Conditional Language Modeling

Chris Dyer

DeepMind

Carnegie Mellon University

Review: Unconditional LMs

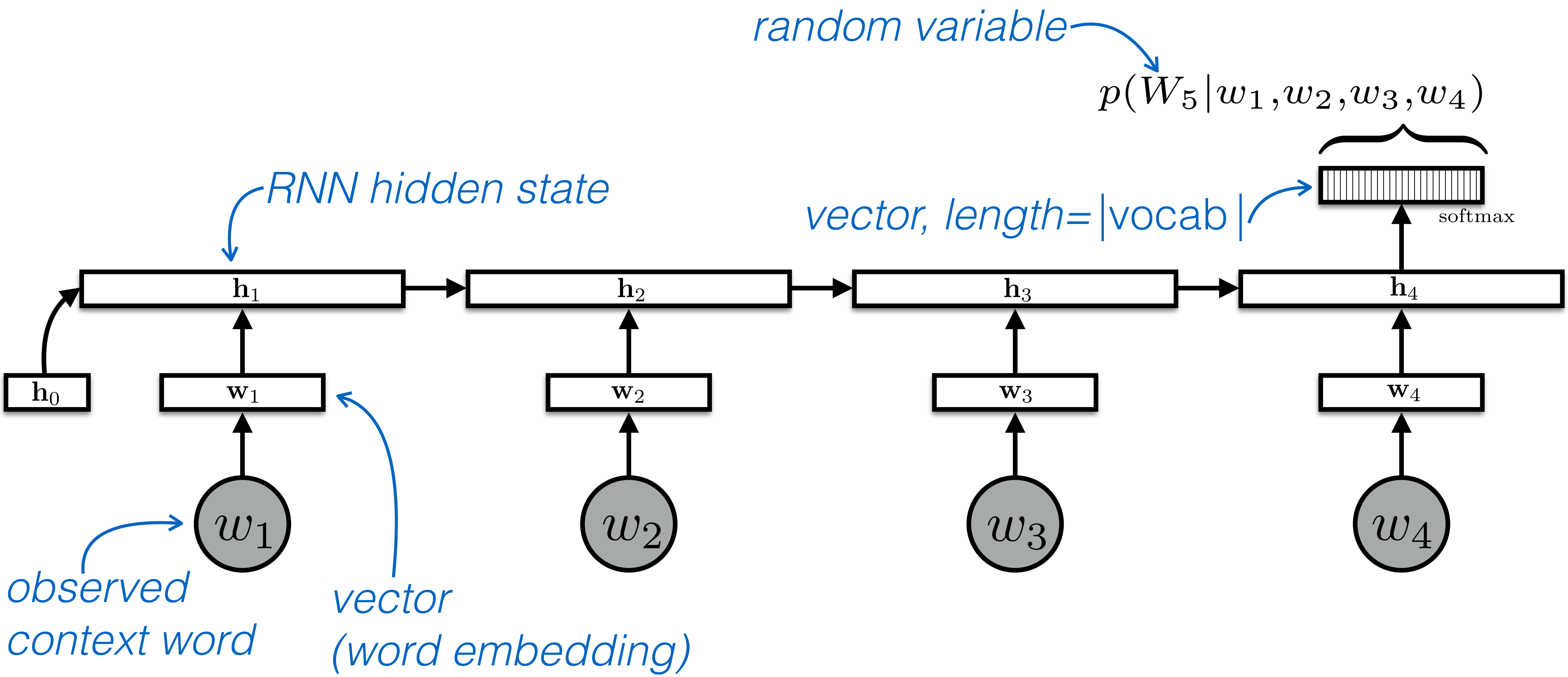
A language model assigns probabilities to sequences of words, $\boldsymbol{w} = (w_1, w_2, \ldots, w_\ell)$.

We saw that it is helpful to decompose this probability using the chain rule, as follows:

$$\begin{aligned} p(\boldsymbol{w}) &= p(w_1) \times p(w_2 \mid w_1) \times p(w_3 \mid w_1, w_2) \times \cdots \times \\ &\qquad p(w_\ell \mid w_1, \ldots, w_{\ell-1}) \\ &= \prod_{t=1}^{|\boldsymbol{w}|} p(w_t \mid w_1, \ldots, w_{t-1}) \end{aligned}$$

This reduces the language modeling problem to **modeling the probability of the next word**, given the history of preceding words.

