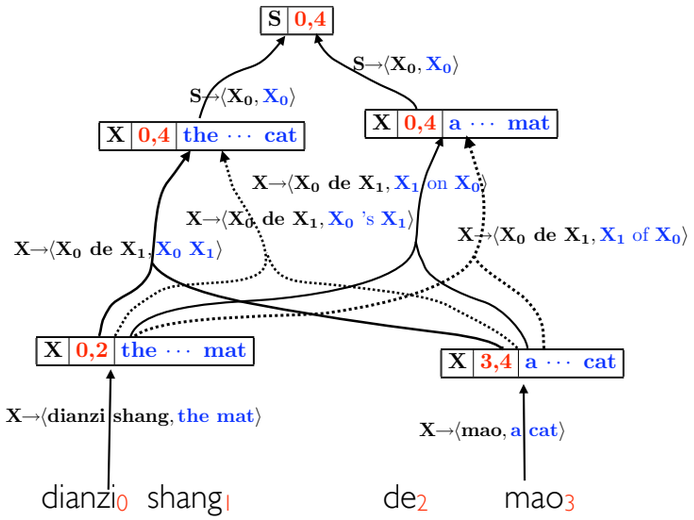
semi-supervised learning

semirings for parameter estimation

Outline

- Hypergraph as hypothesis space
- Training methods (Li and Eisner, 2009)
- Exact and approximate inference
  - exact inference
    - ★ semiring framework (Li and Eisner, 2009)
  - approximate inference
    - ★ "principled" ones
      - ✓ variational decoding (Li et. al, 2009)
      - ✓ sampling and message passing methods
    - ★ heuristic-based ones
      - ✓ cube-pruning (Chiang, 2007)
- Joshua project (Li et. al, 2009)

Decoding over a hypergraph



Given a hypergraph of possible translations
(generated for a given foreign sentence by already-trained model)

Pick a single translation to output
(why not just pick the tree with the highest weight?)

Spurious Ambiguity

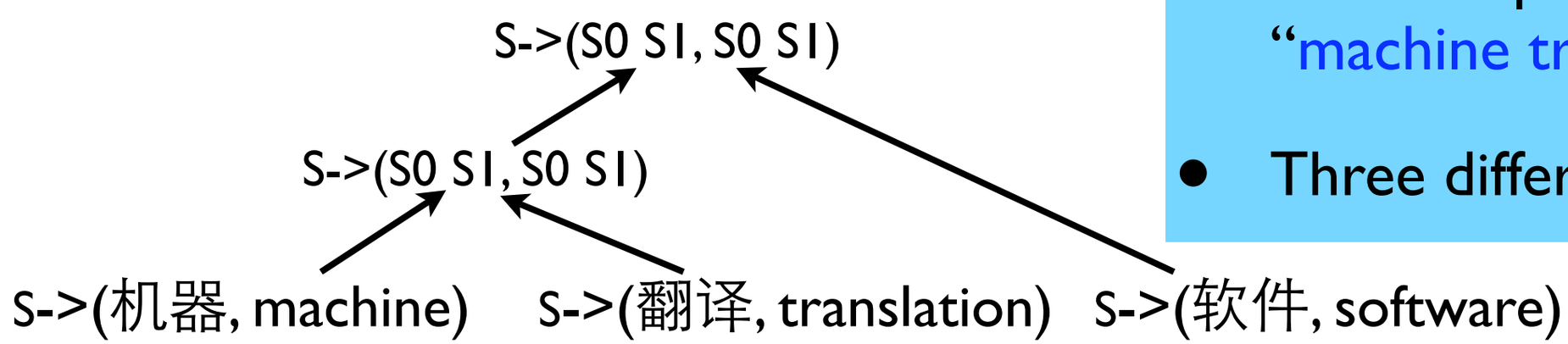
- Statistical models in MT exhibit **spurious ambiguity**
  - Many **different derivations** (e.g., trees or segmentations) generate the **same translation string**
- Tree-based MT systems
  - derivation tree ambiguity
- Regular phrase-based MT systems
  - phrase segmentation ambiguity

Spurious Ambiguity in Derivation Trees

机器 翻译 软件 machine translation software

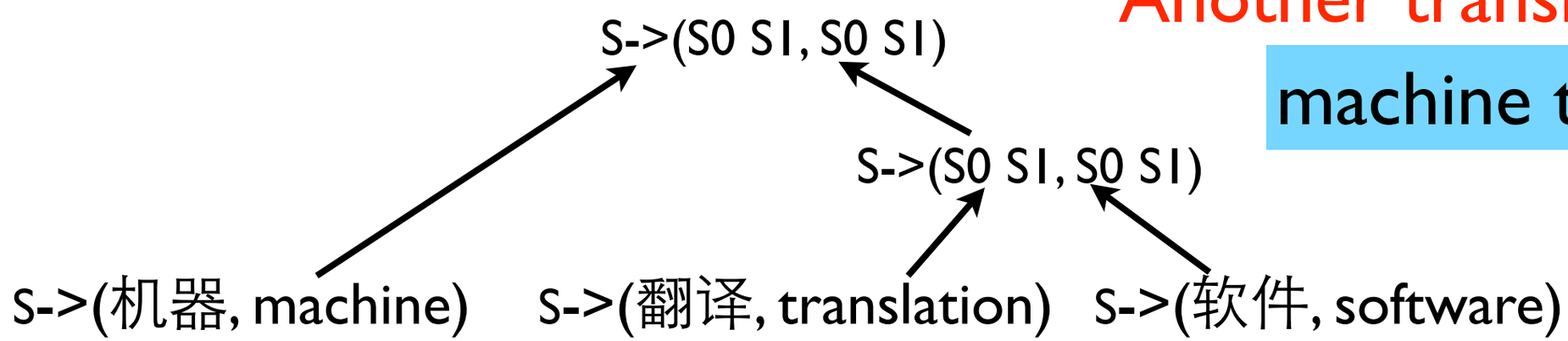
jiqi fanyi yuanjian

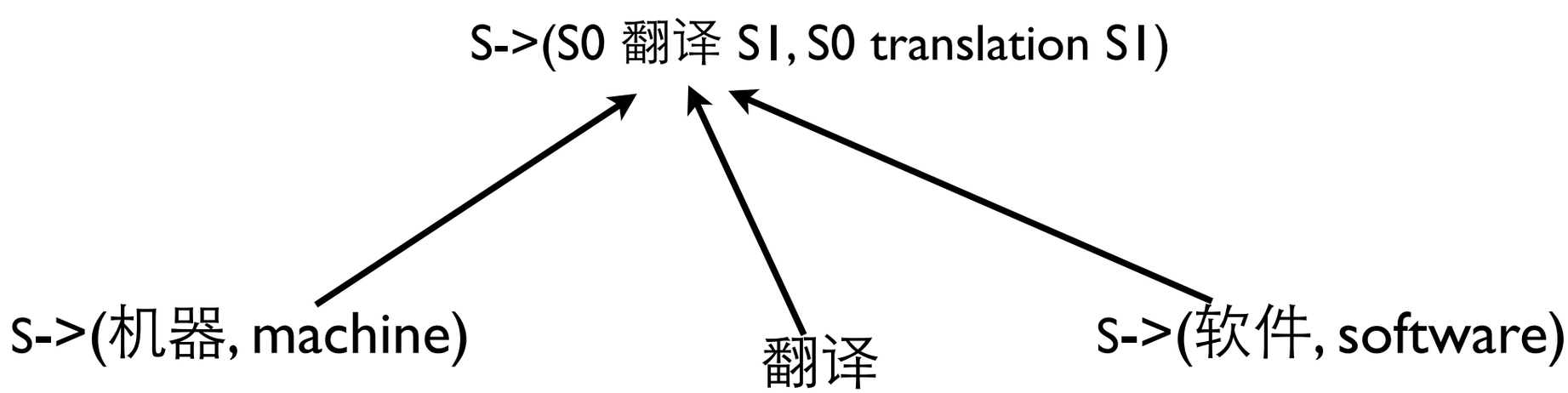
- Same output: "machine translation software"
- Three different derivation trees



Another translation:

machine transfer software





Maximum A Posterior (MAP) Decoding

translation string	MAP	derivation	probability
red translation			0.16
			0.14
blue translation			0.14
			0.13
green translation			0.12
			0.11
			0.10
			0.10

- Exact MAP decoding

$$y^* = \arg\max_{y \in \mathrm{Trans}(x)} p(y|x)$$

$$= \arg\max_{y \in \mathrm{Trans}(x)} \sum_{d \in \mathrm{D}(x,y)} p(y, d|x)$$

- x: Foreign sentence
- y: English sentence
- d: derivation

Maximum A Posterior (MAP) Decoding

translation string	MAP
red translation	0.28
blue translation	
green translation	

derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

- Exact MAP decoding

$$
\begin{aligned}
y^* &= \arg\max_{y \in \mathrm{Trans}(x)} p(y|x) \\
&= \arg\max_{y \in \mathrm{Trans}(x)} \sum_{d \in \mathrm{D}(x,y)} p(y, d|x)
\end{aligned}
$$

- x: Foreign sentence
- y: English sentence
- d: derivation

Maximum A Posterior (MAP) Decoding

translation string	MAP
red translation	0.28
blue translation	0.28
green translation	

derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

- Exact MAP decoding

$$
\begin{aligned}
y^* &= \arg\max_{y \in \mathrm{Trans}(x)} p(y|x) \\
&= \arg\max_{y \in \mathrm{Trans}(x)} \sum_{d \in \mathrm{D}(x,y)} p(y, d|x)
\end{aligned}
$$

- x: Foreign sentence
- y: English sentence
- d: derivation

Maximum A Posterior (MAP) Decoding

translation string	MAP
red translation	0.28
blue translation	0.28
green translation	0.44

derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

- Exact MAP decoding

$$
\begin{aligned}
y^* &= \arg\max_{y \in \mathrm{Trans}(x)} p(y|x) \\
&= \arg\max_{y \in \mathrm{Trans}(x)} \sum_{d \in \mathrm{D}(x,y)} p(y, d|x)
\end{aligned}
$$

