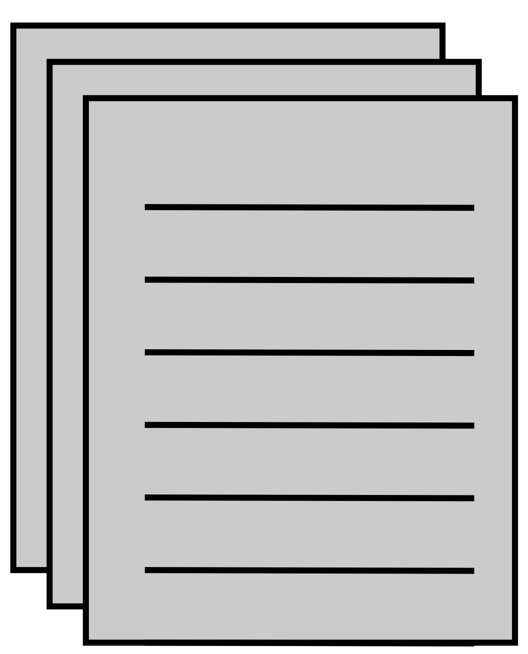
Phrase-Based Machine Translation

cat ||| chat ||| 0.9
the cat ||| le chat ||| 0.8
dog ||| chien ||| 0.8
house ||| maison ||| 0.6
my house ||| ma maison ||| 0.9
language ||| langue ||| 0.9
…

Phrase table P(f|e)



Language model P(e)

Unlabeled English data

$$P(e|f) \propto P(f|e)P(e)$$

Noisy channel model: combine scores from translation model + language model to translate foreign to English

"Translate faithfully but make fluent English"

Phrase-Based Machine Translation

- Noisy channel model: P(**e**|**f**) ∝ P(**f**|**e**) P(**e**) (ignore P(**f**) term)

  Translation model (TM) Language model (LM)

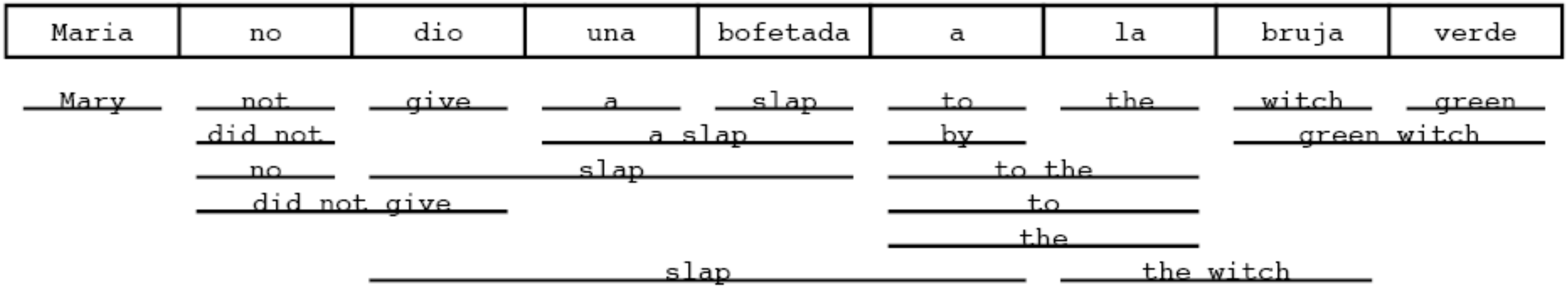
- Inputs needed
  - Language model that scores $P(e_i|e_1,\ldots,e_{i-1}) \approx P(e_i|e_{i-n-1},\ldots,e_{i-1})$
  - Phrase table: set of phrase pairs (**e**, **f**) with probabilities P(**f**|**e**)
- What we want to find: **e** produced by a series of phrase-by-phrase translations from an input **f**

Phrase Lattice

- Given an input sentence, look at our phrase table to find all possible translations of all possible spans
- Monotonic translation: need to translate each word in order, explore paths in the lattice that don't skip any words
- Looks like Viterbi, but the scoring is more complicated

