Variational Decoding for Statistical Machine Translation

Zhifei Li, Jason Eisner, and Sanjeev Khudanpur

Center for Language and Speech Processing

Computer Science Department

Johns Hopkins University

Spurious Ambiguity

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

Spurious Ambiguity in Phrase Segmentations

机器 翻译 软件

machine translation software

machine translation software

机器 翻译 软件

machine translation software

- Same output: "machine translation software"
- Three different phrase segmentations

机器 翻译 软件

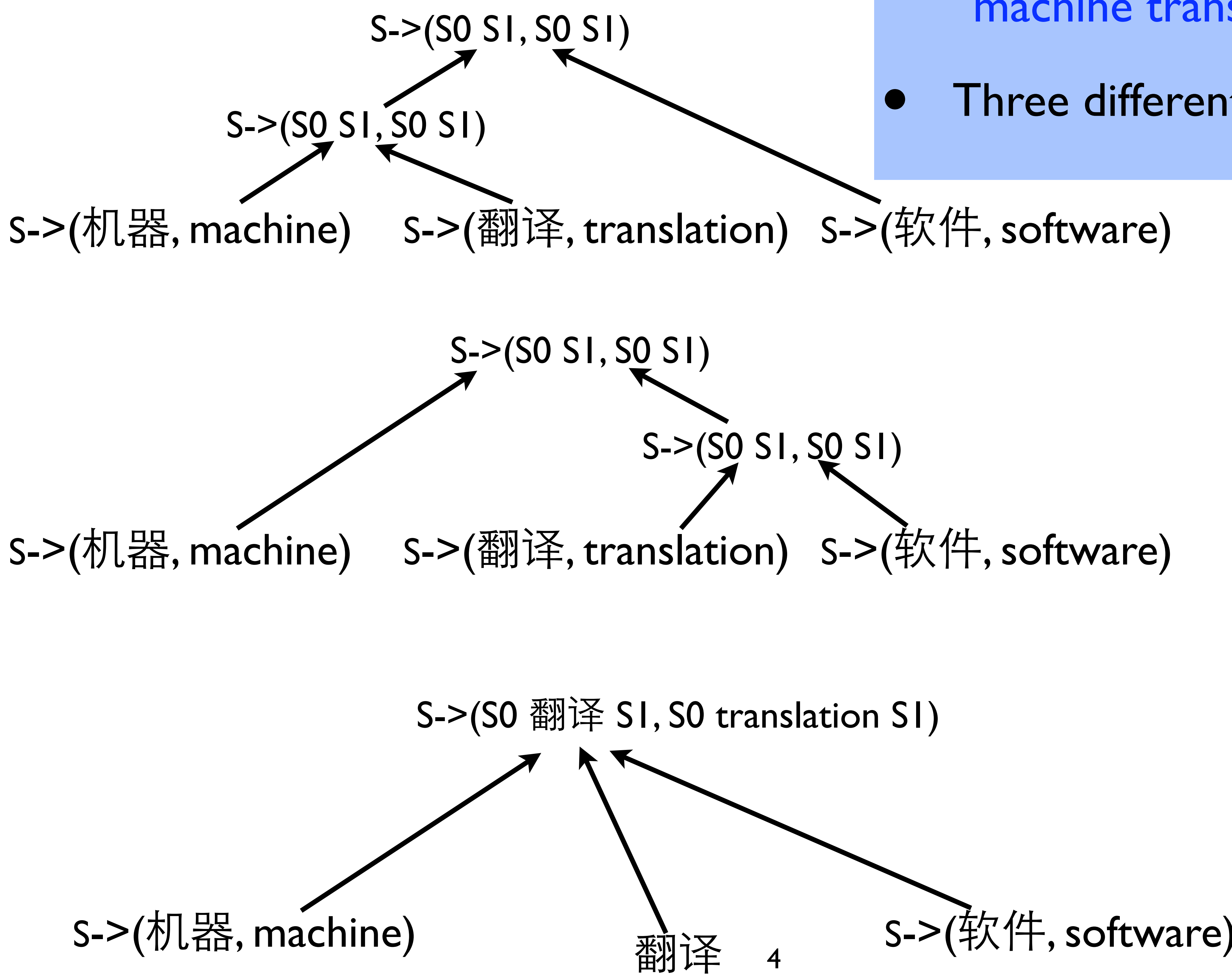
machine translation software

machine transfer software

Spurious Ambiguity in Derivation Trees

机器 翻译 软件

- Same output:
  “machine translation software”
- Three different derivation trees



Maximum A Posterior (MAP) Decoding

translation string

red translation

blue translation

green translation

MAP

derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

- Exact MAP decoding

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

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

Maximum A Posterior (MAP) Decoding

translation string	MAP
red translation	0.28
blue translation	
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

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

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

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

Maximum A Posterior (MAP) Decoding

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

derivation	probability
red	0.16
blue	0.14
blue	0.14
green	0.13
red	0.12
green	0.11
green	0.10
green	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

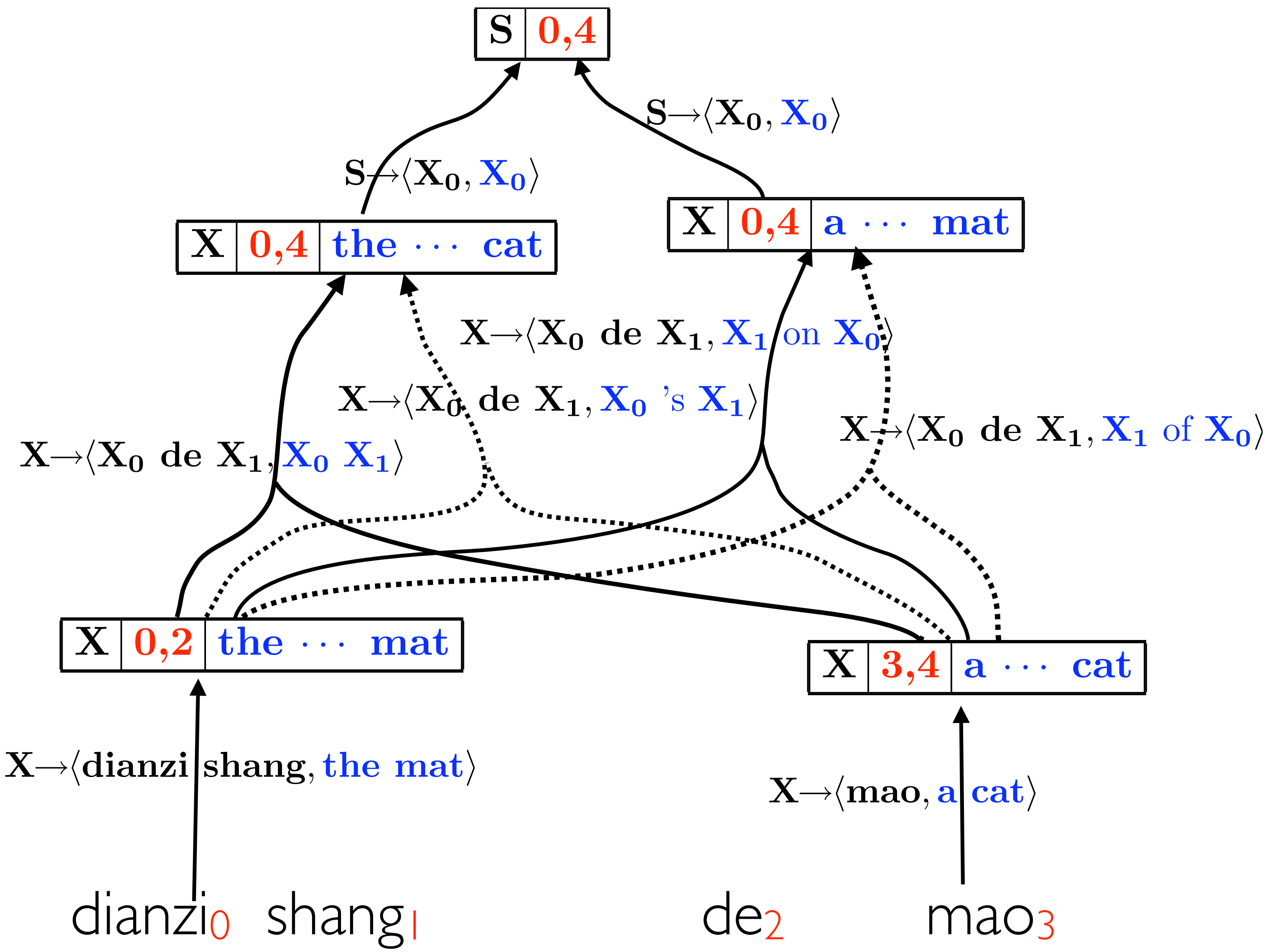
$$y^* = \arg\max_{y \in \mathrm{Trans}(x)} p(y|x)$$