- x: Foreign sentence
- y: English sentence
- d: derivation

Maximum A Posterior (MAP) Decoding

translation string	MAP
red translation	0.28
blue translation	0.28
green translation	0.44

derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

- Exact MAP decoding

$$
\begin{aligned}
y^* &= \arg\max_{y \in \mathrm{Trans}(x)} p(y|x) \\
&= \arg\max_{y \in \mathrm{Trans}(x)} \sum_{d \in \mathrm{D}(x,y)} p(y, d|x)
\end{aligned}
$$

exponential size

NP-hard (Sima'an 1996)

- x: Foreign sentence
- y: English sentence
- d: derivation

Viterbi Approximation

translation string	MAP	Viterbi	derivation	probability
red translation	0.28			0.16
				0.14
blue translation	0.28			0.14
green translation	0.44			0.13
				0.12
				0.11
				0.10
				0.10

- Viterbi approximation

$$
\begin{aligned}
y^* &= \arg \max_{y \in \mathrm{Trans}(x)} \boxed{\max_{d \in \mathrm{D}(x,y)}} p(y, d|x) \\
&= \mathrm{Y}(\arg \max_{d \in \mathrm{D}(x)} p(y, d|x))
\end{aligned}
$$

Viterbi Approximation

translation string	MAP	Viterbi
red translation	0.28	0.16
blue translation	0.28	0.14
green translation	0.44	0.13

derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

- Viterbi approximation

$$
\begin{aligned}
y^* &= \arg \max_{y \in \mathrm{Trans}(x)} \boxed{\max_{d \in \mathrm{D}(x,y)}} p(y, d|x) \\
&= \mathrm{Y}(\arg \max_{d \in \mathrm{D}(x)} p(y, d|x))
\end{aligned}
$$

N-best Approximation

translation string	MAP	Viterbi	4-best crunching	derivation	probability
red translation	0.28	0.16			0.16
blue translation	0.28	0.14			0.14
green translation	0.44	0.13			0.14
					0.13
					0.12
					0.11
					0.10
					0.10

- N-best approximation (crunching) (May and Knight, 2006)

$$y^* = \arg\max_{y \in \mathrm{Trans}(x)} \sum_{\boxed{d \in \mathrm{D}(x,y) \cap \mathrm{ND}(x)}} p(y, d|x)$$

N-best Approximation

translation string	MAP	Viterbi	4-best crunching	derivation	probability
red translation	0.28	0.16	0.16		0.16
blue translation	0.28	0.14	0.28		0.14
					0.14
green translation	0.44	0.13	0.13		0.13
					0.12
					0.11
					0.10
					0.10

- N-best approximation (crunching) (May and Knight, 2006)

$$y^* = \arg\max_{y \in \mathrm{Trans}(x)} \sum_{\boxed{d \in \mathrm{D}(x,y) \cap \mathrm{ND}(x)}} p(y, d|x)$$

MAP vs. Approximations

translation string	MAP	Viterbi	4-best crunching	derivation	probability
red translation	0.28	0.16	0.16		0.16
					0.14
blue translation	0.28	0.14	0.28		0.14
green translation	0.44	0.13	0.13		0.13
					0.12
					0.11
					0.10
					0.10

- Exact MAP decoding under spurious ambiguity is intractable on HG
- Viterbi and crunching are efficient, but ignore most derivations
- Our goal: develop an approximation that considers all the derivations but still allows tractable decoding

Variational Decoding

Decoding using **Variational** approximation

Decoding using a sentence-specific **approximate** distribution

Variational Decoding for MT: an Overview

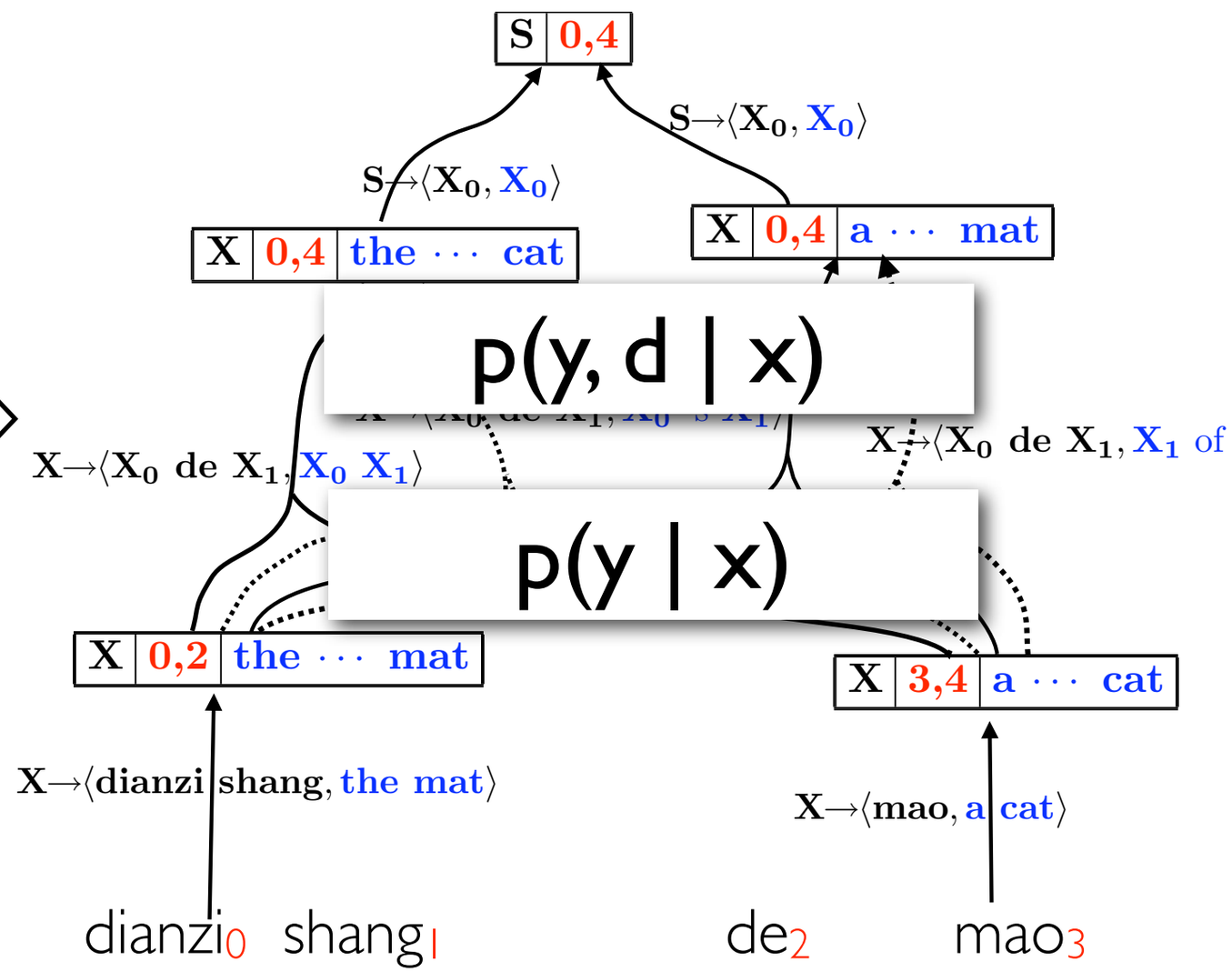
Sentence-specific decoding

MAP decoding under P is intractable

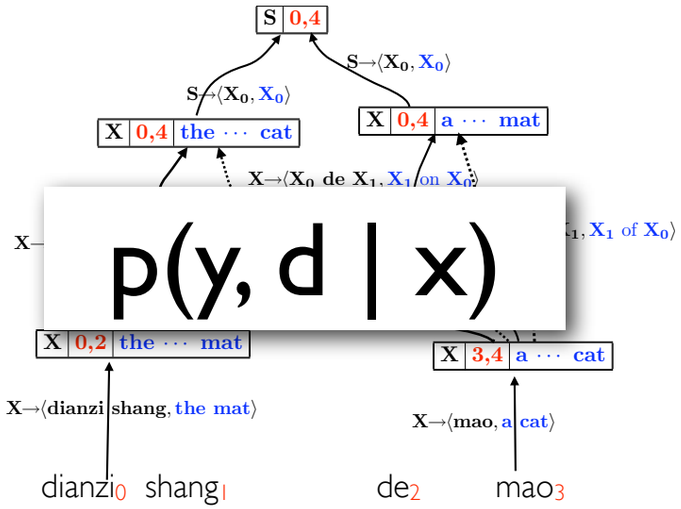
Three steps:

1. Generate a hypergraph for the foreign sentence

Foreign sentence x ⇒ SMT ⇒



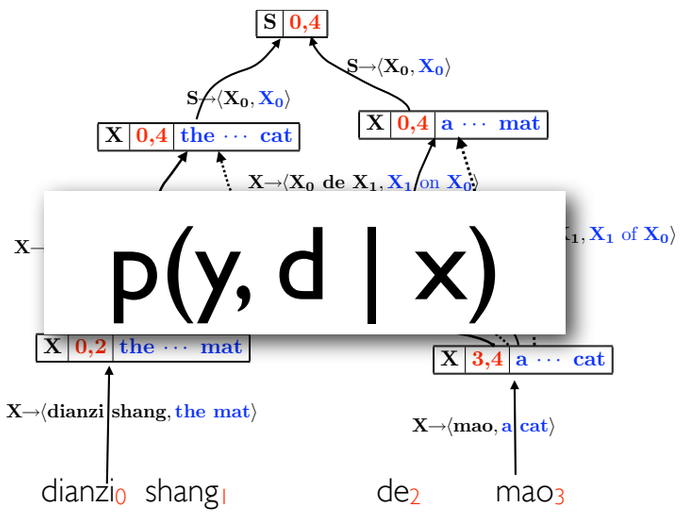
1



$p(y, d \mid x)$

Generate a hypergraph

2



$p(y, d \mid x)$

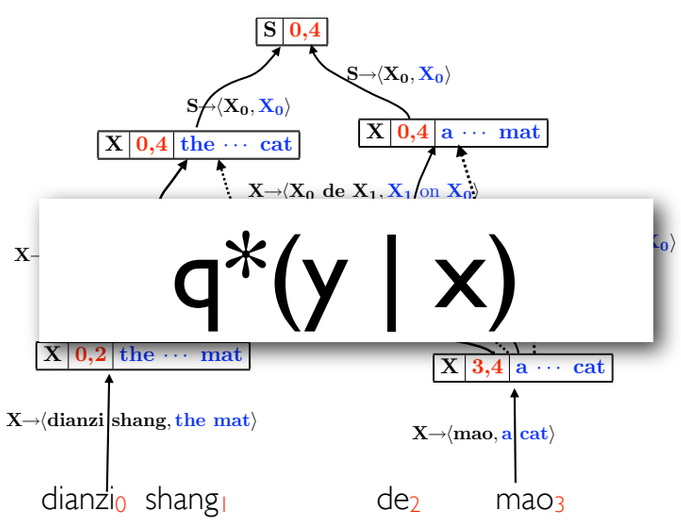
Estimate a model from the hypergraph

q* is an n-gram model over output strings.