Unconditional LMs with RNNs



Conditional LMs

A **conditional language model** assigns probabilities to sequences of words, $\boldsymbol{w} = (w_1, w_2, \ldots, w_\ell)$, given some conditioning context, $\boldsymbol{x}$.

As with unconditional models, it is again helpful to use the chain rule to decompose this probability:

$$p(\boldsymbol{w} \mid \boldsymbol{x}) = \prod_{t=1}^{\ell} p(w_t \mid \boldsymbol{x}, w_1, w_2, \ldots, w_{t-1})$$

What is the probability of the next word, given the history of previously generated words ***and*** *conditioning context* $\boldsymbol{x}$*?*

Conditional LMs

x "input"	w "**text** output"
An author	A document written by that author
A topic label	An article about that topic
{SPAM, NOT_SPAM}	An email
A sentence in French	Its English translation
A sentence in English	Its French translation
A sentence in English	Its Chinese translation
An image	A text description of the image
A document	Its summary
A document	Its translation
Meterological measurements	A weather report
Acoustic signal	Transcription of speech
Conversational history + database	Dialogue system response
A question + a document	Its answer
A question + an image	Its answer

Conditional LMs

x "input"	w "**text** output"
An author	A document written by that author
A topic label	An article about that topic
{SPAM, NOT_SPAM}	An email
A sentence in French	Its English translation
A sentence in English	Its French translation
A sentence in English	Its Chinese translation
An image	A text description of the image
A document	Its summary
A document	Its translation
Meterological measurements	A weather report
Acoustic signal	Transcription of speech
Conversational history + database	Dialogue system response
A question + a document	Its answer
A question + an image	Its answer

Conditional LMs

x "input"	w "**text** output"
An author	A document written by that author
A topic label	An article about that topic
{SPAM, NOT_SPAM}	An email
A sentence in French	Its English translation
A sentence in English	Its French translation
A sentence in English	Its Chinese translation
An image	A text description of the image
A document	Its summary
A document	Its translation
Meterological measurements	A weather report
Acoustic signal	Transcription of speech
Conversational history + database	Dialogue system response
A question + a document	Its answer
A question + an image	Its answer

this week

next week

two weeks

Data for training conditional LMs

To train conditional language models, we need *paired samples*, $\{(\boldsymbol{x}_i, \boldsymbol{w}_i)\}_{i=1}^{N}$.

Data availability varies. It's easy to think of tasks that could be solved by conditional language models, but the data just doesn't exist.

Relatively large amounts of data for:

Translation, summarisation, caption generation, speech recognition

Algorithmic challenges

We often want to find the most likely $\boldsymbol{w}$ given some $\boldsymbol{x}$. This is unfortunately generally an *intractable problem*.

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x})$$

We therefore approximate it using a **beam search** or with Monte Carlo methods since $\boldsymbol{w}^{(i)} \sim p(\boldsymbol{w} \mid \boldsymbol{x})$ is often computationally easy.

Improving search/inference is an open research question.

How can we search more effectively?

Can we get guarantees that we have found the max?

Can we limit the model a bit to make search easier?

Evaluating conditional LMs

How good is our conditional language model?

These are language models, we can use **cross-entropy** or **perplexity**. *okay to implement, hard to interpret*

Task-specific evaluation. Compare the model's most likely output to human-generated expected output using a task-specific evaluation metric L.

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x}) \qquad L(\boldsymbol{w}^*, \boldsymbol{w}_{ref})$$

Examples of L: BLEU, METEOR, WER, ROUGE.
easy to implement, okay to interpret

Human evaluation.
hard to implement, easy to interpret

Evaluating conditional LMs

How good is our conditional language model?

These are language models, we can use **cross-entropy** or **perplexity**. *okay to implement, hard to interpret*

Task-specific evaluation. Compare the model's most likely output to human-generated expected output using a task-specific evaluation metric L.