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

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

Hypergraph as a search space

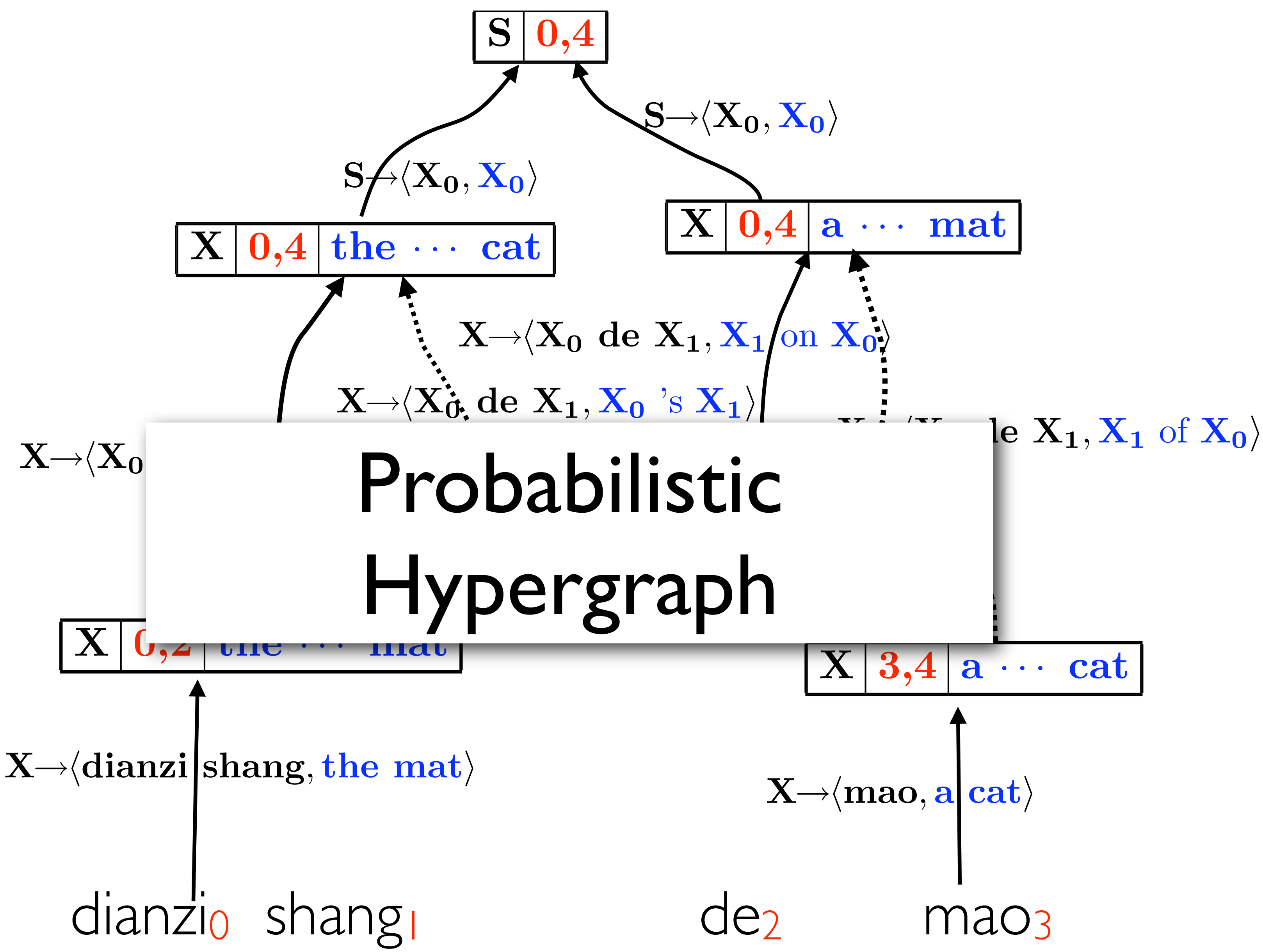
A hypergraph is a compact structure to encode exponentially many trees.



Hypergraph as a search space

The hypergraph defines a probability distribution over **derivation trees**, i.e. p(y, d | x),

and also a distribution (implicit) over **strings**, i.e. p(y | x).



- Exact MAP decoding

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

exponential size

NP-hard (Sima’an 1996)

Decoding with spurious ambiguity?

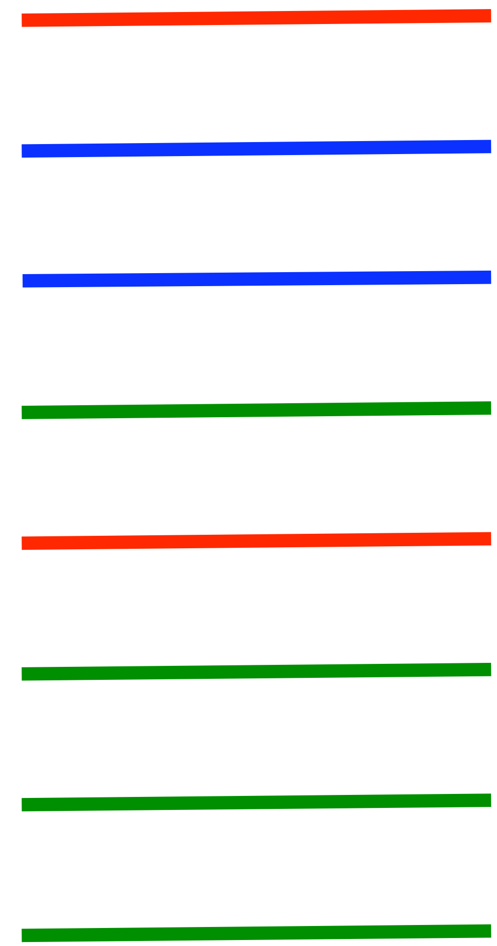
- Maximum a posterior (MAP) decoding

- Viterbi approximation

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

Viterbi Approximation

translation string	MAP	Viterbi
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

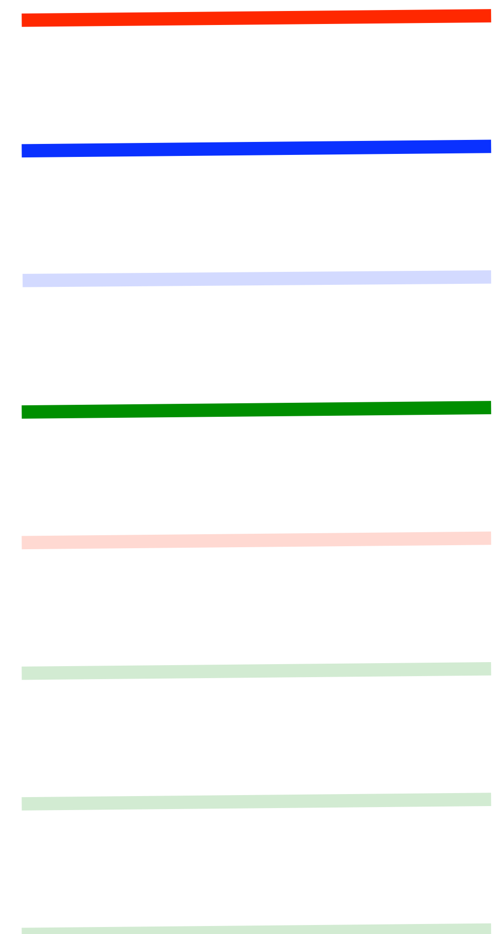
- Viterbi approximation

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

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

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

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

N-best Approximation

translation string	MAP	Viterbi	4-best crunching
red translation	0.28	0.16	
blue translation	0.28	0.14	
green translation	0.44	0.13	