$q^*(y \mid x)$

$\approx \sum_{d \in D(x,y)} p(y,d|x)$

3



$q^*(y \mid x)$

Decode using q* on the hypergraph

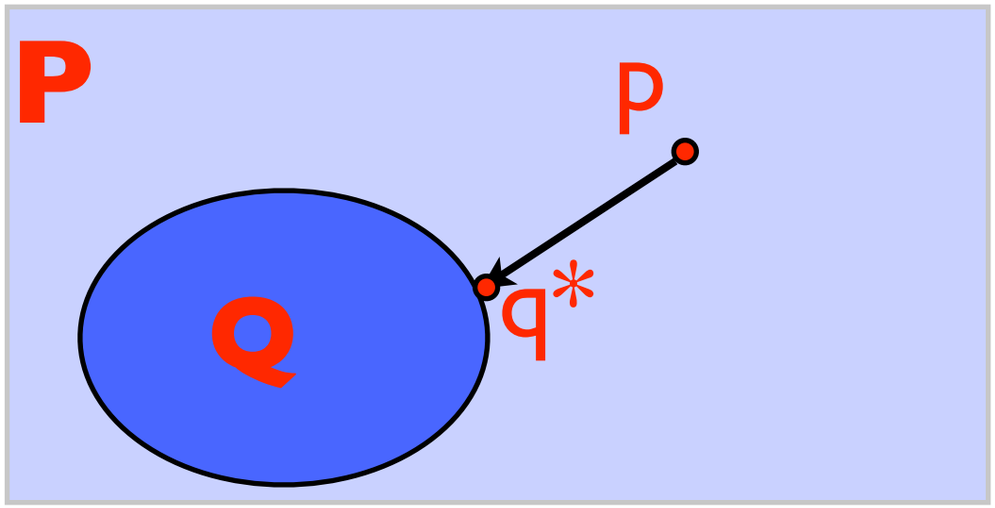
Variational Inference

- We want to do inference under p, but it is intractable

$$y^* = \arg\max_y p(y|x)$$

- Instead, we derive a simpler distribution q*

$$q^* = \arg\min_{q \in Q} \mathrm{KL}(p||q)$$



- Then, we will use q* as a surrogate for p in inference

$$y^* = \arg\max_y q^*(y \mid x)$$

Variational Approximation

- q^*: an approximation having minimum distance to p

$$
\begin{aligned}
q^* &= \arg\min_{q \in Q} \mathrm{KL}(p||q) \\
&= \arg\min_{q \in Q} \sum_{y \in \mathrm{Trans}(x)} p \log \frac{p}{q} \\
&= \arg\min_{q \in Q} \sum_{y \in \mathrm{Trans}(x)} (p \log p - p \log q) \\
&= \arg\max_{q \in Q} \sum_{y \in \mathrm{Trans}(x)} p \log q
\end{aligned}
$$

a family of distributions

constant

- Three questions
  - how to parameterize q?
  - how to estimate q^*?
  - how to use q^* for decoding?

Parameterization of $q \in Q$

- Naturally, we parameterize q as an n-gram model
  - The probability of a *string* is a product of the probabilities of those *n-grams* appearing in that string

3-gram model

y: a b c d e f

$$q(y) \quad = \quad q(a) \cdot q(b|a) \cdot q(c|ab) \cdot q(d|bc) \cdot q(e|cd) \cdot q(f|de)$$

Other ways of parameterizations are possible!

Parameterization of $q \in Q$

- Naturally, we parameterize q as an n-gram model
  - The probability of a *string* is a product of the probabilities of those *n-grams* appearing in that string

3-gram model

y: a b c d e f

$$q(y) \quad = \quad q(a) \cdot q(b|a) \cdot q(c|ab) \cdot q(d|bc) \cdot q(e|cd) \cdot q(f|de)$$

how to estimate these n-gram probabilities?

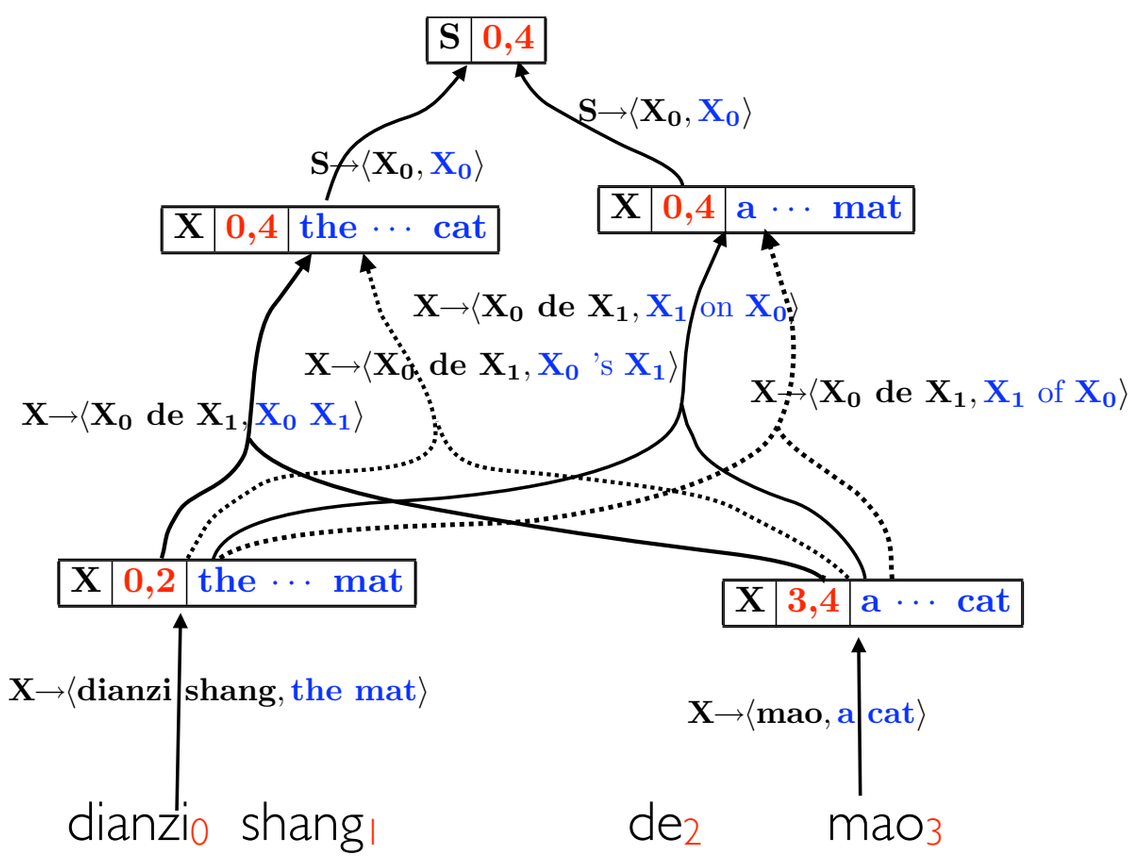
Estimation of $q^* \in Q$

- Variational approximation

$$q^* = \arg\max_{q \in Q} \sum_{y \in \mathrm{Trans}(x)} p \log q$$

- q^* is a maximum likelihood estimate (MLE) where p is the empirical distribution