Koehn (2004)

Monotonic Translation

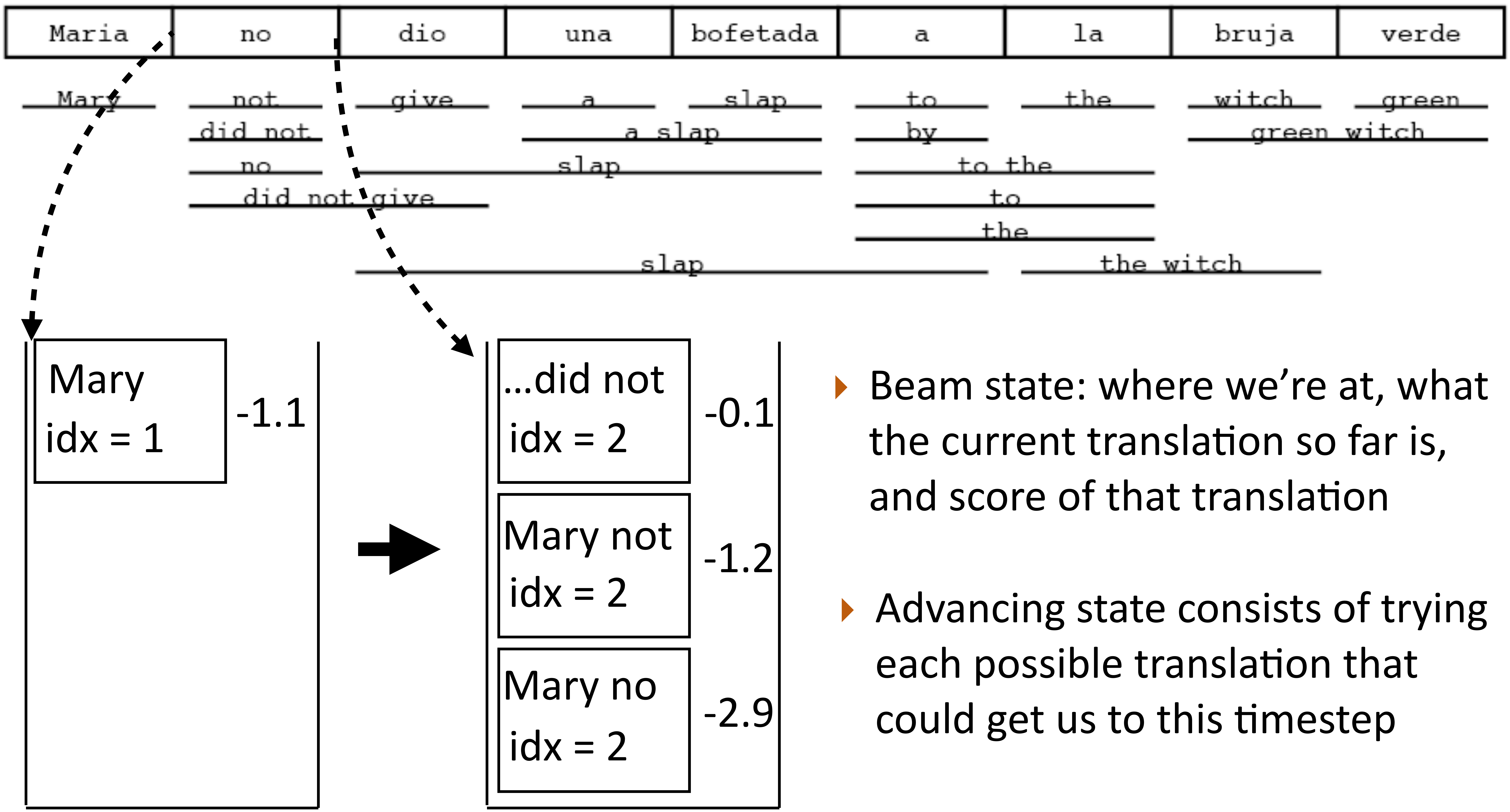


- If we translate with beam search, what state do we need to keep in the beam?
  - Score

$$\arg\max_{\mathbf{e}} \left[\prod_{\langle \bar{e}, \bar{f} \rangle} P(\bar{f}|\bar{e}) \cdot \prod_{i=1}^{|\mathbf{e}|} P(e_i|e_{i-1}, e_{i-2}) \right]$$