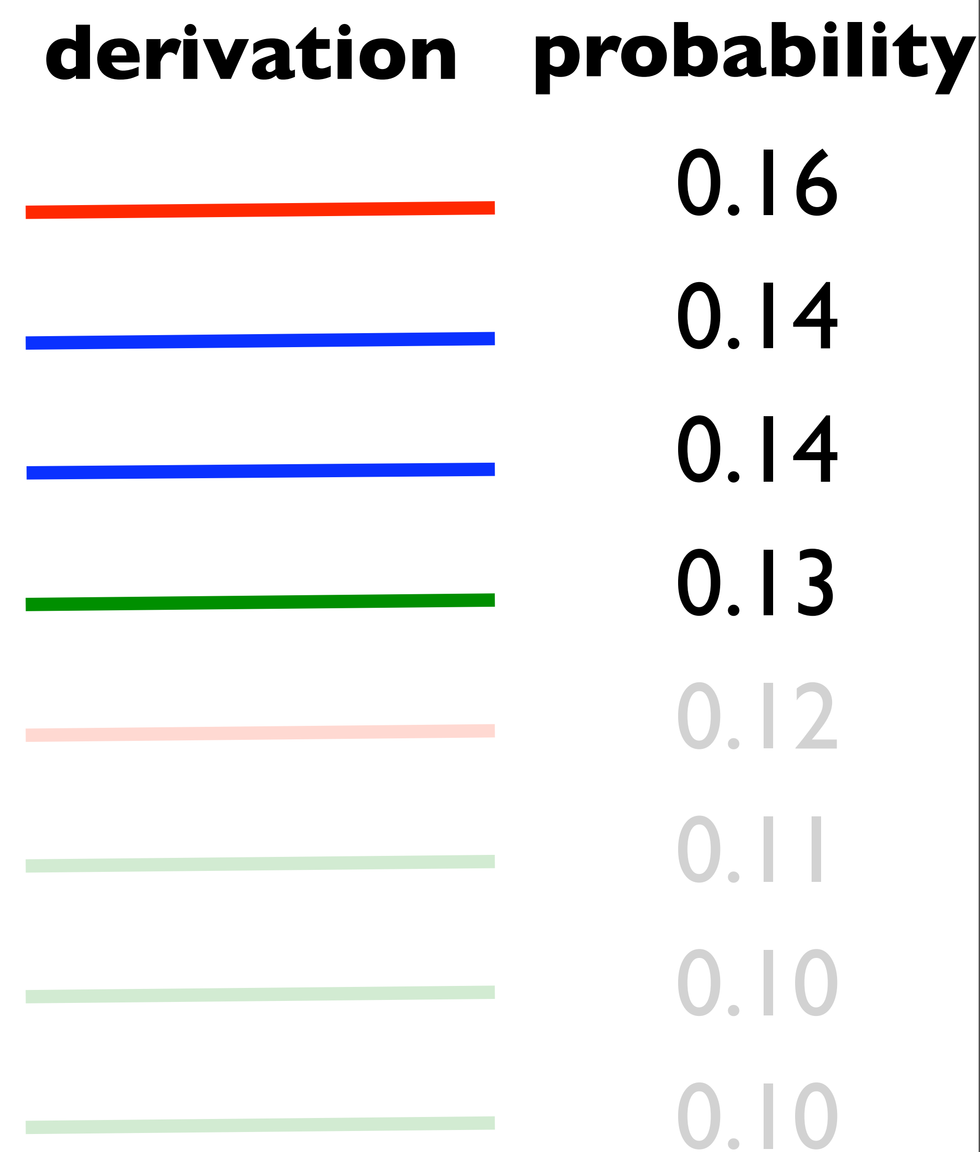
derivation	probability
	0.16
	0.14
	0.14
	0.13
	0.12
	0.11
	0.10
	0.10

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

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

N-best Approximation

translation string	MAP	Viterbi	4-best crunching
red translation	0.28	0.16	0.16
blue translation	0.28	0.14	0.28
green translation	0.44	0.13	0.13



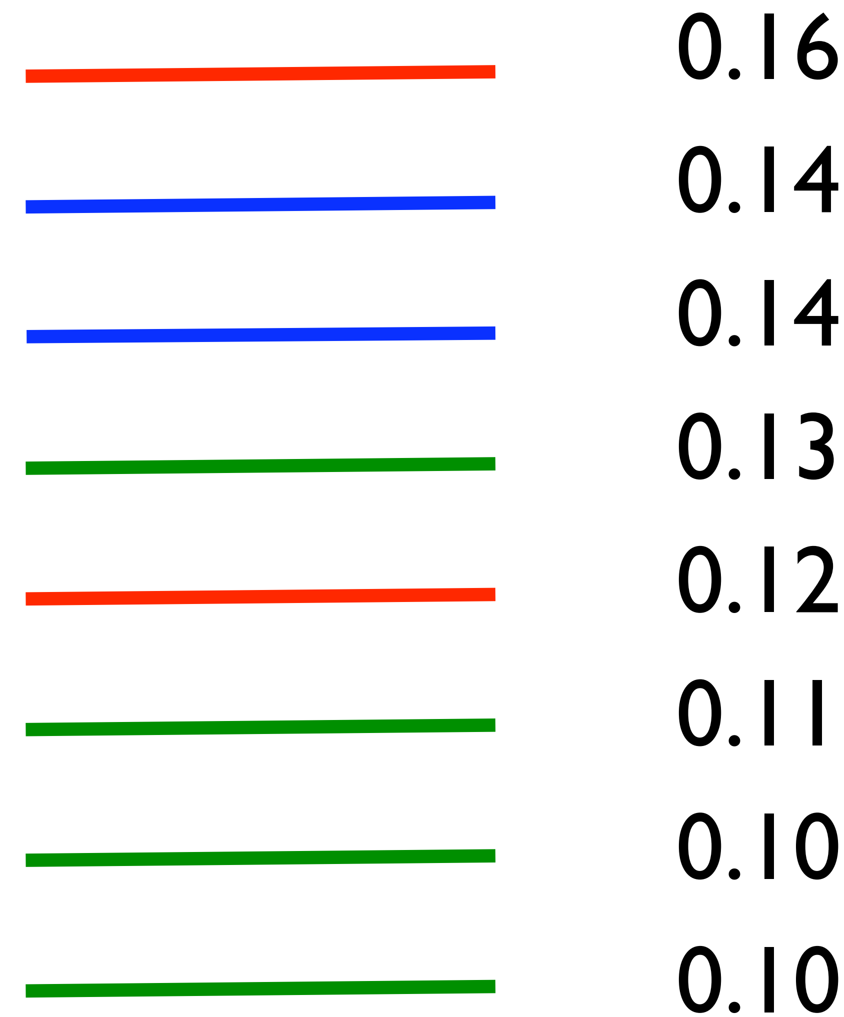
- N-best approximation (crunching) (May and Knight 2006)

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

MAP vs. Approximations

translation string	MAP	Viterbi	4-best crunching
red translation	0.28	0.16	0.16
blue translation	0.28	0.14	0.28
green translation	0.44	0.13	0.13

derivation **probability**



- Exact MAP decoding under spurious ambiguity is intractable
- 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