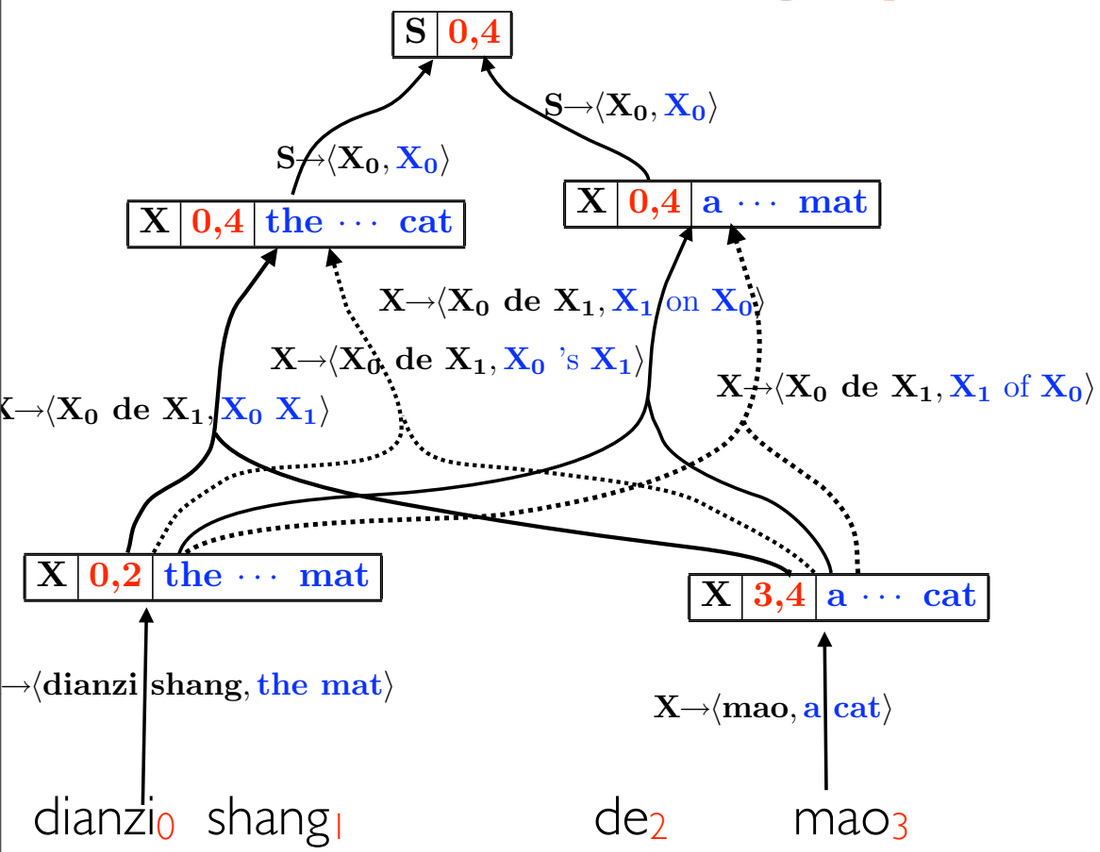
But in our case, p is defined **not** by a **corpus**, but by a **hypergraph** for a given test sentence!



estimate ⟹ bi-gram model

- brute force
- dynamic programming

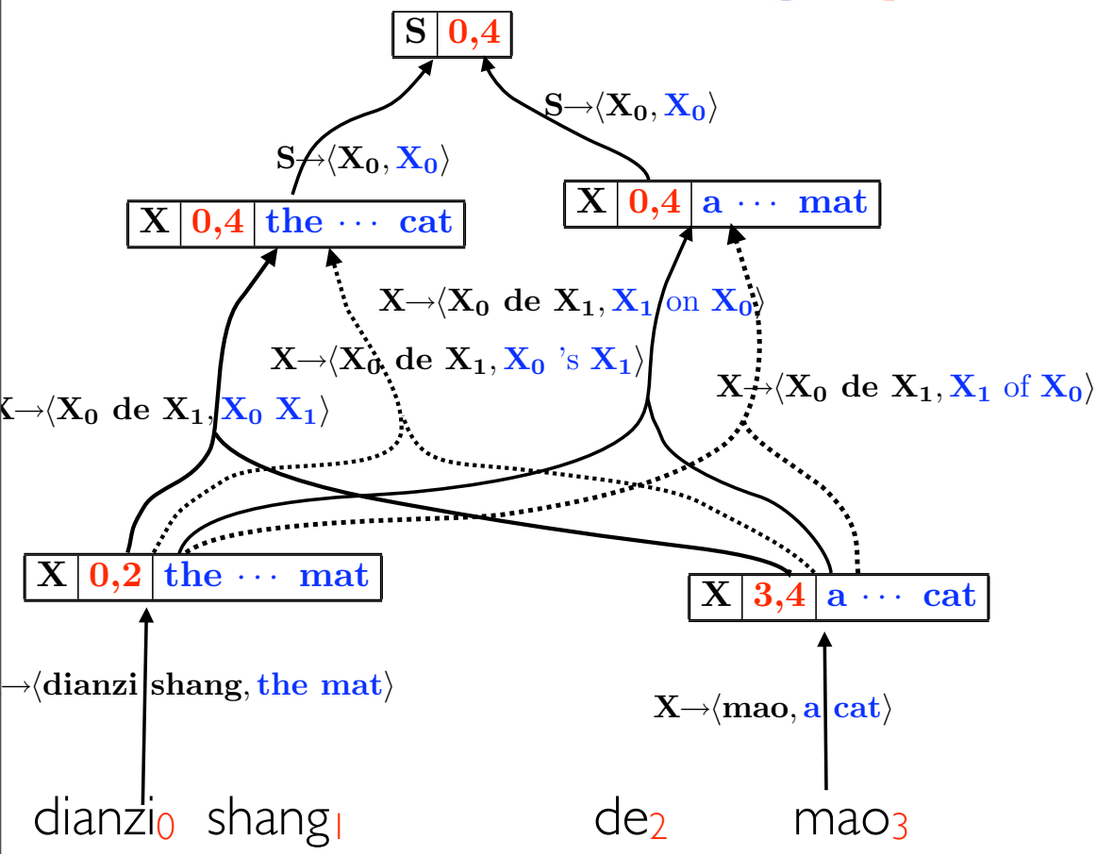
Estimating q* from a hypergraph: brute force



Bi-gram estimation:

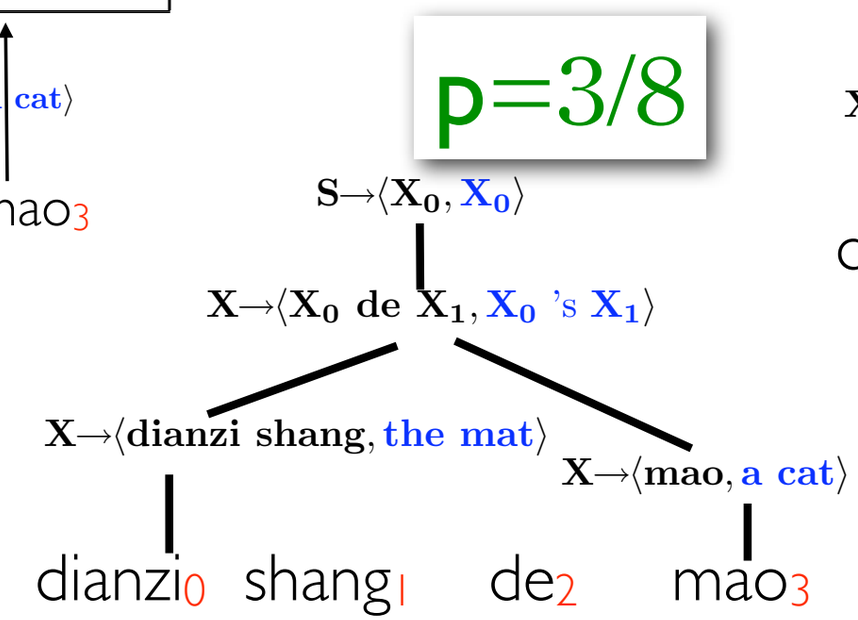
- unpack the hypergraph

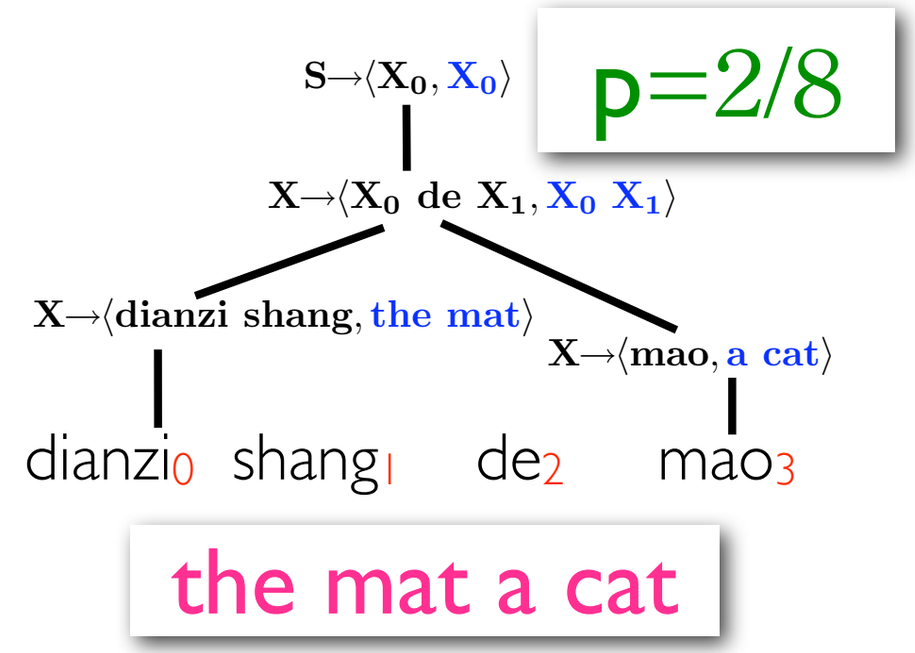
Estimating q* from a hypergraph: brute force