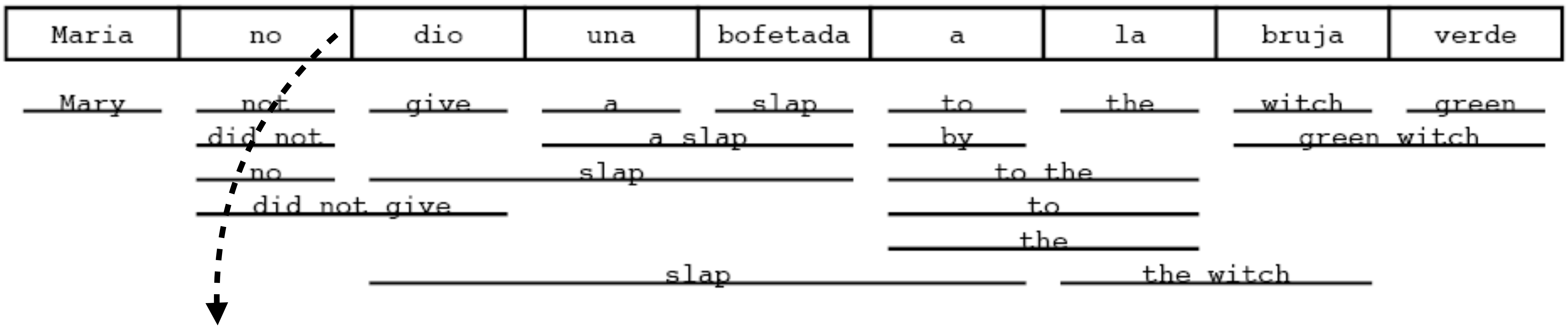
  - Where are we in the sentence
  - What words have we produced so far (actually only need to remember the last 2 words when using a 3-gram LM)