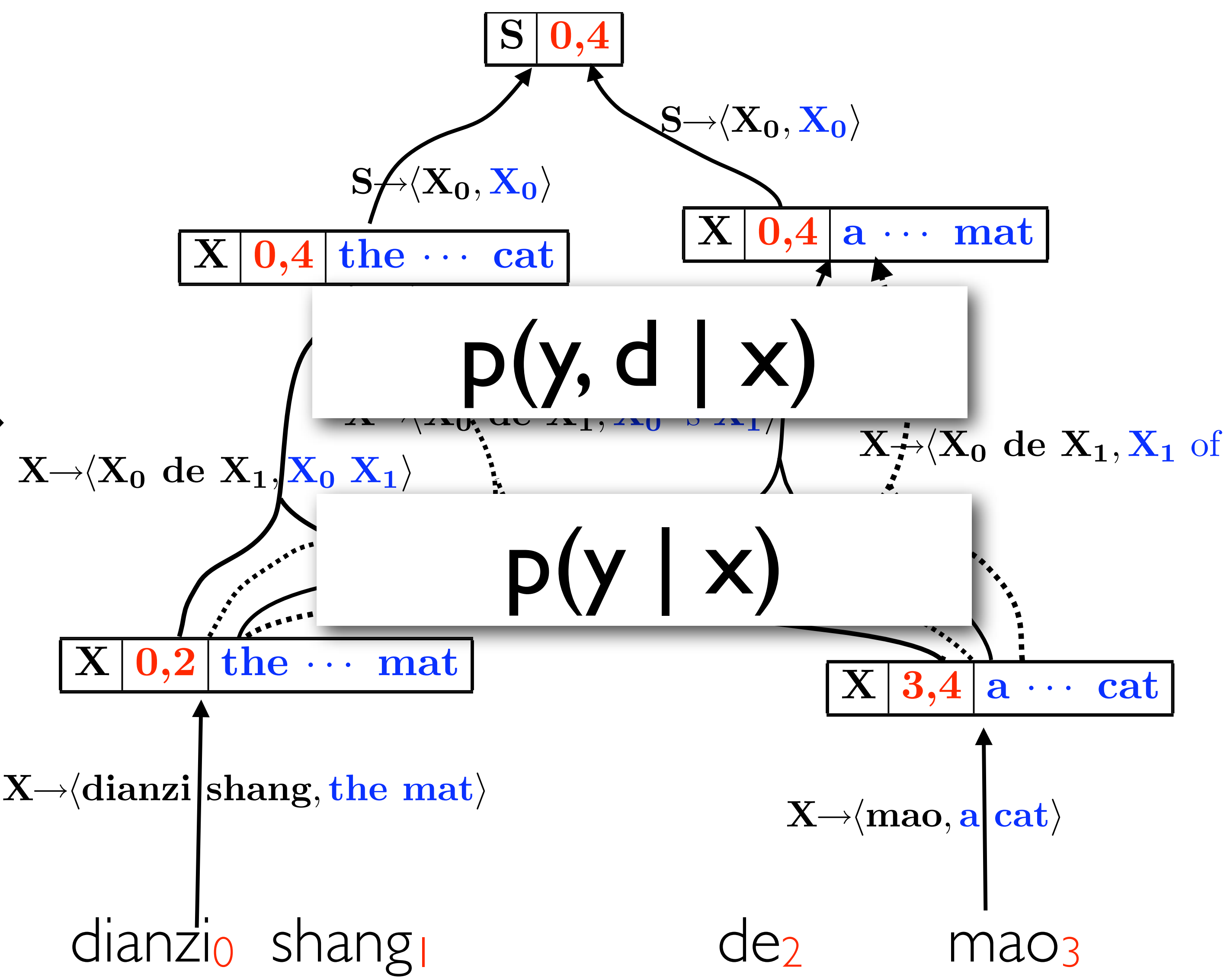
Sentence-specific decoding

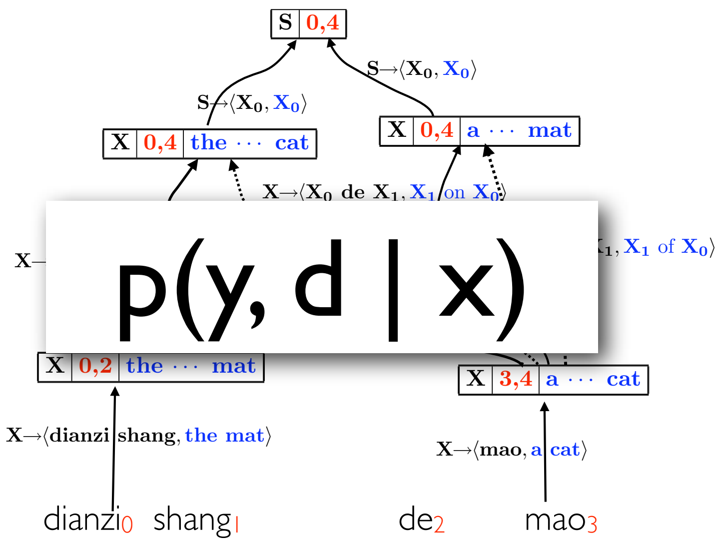
MAP decoding under P is intractable

Three steps:

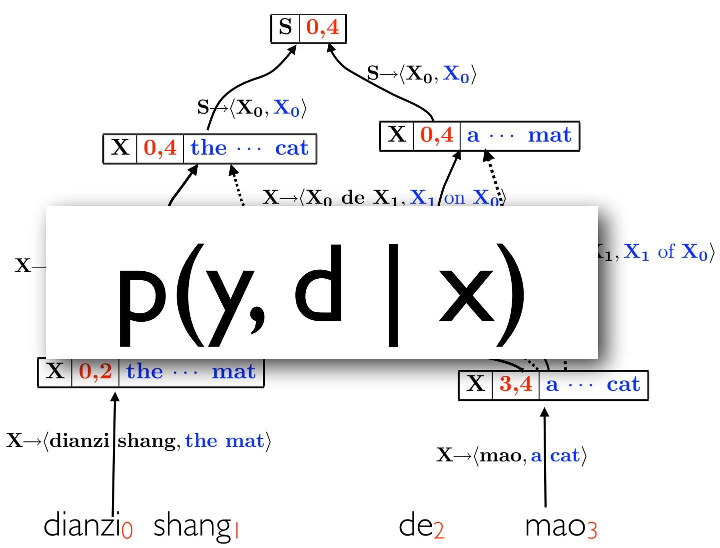
1 Generate a hypergraph

Foreign sentence x ⇨ SMT ⇨





Generate a hypergraph

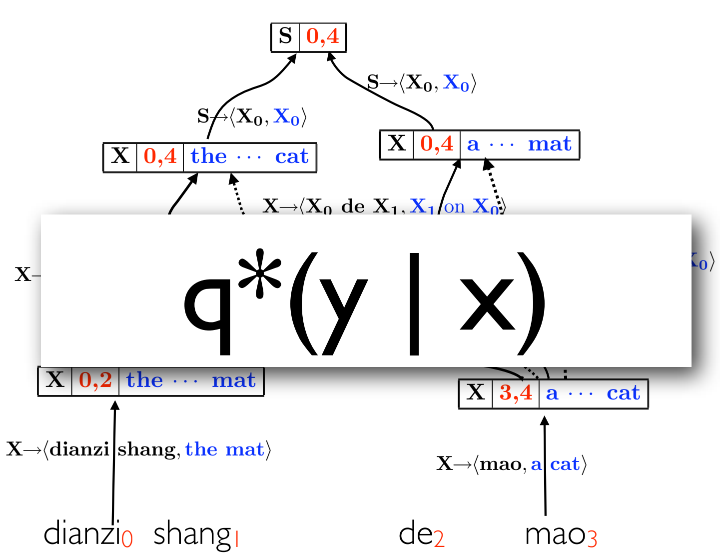


Estimate a model from the hypergraph

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

$$q^*(y \mid x)$$

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



Decode using q* on the hypergraph

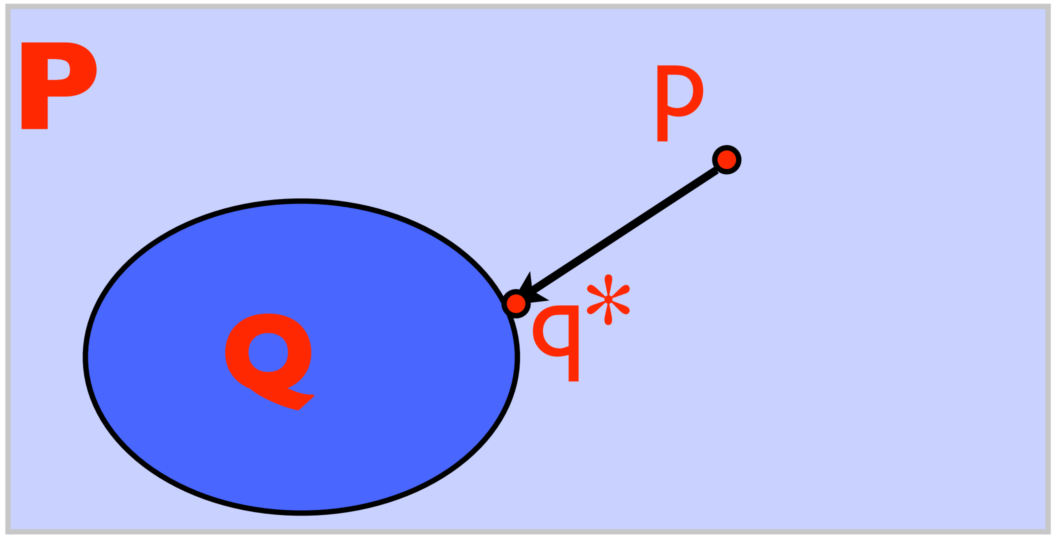
Variational Inference

- 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∈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?

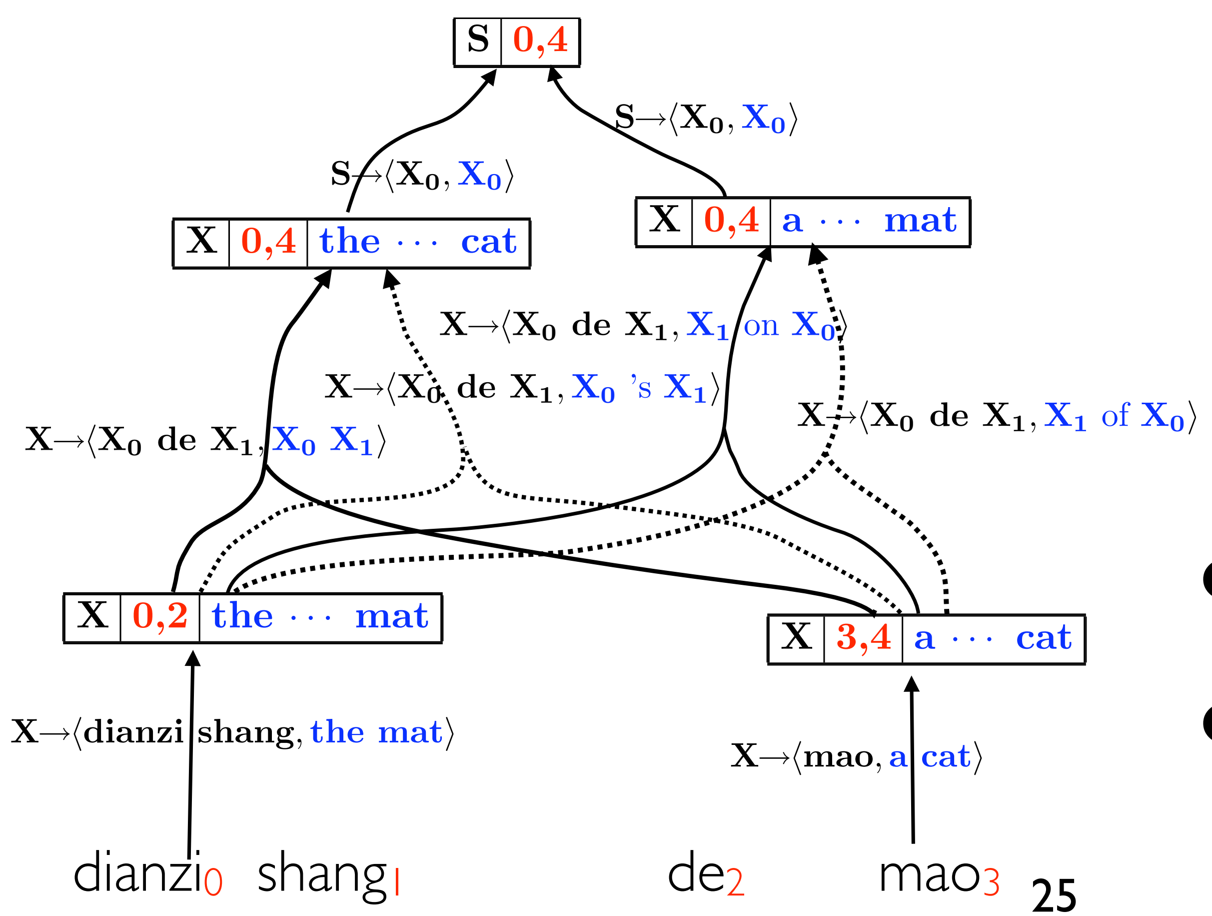
Estimation of q*∈Q

- 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

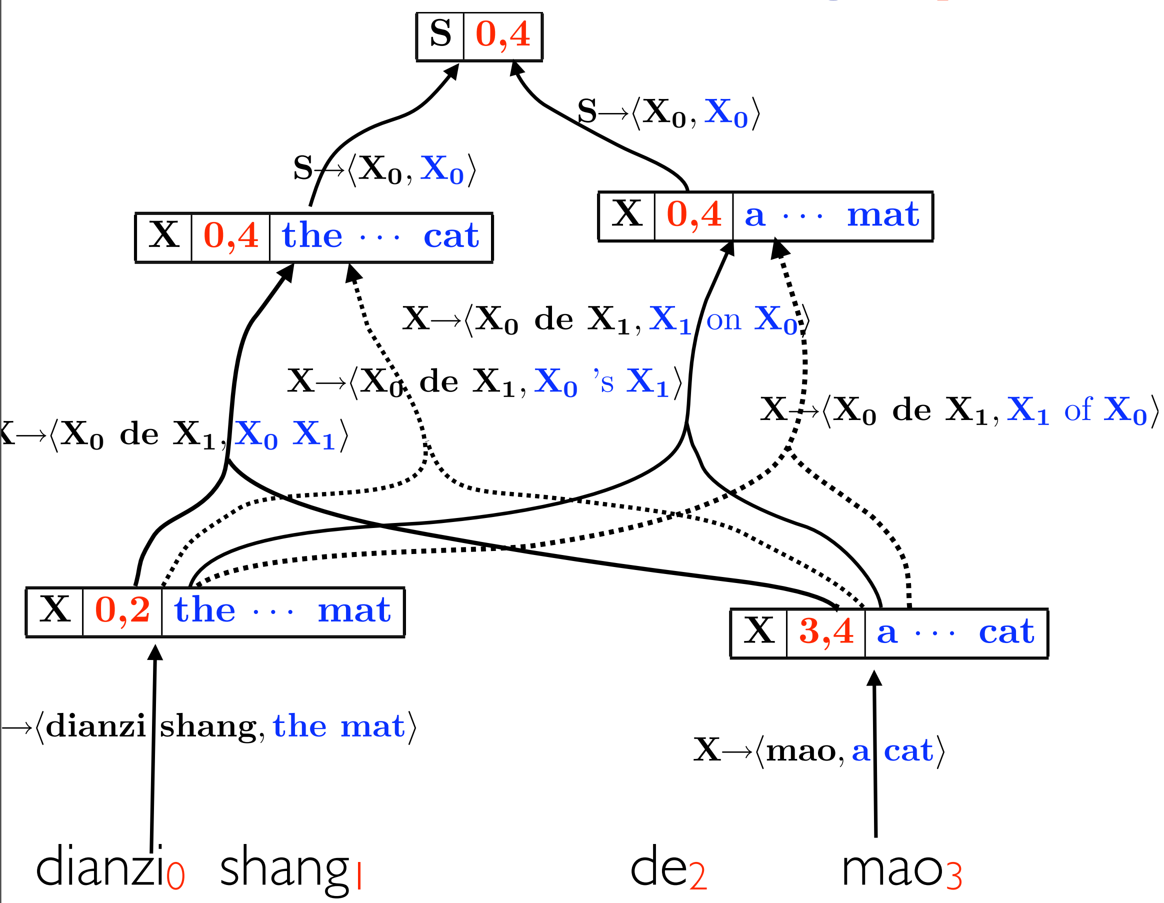
But in our case, p is defined **not** by a **corpus**, but by a **hypergraph** for a given test sentence!