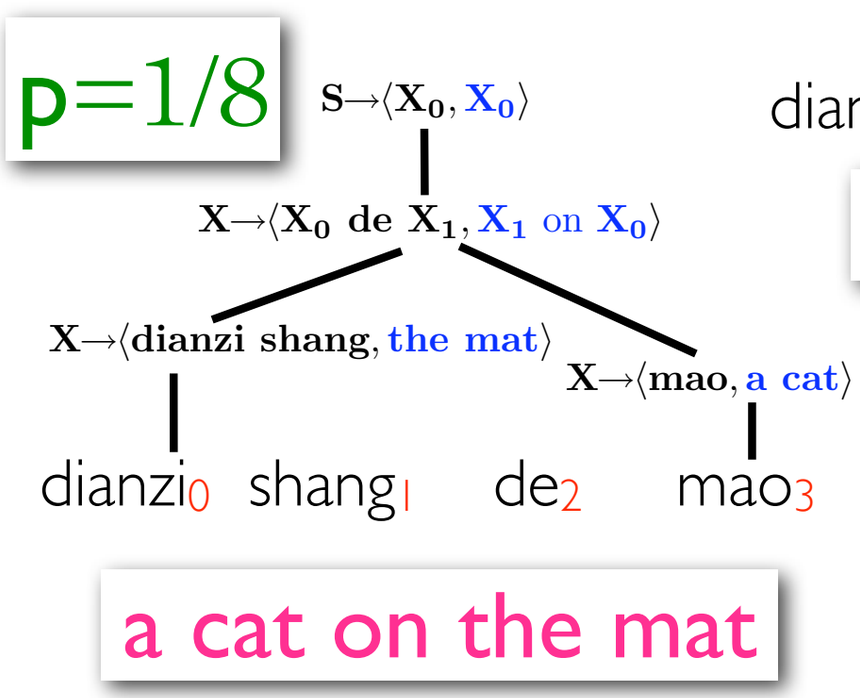


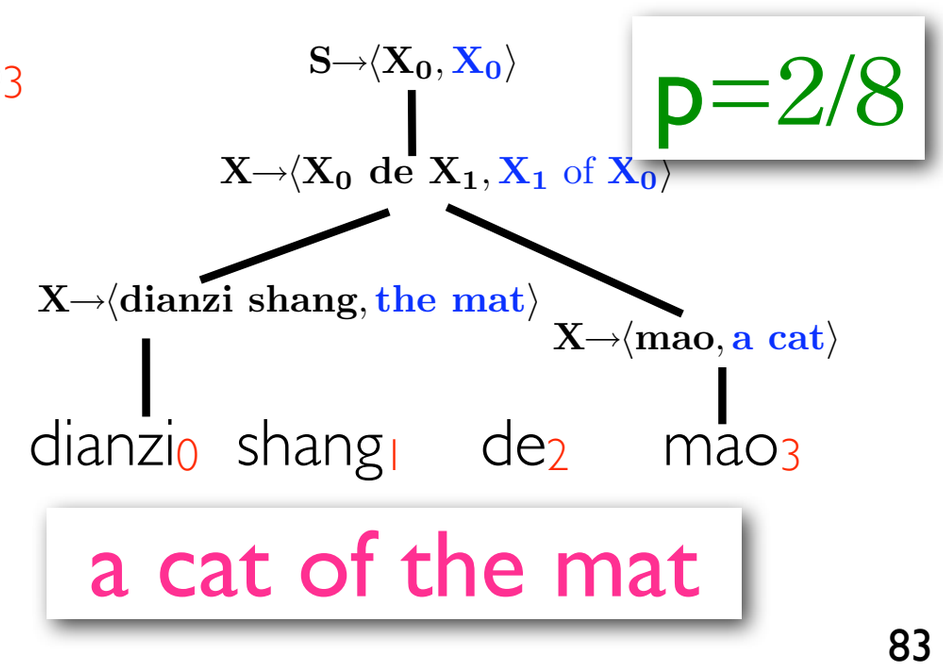
Bi-gram estimation:

- unpack the hypergraph









Estimating q* from a hypergraph: brute force

a cat on the mat — p=1/8

the mat 's a cat — p=3/8

the mat a cat — p=2/8

a cat of the mat — p=2/8

Bi-gram estimation:

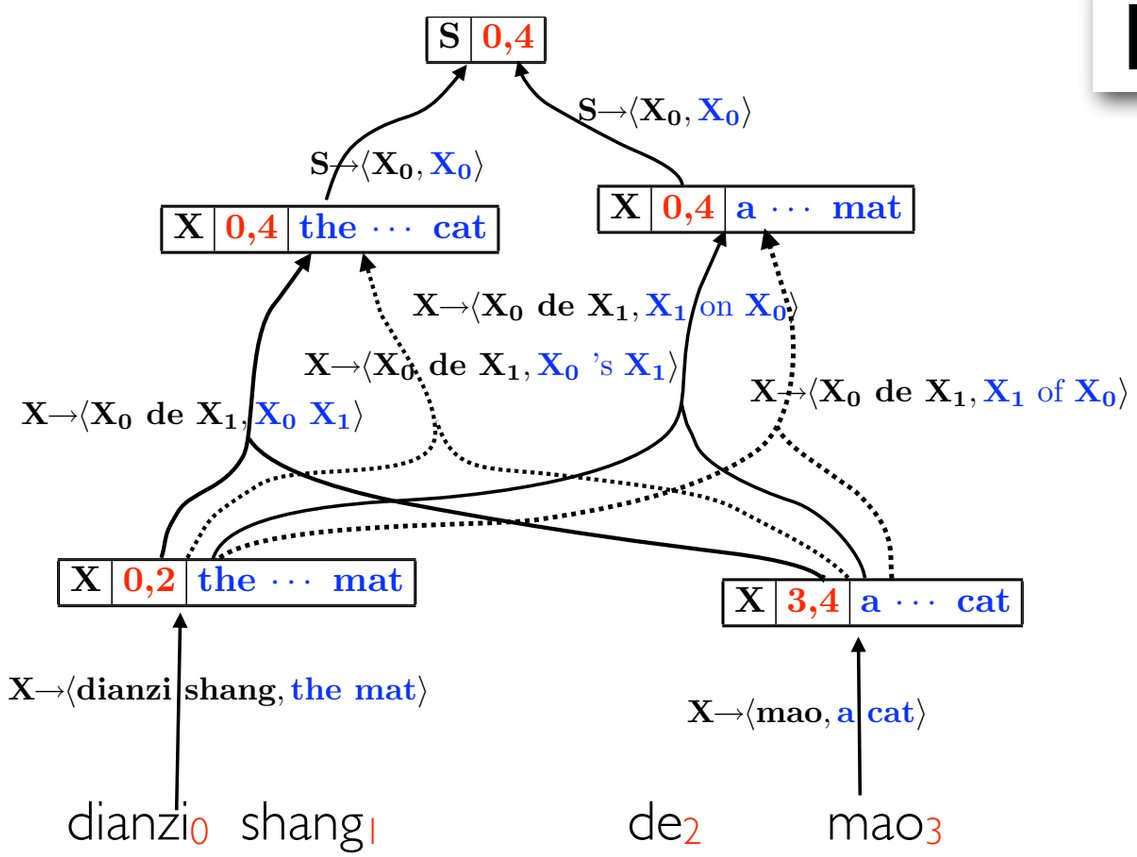
- unpack the hypergraph
- accumulate the **soft-count** of each bigram
- normalize the counts

q*(on | cat)=1/8

q*(</s> | cat)=5/8

q*(of | cat)=2/8

Estimating q* from a hypergraph: dynamic programming



Bi-gram estimation:

- run inside-outside on the hypergraph
  expectation semirings!
- accumulate the **soft-count** of each bigram at each hyperedge
- normalize the counts

Decoding using $q^* \in Q$

- Rescore the hypergraph HG(x)

$$y^* = \arg \max_{\boxed{y \in \mathrm{HG}(x)}} q^*(y|x)$$

q^* is an n-gram model.

- have efficient dynamic programming algorithms
  - score the hypergraph using an n-gram model

KL divergences under different variational models

$$q^* \quad = \quad \arg\min_{q \in Q} \mathrm{KL}(p||q) = \mathrm{H}(p,q) - \mathrm{H}(p)$$

Measure bits/word	$\overline{\mathrm{H}}(p)$	$\overline{\mathrm{KL}}(p\|\|\cdot)$			
		q_1^*	q_2^*	q_3^*	q_4^*
MT'04	1.36	0.97	0.32	0.21	0.17
MT'05	1.37	0.94	0.32	0.21	0.17

- Larger n ==> better approximation q_n ==> smaller KL divergence from p
- The reduction of KL divergence happens mostly when switching from unigram to bigram

KL divergences under different variational models

$$q^* \quad = \quad \arg\min_{q \in Q} \mathrm{KL}(p||q) = \mathrm{H}(p,q) - \mathrm{H}(p)$$

Measure bits/word	$\overline{\mathrm{H}}(p)$	$\overline{\mathrm{KL}}(p\|\|\cdot)$			
		q_1^*	q_2^*	q_3^*	q_4^*
MT'04	1.36	0.97	0.32	0.21	0.17
MT'05	1.37	0.94	0.32	0.21	0.17

How to compute KL on a **hypergraph**?

use expectation semiring, again ☺

BLEU scores when using a single variational n-gram model

Decoding scheme	MT'04	MT'05
Viterbi	35.4	32.6
1gram	25.9	24.5
2gram	**36.1**	**33.4**
3gram	36.0	33.1
4gram	35.8	32.9

- unigram performs very badly
- bigram achieves best BLEU scores ???

modeling error in p

BLEU cares about both low- and high-order n-gram matches

- Interpolate variational n-gram approximations for different n

$$y^* = \arg \max_{y \in \mathrm{HG}(x)} \sum_n \theta_n \cdot \log q_n^*(y \mid x)$$

- Viterbi is yet another approximation of p, so throw it in too

$$y^* = \arg \max_{y \in \mathrm{HG}(x)} \left(\sum_n \theta_n \cdot \log q_n^*(y \mid x) + \theta_v \cdot \log p_{\mathrm{Viterbi}}(y \mid x) \right)$$

Minimum Bayes Risk (MBR) decoding?