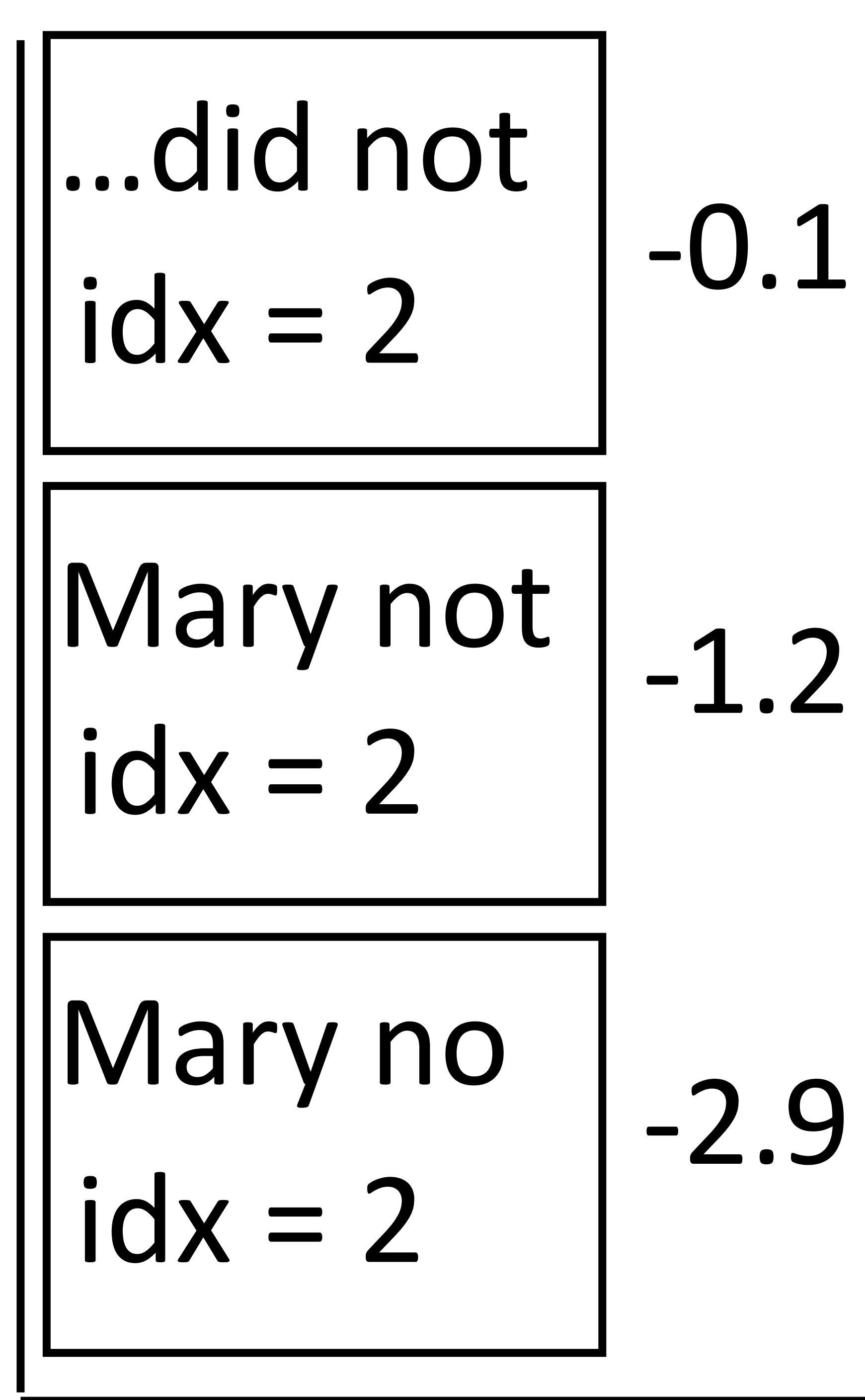
Monotonic Translation



- Beam state: where we're at, what the current translation so far is, and score of that translation
- Advancing state consists of trying each possible translation that could get us to this timestep