estimate ⟹ bi-gram model

- brute force
- dynamic programming

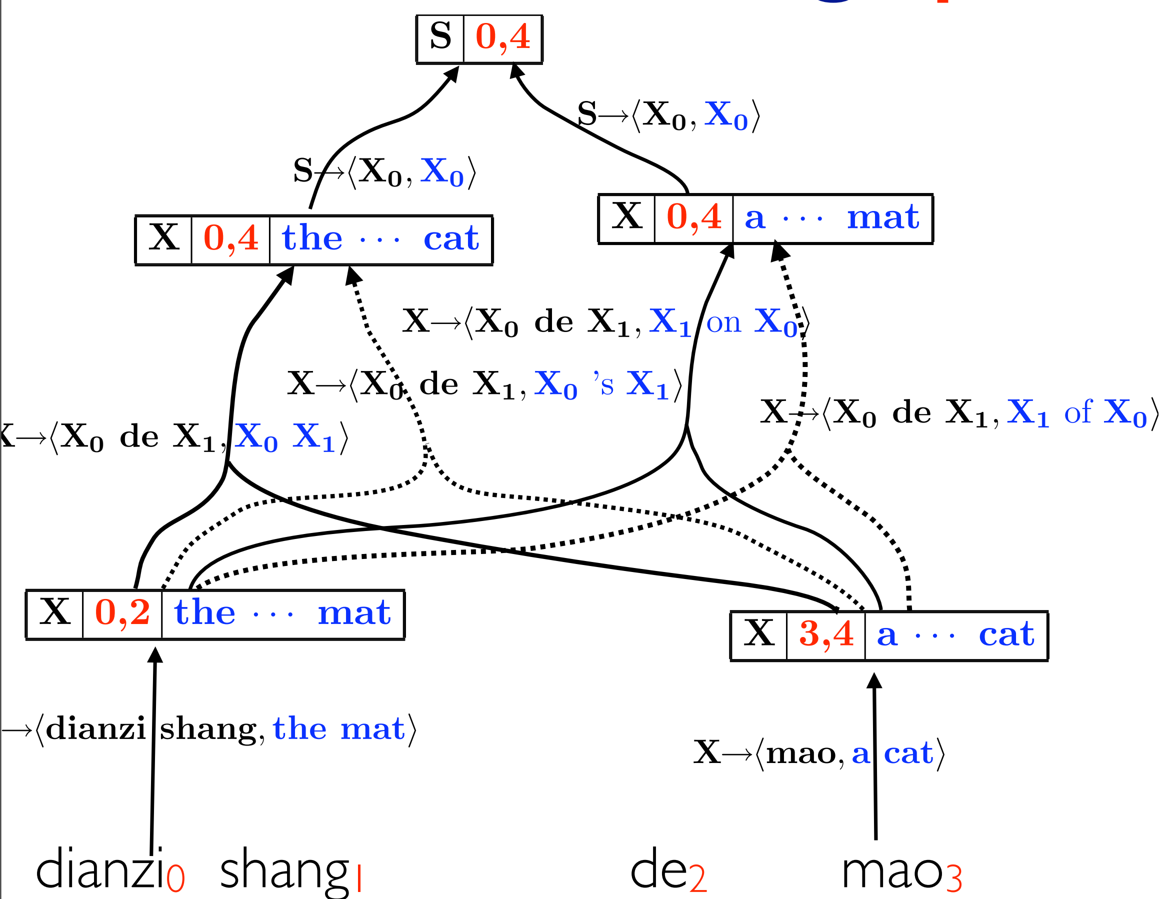
Estimating q* from a hypergraph: brute force



Bi-gram estimation:

- unpack the hypergraph

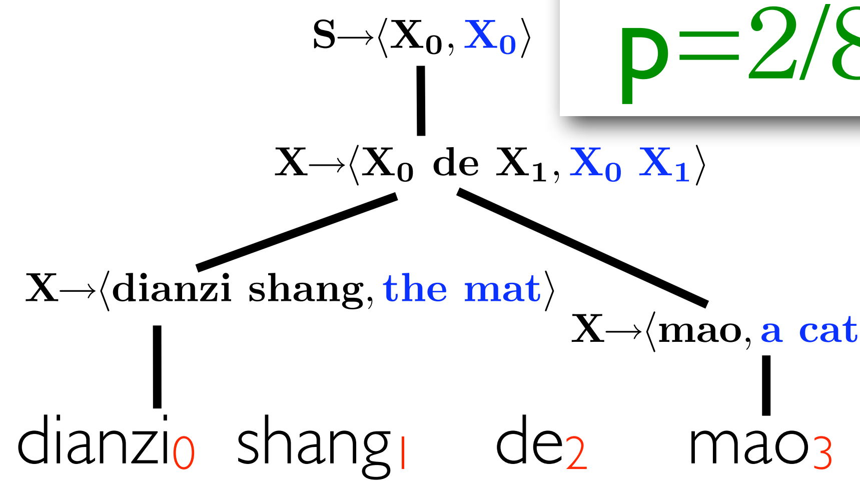
Estimating q* from a hypergraph: brute force



Bi-gram estimation:

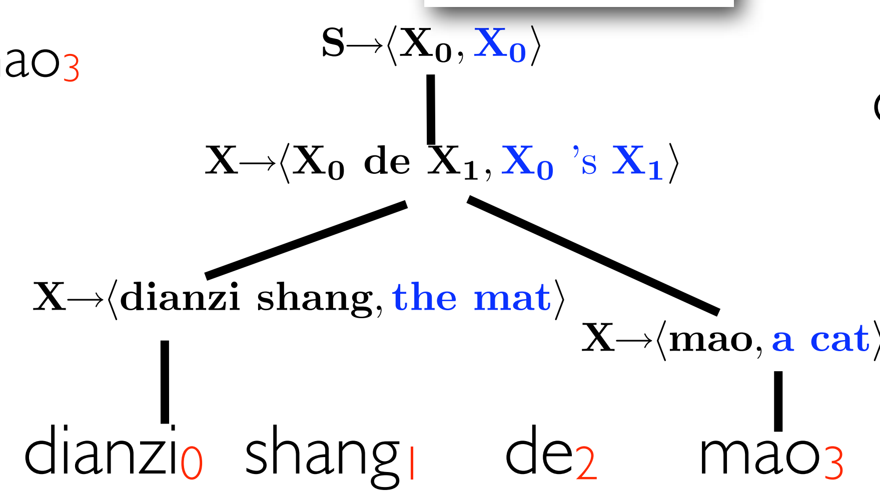
- unpack the hypergraph

p=2/8



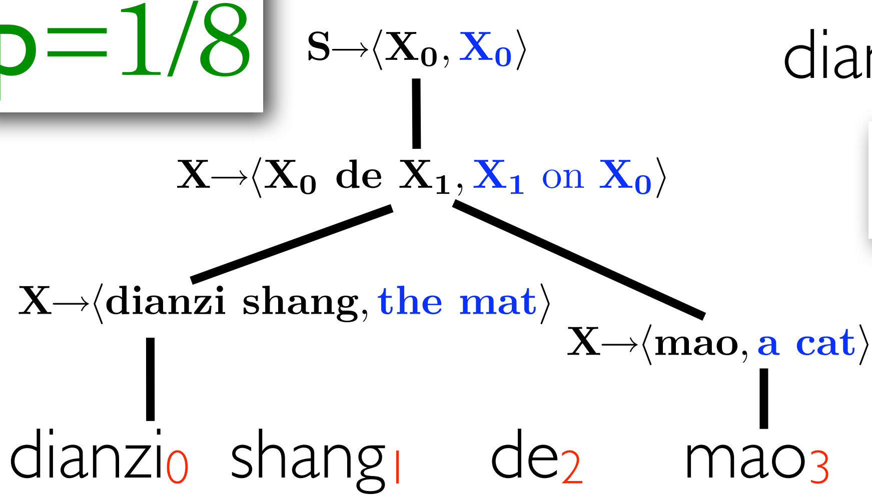
the mat a cat

p=3/8



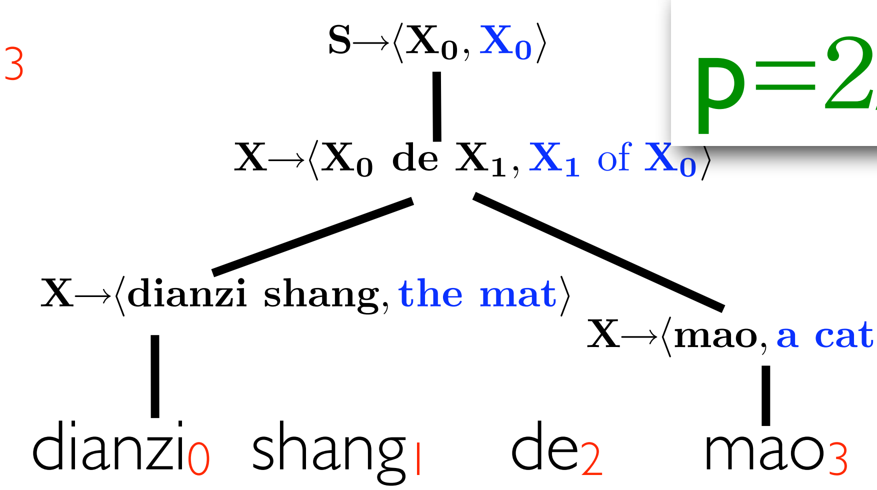
the mat 's a cat

p=1/8



a cat on the mat

p=2/8



a cat of the mat

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:

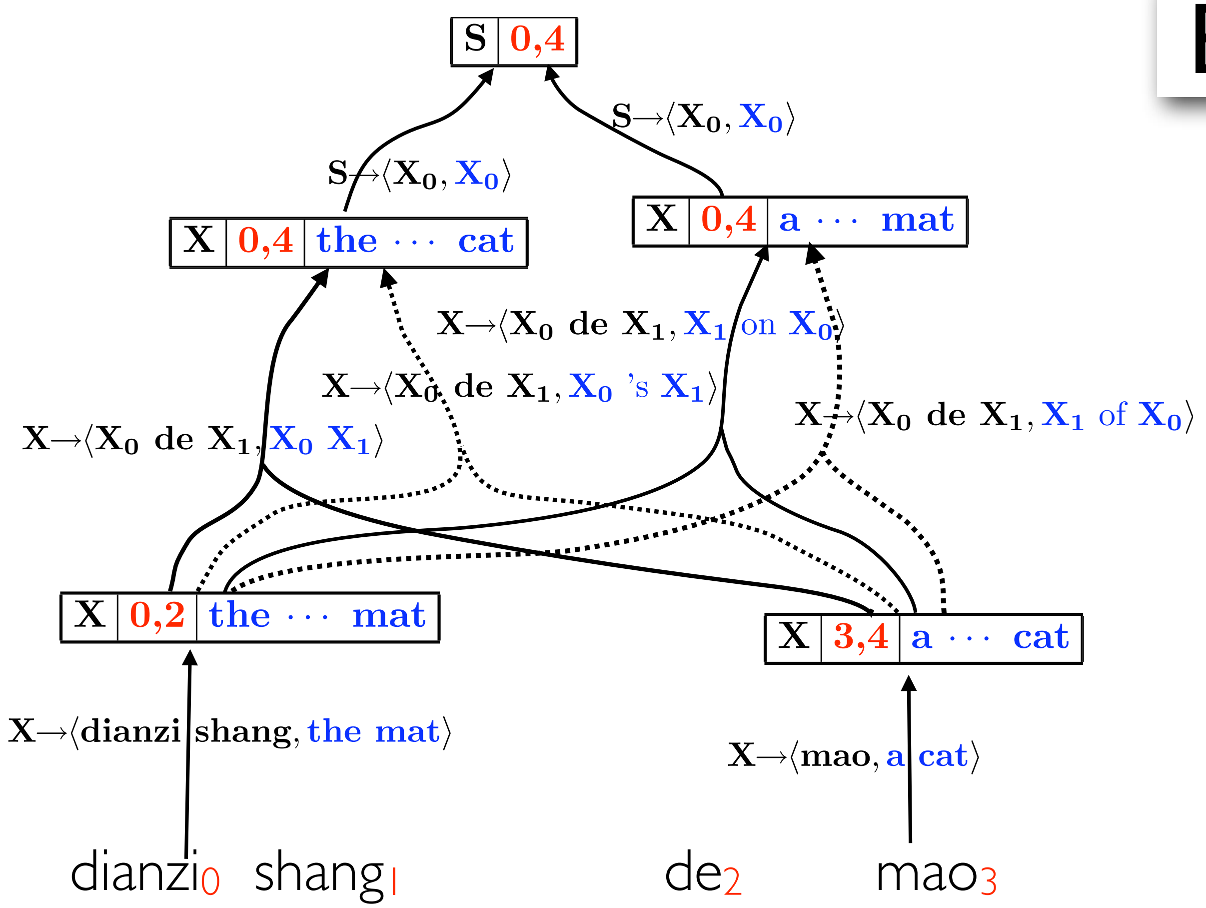
- unpack the hypergraph
- accumulate the **soft-count** of each bigram
- normalize the counts

Pr(on | cat)=1/8

Pr(</s> | cat)=5/8

Pr(of | cat)=2/8

Estimating q* from a hypergraph: dynamic programming



Bi-gram estimation:

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

Decoding using q*∈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

John already told you how to do this ☺

KL divergences under different variational models

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

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

- The larger the order n is, the smaller the KL divergence is!
- The reduction of KL divergence happens mostly when switching from unigram to bigram

KL divergences under different variational models

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

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 them on a **hypergraph**?

see (Li and Eisner, EMNLP'09)

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

- Interpolating variational n-gram model for different n

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

Viterbi and variational are different ways in approximating p

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

Minimum Bayes Risk (MBR) decoding?