(Kumar and Byrne, 2004)

(Tromble et al. 2008)

(Denero et al. 2009)

(Li et al. 2009)

Minimum Risk Decoding

- Maximum A Posterior (MAP) decoding
  - find the most **probable** translation string

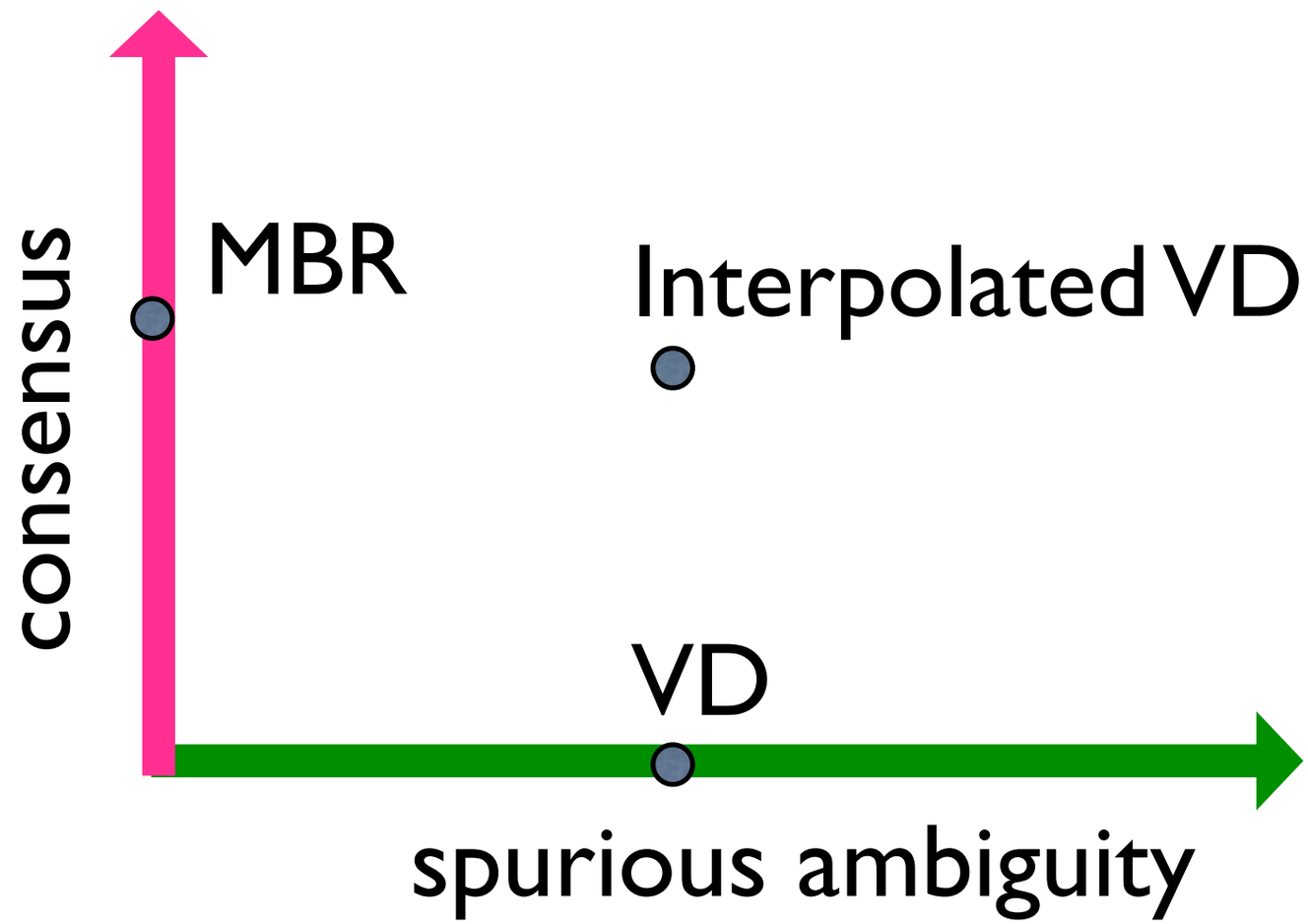
$$y^* = \arg\max_{y \in \mathrm{HG}(x)} p(y|x)$$

- Minimum risk decoding
  - find the **consensus** translation string

$$y^* = \arg\min_{y \in \mathrm{HG}(x)} \mathrm{Risk}(y)$$

risk = expected loss = expected error rate

Variational Decoding(VD) vs. MBR (Tromble et al. 2008)



Both BLEU metric and our variational distributions happen to use n-gram dependencies.

- Variational decoding with interpolation

$$y^* = \arg\max_{y \in \mathrm{HG}(x)} \sum_n \theta_n \cdot \log q_n^*(y \mid x)$$

decision rule

$$q_n(y \mid x) = \prod_{w \in W_n} q(r(w) \mid h(w), x)^{c_w(y)}$$

n-gram model

$$q(r(w) \mid h(w), x) = \frac{\sum_{y'} c_w(y')p(y' \mid x)}{\sum_{y'} c_{h(w)}(y')p(y' \mid x)}$$

n-gram probability

- Minimum risk decoding (Tromble et al. 2008)

$$y^* = \arg\max_{y \in \mathrm{HG}(x)} \sum_n \theta_n \cdot g_n(y \mid x)$$

decision rule

$$g_n(y \mid x) = \sum_{w \in W_n} g(w \mid x) c_w(y)$$

n-gram model

$$g(w \mid x) = \sum_{y'} \delta_w(y') p(y' \mid x)$$

n-gram probability

- Variational decoding with interpolation

$$y^* = \arg\max_{y \in \mathrm{HG}(x)} \sum_n \theta_n \cdot \log q_n^*(y \mid x)$$

$$q_n(y \mid x) = \prod_{w \in W_n} q(r(w) \mid h(w), x)^{c_w(y)}$$

$$q(r(w) \mid h(w), x) = \frac{\sum_{y'} c_w(y')p(y' \mid x)}{\sum_{y'} c_{h(w)}(y')p(y' \mid x)}$$

- Minimum risk decoding (Tromble et al. 2008)

$$y^* = \arg\max_{y \in \mathrm{HG}(x)} \sum_n \theta_n \cdot g_n(y \mid x)$$

$$\boxed{g_n(y \mid x) = \sum_{w \in W_n} g(w \mid x) c_w(y)} \longrightarrow \text{non-probabilistic}$$

$$\boxed{g(w \mid x) = \sum_{y'} \delta_w(y') p(y' \mid x)} \longrightarrow \text{very expensive to compute}$$

BLEU Results on Chinese-English NIST MT 2004 Tasks

	Decoding scheme	BLEU
	Viterbi	35.4
(Kumar and Byrne, 2004)	MBR (K=1000)	35.8
(May and Knight, 2006)	Crunching (N=10000)	35.7
	Crunching+MBR (N=10000)	35.8
New!	Variational (1to4gram+wp+vt)	**36.6**

- variational decoding improves over Viterbi, MBR, and crunching

Variational Inference

- We want to do inference under p, but it is intractable

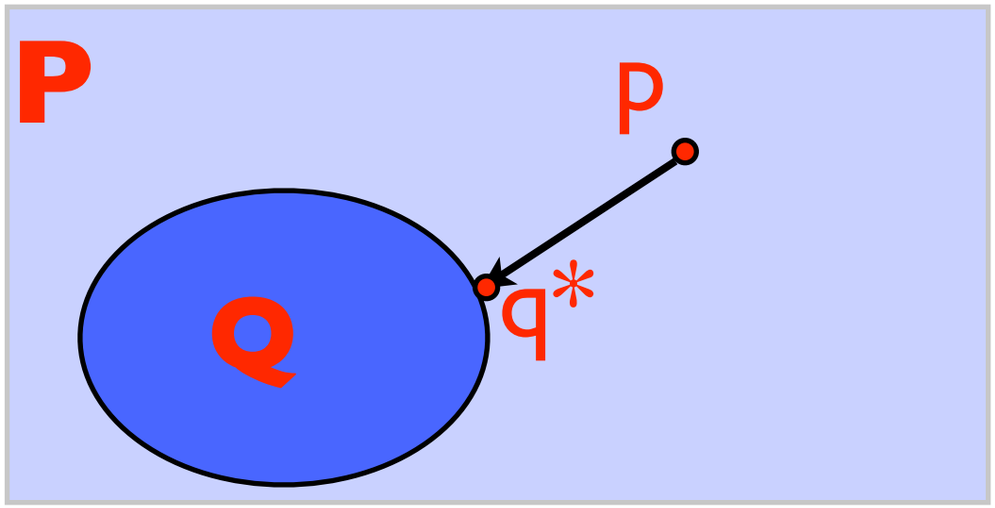
intractable

$$y^* = \arg\max_y p(y|x)$$

- Instead, we derive a simpler distribution q*

tractable

$$q^* = \arg\min_{q \in Q} \mathrm{KL}(p||q)$$



- Then, we will use q* as a surrogate for p in inference

tractable

$$y^* = \arg\max_y q^*(y \mid x)$$

Summary of Variational Decoding

- Exact MAP decoding with spurious ambiguity is intractable
- Viterbi or N-best approximations are efficient, but ignore most derivations
- We developed a variational approximation, which considers all derivations but still allows tractable decoding
- Our variational decoding improves a state of the art baseline