Monotonic Translation



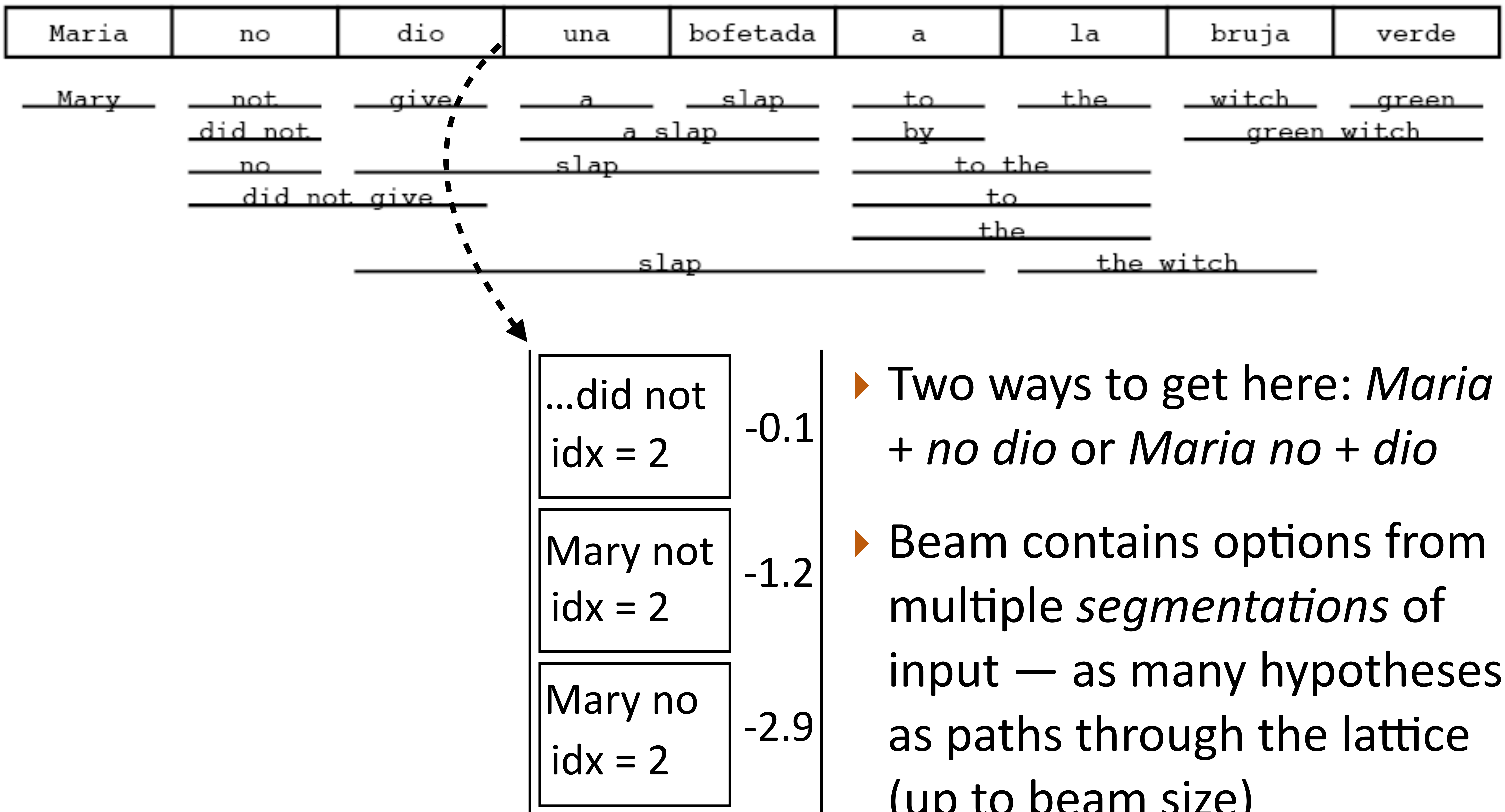


$$\text{score} = \log\,[\underbrace{P(\text{Mary})\;P(\text{not} \mid \text{Mary})}_{\text{LM}}\;\underbrace{P(\text{Maria} \mid \text{Mary})\;P(\text{no} \mid \text{not})}_{\text{TM}}]$$