(Tromble et al. 2008)

(Denero et al. 2009)

Minimum Risk Decoding

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

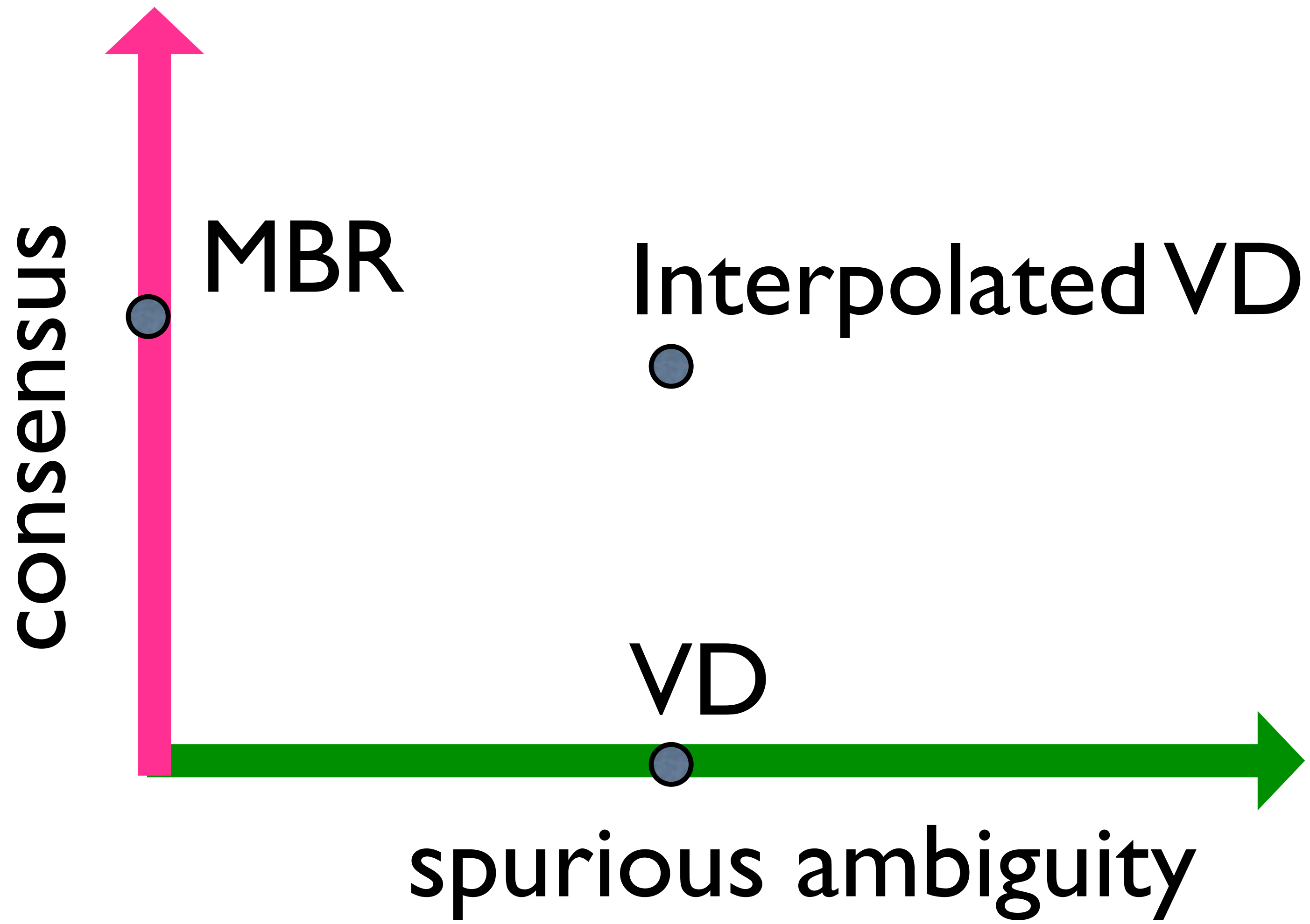
$$y^* = \arg\max_{y \in \mathrm{HG}(x)} p(y|x)$$

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

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

$$\mathrm{Risk}(y) = \sum_{y'} L(y, y')p(y'|x)$$

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 Tasks

Decoding scheme	MT'04	MT'05
Viterbi	35.4	32.6
MBR (K=1000)	35.8	32.7
Crunching (N=10000)	35.7	32.8
Crunching+MBR (N=10000)	35.8	32.7
Variational (1to4gram+wp+vt)	**36.6**	**33.5**

- variational decoding improves over Viterbi, MBR, and crunching

Conclusions

- 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

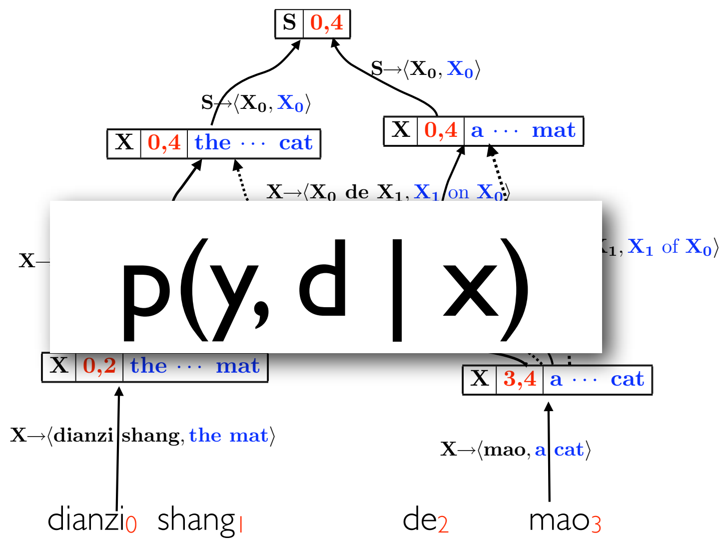
Future directions

- The MT pipeline is full of intractable problems
  - variational approximation is a principled way to tackle these problems
- 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)
  - ......

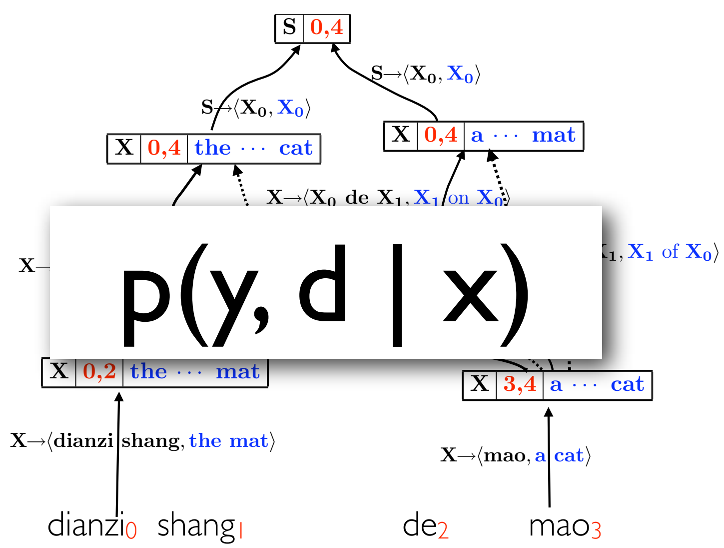
Thank you!

谢谢！


Joshua



1. Generate a hypergraph

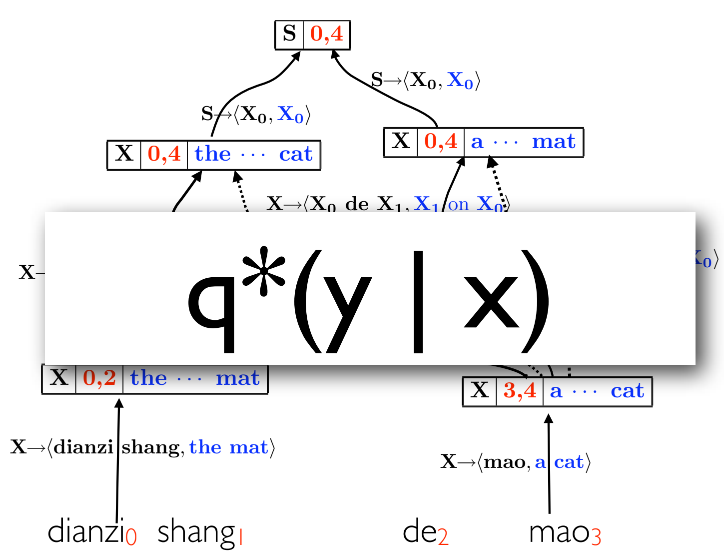


2. Estimate a model from the hypergraph ⟹

$q^*(y \mid x)$

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

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



3. Decode using q^* on the hypergraph