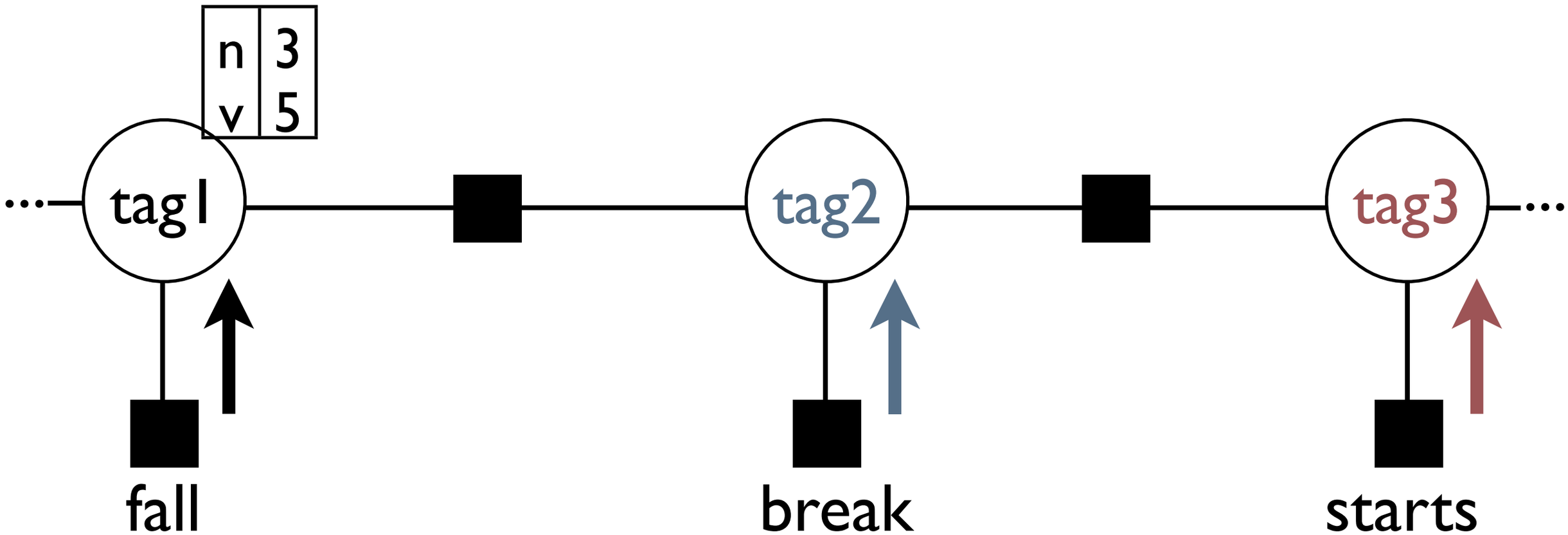
Future directions

- Decoding with spurious ambiguity is a common problem in many other NLP applications
  - Models with latent variables
  - Data oriented parsing (DOP)
  - Hidden Markov Models (HMM)
  - .....

Outline

- Hypergraph as hypothesis space
- Training methods (Li and Eisner, 2009)
- Exact and approximate inference
  - exact inference
    - ★ semiring framework (Li and Eisner, 2009)
  - approximate inference
    - ★ "principled" ones
      - ✓ variational decoding (Li et. al, 2009)
      - ✓ sampling and message passing methods
    - ★ heuristic-based ones
      - ✓ cube-pruning (Chiang, 2007)
- Joshua project (Li et. al, 2009)

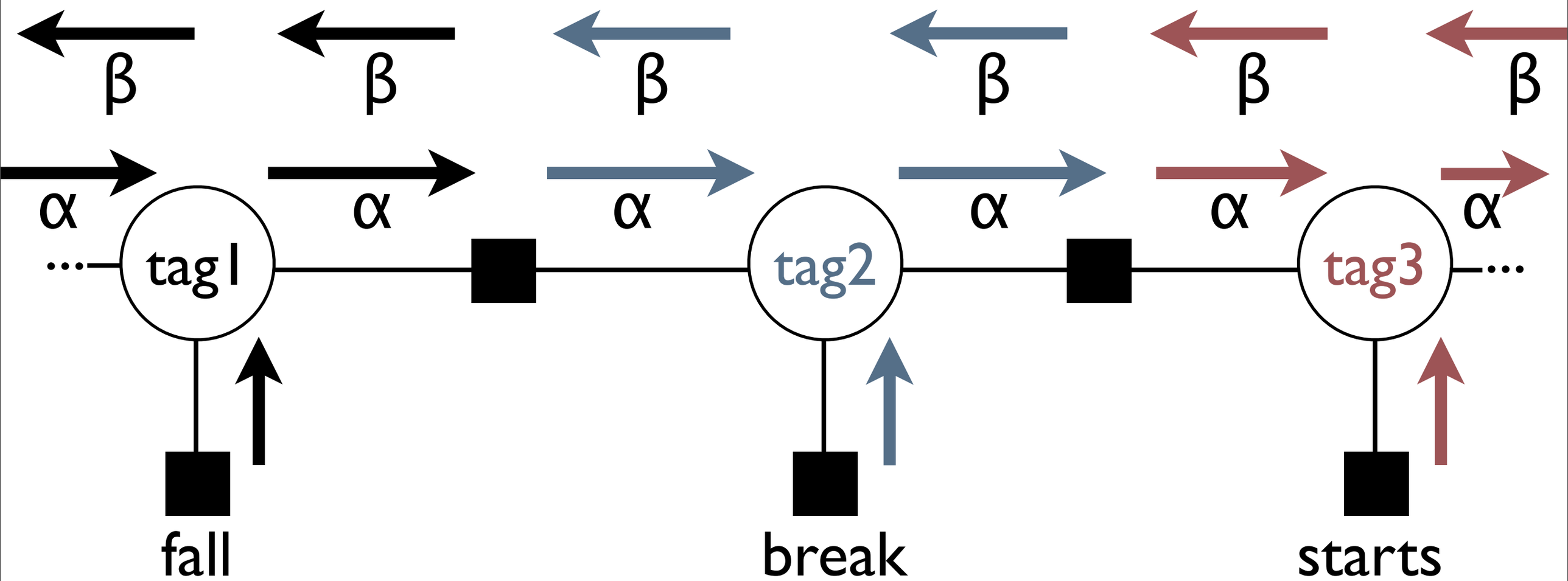
Graphical Model for Part of Speech Tagging



Variable: pos tags as values

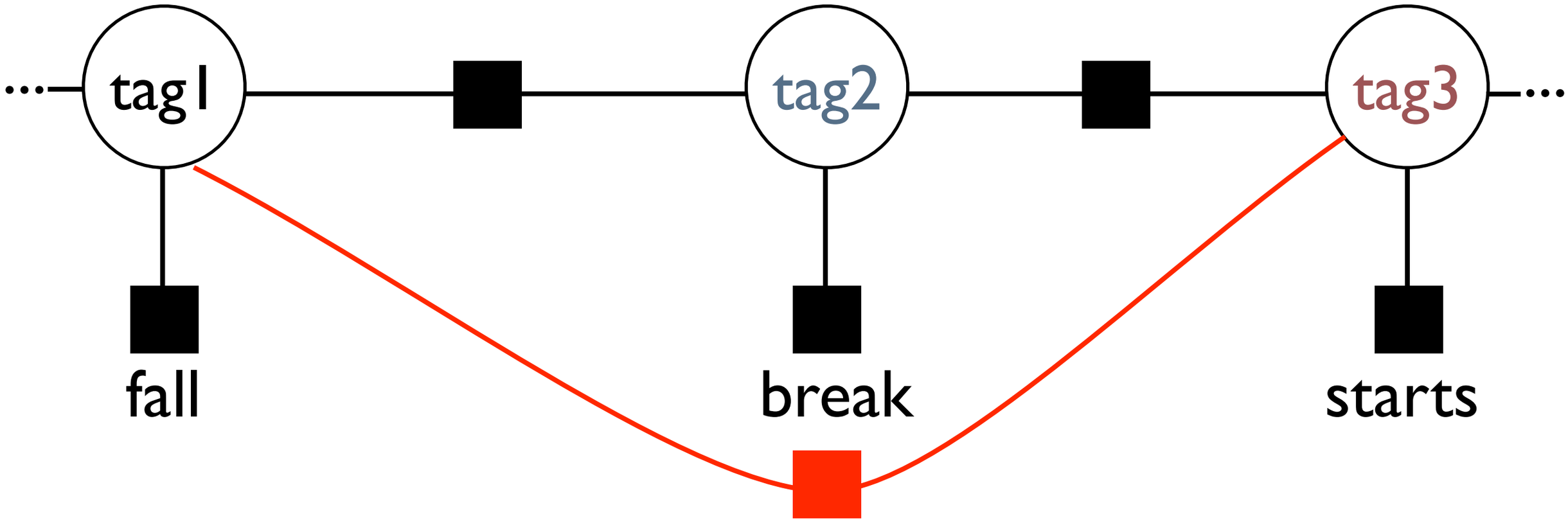
Factor: encode transition or observation probabilities

Belief Propagation



- α sends evidence from left part of graph
- β sends evidence from right part of graph

Non-local dependency



Loopy BP!

Variable: pos tags as values

Factor: encode transition or observation probabilities

Belief Propagation for NLP

- Dependency parsing with non-local information (Smith and Eisner, 2008)
- Word alignment (Cromieres and Kurohashi, 2009)
- String to string transducers (Dreyer and Eisner, 2009)

Sampling

- General idea
  - Generate a randomized n-best list
  - Compute posterior or max over the randomized n-best list
- Sampling used for MT
  - Gibbs sampler for Bayesian inference (Blunsom et. al, 2009)
  - Gibbs sampling for MAP decoding with spurious ambiguity (Arun et. al, 2009)
  - Gibbs sampling for word alignment (Denero et. al, 2008)

Heuristic-based Approximate Inference

- Beam pruning
- Cube pruning (Chiang, 2007)
- Cube growing (Huang and Chiang, 2007)
- Forest re-ranking (Huang, 2008)
- Oracle extraction (Li and Khudanpur, 2009)

Outline

- Hypergraph as hypothesis space
- Training methods (Li and Eisner, 2009)
- Exact and approximate inference
  - exact inference
    - ★ semiring framework (Li and Eisner, 2009)
  - approximate inference
    - ★ "principled" ones
      - ✓ variational decoding (Li et. al, 2009)
      - ✓ sampling and message passing methods
    - ★ heuristic-based ones
      - ✓ cube-pruning (Chiang, 2007)
- Joshua project (Li et. al, 2009)

Conclusions

- A hypergraph is a general way to represent a hypothesis space
- Minimum risk is a promising training objective
- Semiring framework is a general way to do exact inference
- Approximate inference has great applications (e.g., variational decoding) in MT

Future: machine learning for MT

Current paradigm: Approximate model + Exact inference

- rely on a specific grammars (e.g., FST, ITG, Hiero, or GHKM)
- everything is **local**, e.g., we can't even enforce *translation consistence* (similar words should get similar translations)
- rely on dynamic programs for exact inference