In reality: score = $\alpha \log P(\text{LM}) + \beta \log P(\text{TM})$

…and TM is broken down into several features

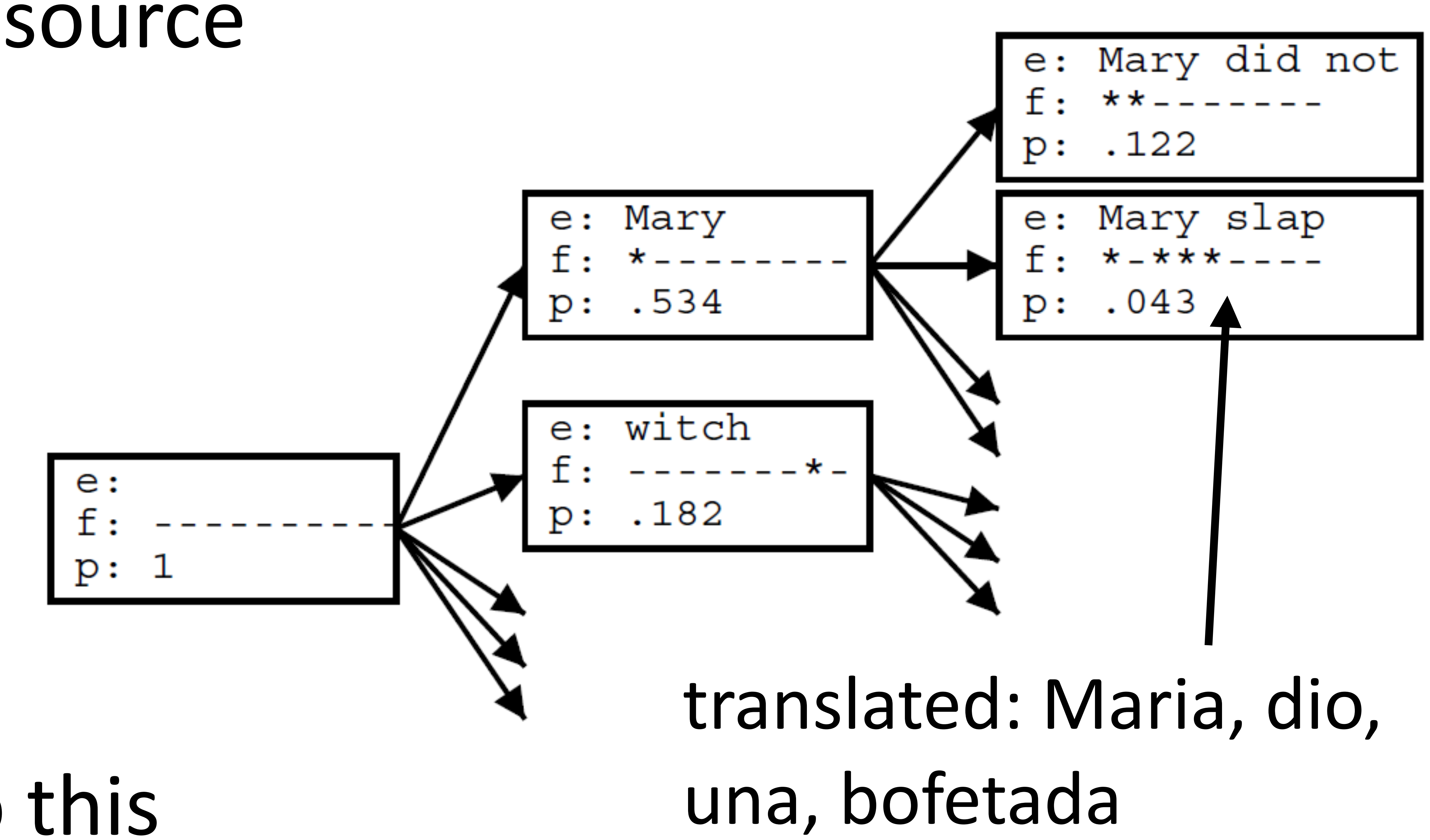
Monotonic Translation



- Two ways to get here: *Maria* + *no dio* or *Maria no* + *dio*
- Beam contains options from multiple *segmentations* of input — as many hypotheses as paths through the lattice (up to beam size)

Non-Monotonic Translation

- More flexible model: can visit source sentence "out of order"
- State needs to describe which words have been translated and which haven't
- Big enough phrases already capture lots of reorderings, so this isn't as important as you think



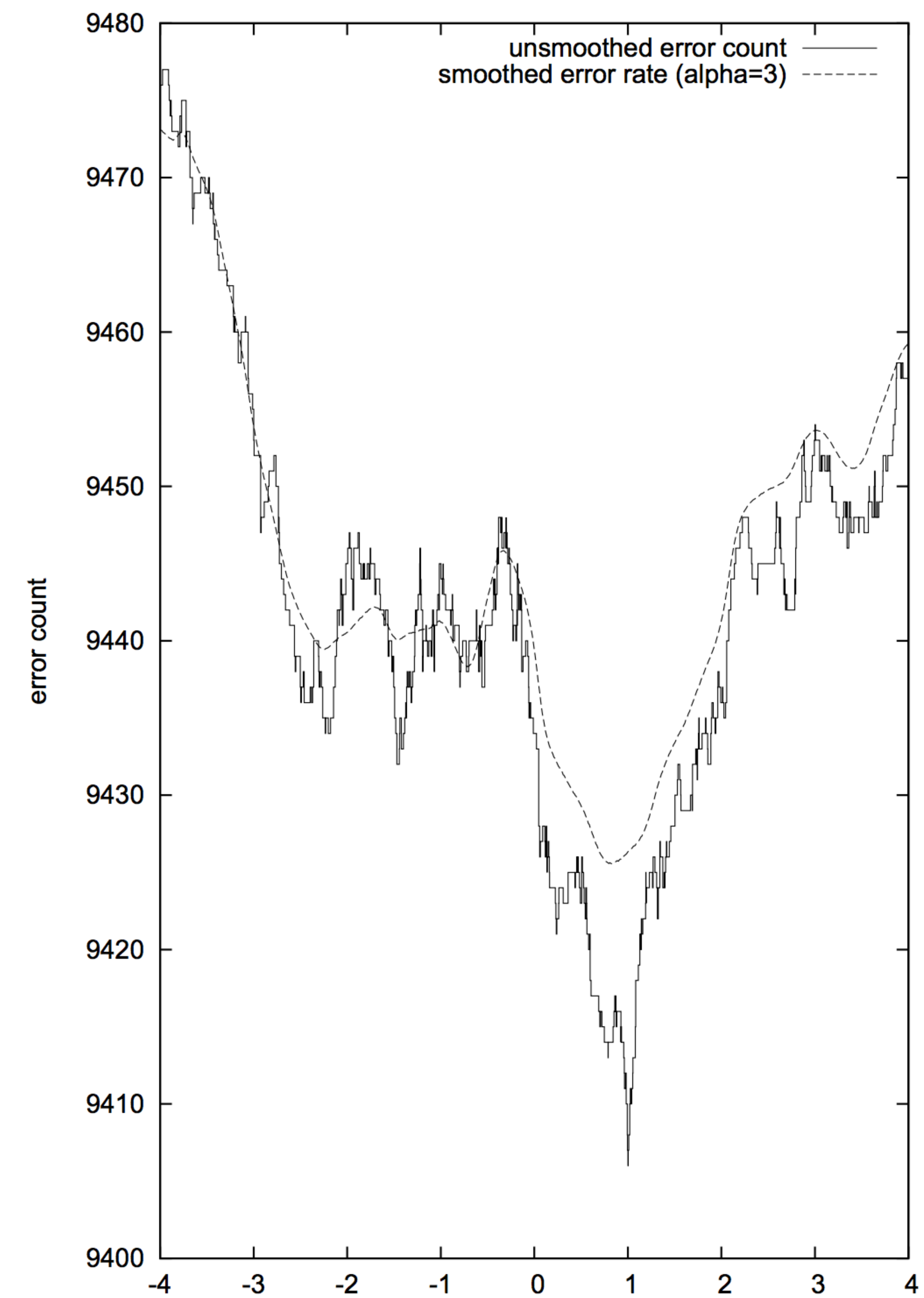
“Training” Decoders

score = α log P(**t**) + β log P(**s**|**t**)

…and P(**s**|**t**) is in fact more complex

- Usually 5-20 feature weights to set, want to optimize for BLEU score which is not differentiable

- MERT (Och 2003): decode to get 1000-best translations for each sentence in a small training set (<1000 sentences), do line search on parameters to directly optimize for BLEU



Moses

- Toolkit for machine translation due to Philipp Koehn + Hieu Hoang
  - Pharaoh (Koehn, 2004) is the decoder from Koehn's thesis

- Moses implements word alignment, language models, and this decoder, plus **a ton** more stuff
  - Highly optimized and heavily engineered, could more or less build SOTA translation systems with this from 2007-2013