New paradigm: "Exact" model + Approximate inference

- use a flexible search space, each grammar is treated as a feature function
- include non-local dependency
- use principled approximate inference techniques, e.g., variational inference, Gibbs sampling, or belief propagation
- approximate inference calls dynamic programs for sub-steps

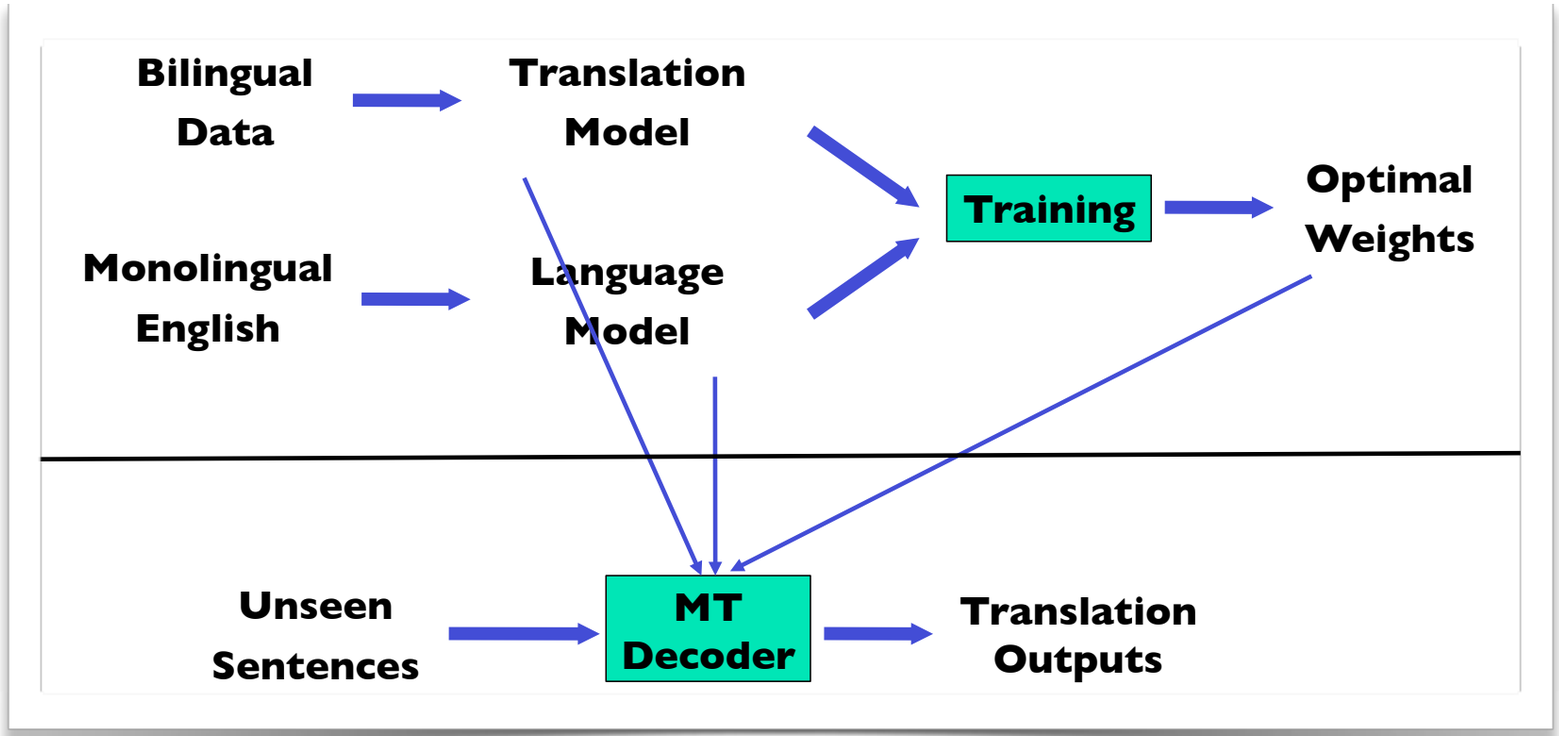
Joshua project

Motivation

- Towards a general purpose **tree**-based MT toolkit
  - string to string with latent tree structures (Chiang, 2005)
  - tree to string (Quirk et. al, 2006); (Liu et. al, 2006)
  - string to tree (Galley et. al, 2006)
  - tree to tree (Eisner, 2003)
- Successor to Moses

Joshua project

- An open-source parsing-based MT toolkit (Li et al. 2009)
  - support Hiero (Chiang, 2007) and SAMT (Venugopal et al., 2007)
- Team members
  - **Zhifei Li**, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Wren Thornton, Jonathan Weese, Juri Ganitkevitch, Lane Schwartz, and Omar Zaidan



Only rely on Giza++ and SRI LM!

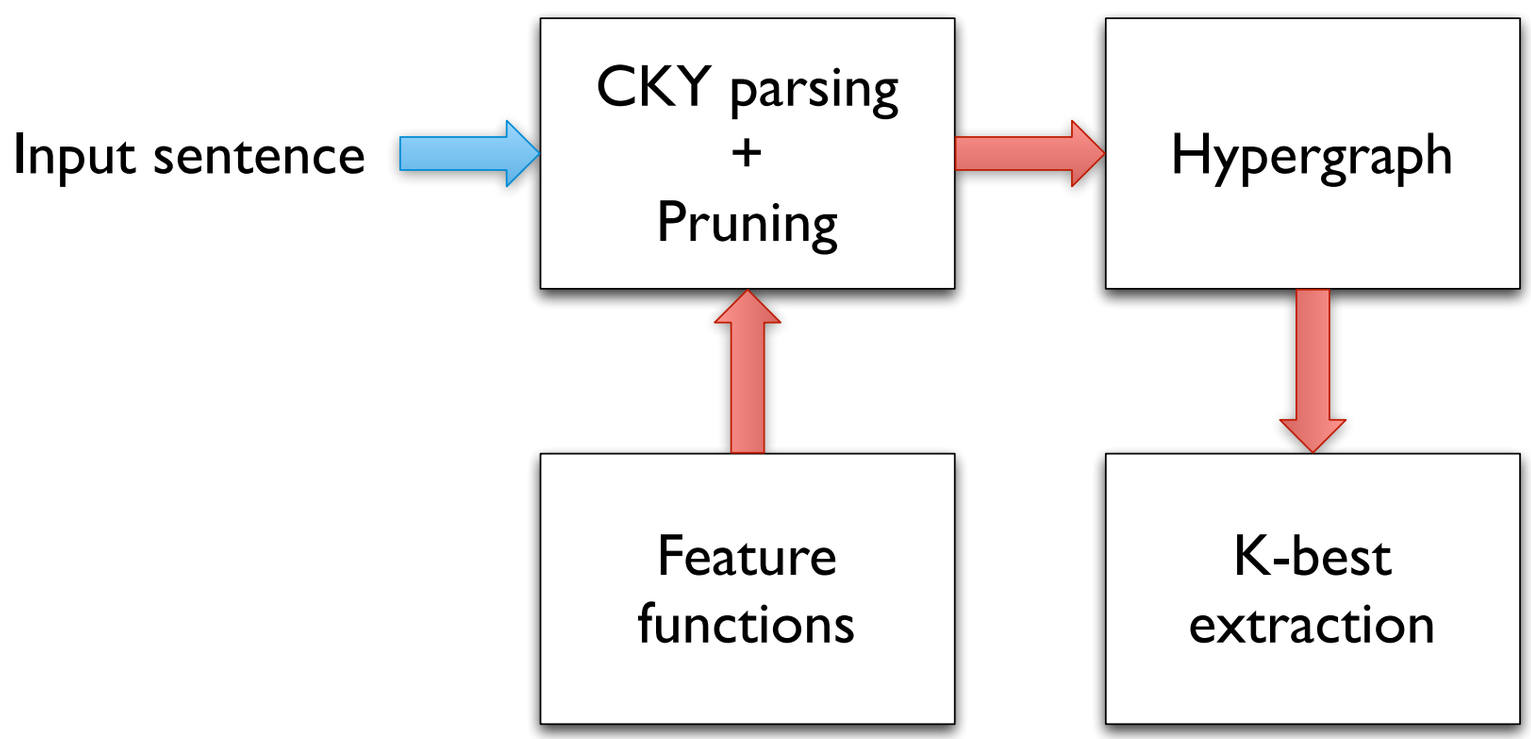
All the methods presented have been implemented in Joshua!

Highlights

- Everything is written in Java
  - easy to run under different platforms including Windows
- Easy to extend
  - organize the code into packages
  - define interfaces across packages
- Scalable
  - decode a sentence with one second

Chart-Parsing Decoder

- Input sentences are parsed using the CKY algorithm
- Feature functions are evaluated during parsing
- High-cost constituents are pruned during parsing
- Parsing results are stored in a hypergraph
- K-best extraction is performed on the hypergraph



Suffix-Array Grammar Extraction (Lopez, 2007)

- Grammars are extracted using suffix arrays
- Grammar rules can be extracted per sentence while decoding
- Allows for very large parallel training corpora
  - eliminates traditional requirement to explicitly extract, sort, and calculate probabilities for all possible rules.
  - only extracts rules actually needed to translate the sentence

Other functions

- Minimum error rate training (MERT)
- Parallel decoding
- Distributed language models
- Variational decoding
- Semiring parsing
- Minimum risk training
- Tree and hypergraph visualization

Quick start

- Check out the software
  - svn co https://joshua.svn.sf.net/svnroot/joshua/trunk joshua
- Prepare monolingual and bilingual training data
- Train a language model using the SRI LM toolkit
- Train a translation model
  - sub-sample bilingual data (optional)
  - create word alignments using GIZA++
  - run suffix-array grammar extraction
- Perform minimum error rate training
- Decode test sets

Thank you!

XieXie!

谢谢!