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x}) \qquad L(\boldsymbol{w}^*, \boldsymbol{w}_{ref})$$

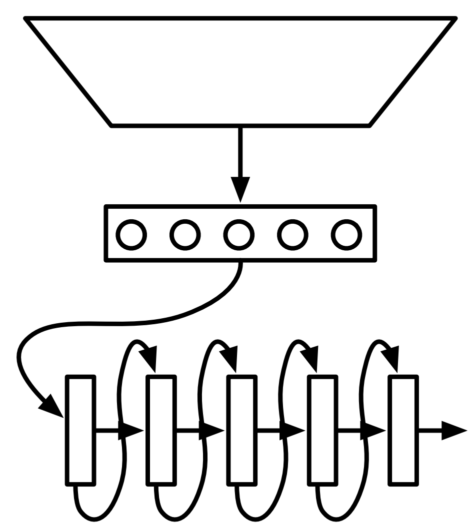
Examples of L: BLEU, METEOR, WER, ROUGE.
easy to implement, okay to interpret

Human evaluation. *hard to implement, easy to interpret*

Lecture overview

The rest of this lecture will look at "encoder-decoder" models that learn a function that maps $\boldsymbol{x}$ into a fixed-size vector and then uses a language model to "decode" that vector into a sequence of words, $\boldsymbol{w}$.

$\boldsymbol{x}$ ***Kunst kann nicht gelehrt werden…***

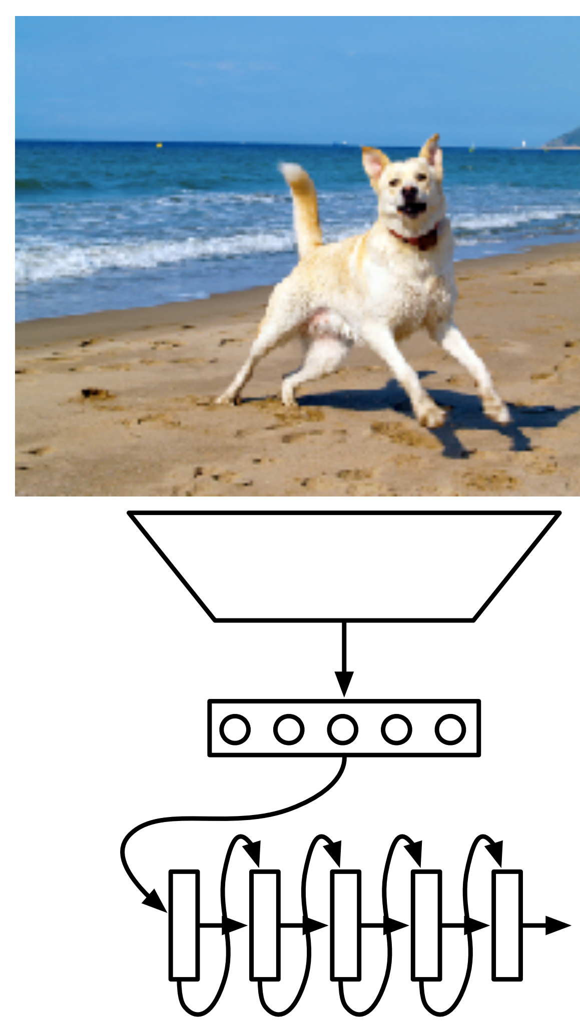


$\boldsymbol{w}$ ***Artistry can't be taught…***

Lecture overview

The rest of this lecture will look at "encoder-decoder" models that learn a function that maps $\boldsymbol{x}$ into a fixed-size vector and then uses a language model to "decode" that vector into a sequence of words, $\boldsymbol{w}$.

$\boldsymbol{x}$



$\boldsymbol{w}$ ***A dog is playing on the beach.***

Lecture overview

- Two questions
  - How do we encode $\boldsymbol{x}$ as a fixed-size vector, $\mathbf{c}$?
    - Problem (or at least modality) specific
    - Think about assumptions
  - How do we condition on $\mathbf{c}$ in the decoding model?
    - Less problem specific
    - We will review solution/architectures

Kalchbrenner and Blunsom 2013

Encoder

$$\mathbf{c} = \text{embed}(\boldsymbol{x})$$
$$\mathbf{s} = \mathbf{V}\mathbf{c}$$

Recurrent decoder

Recurrent connection

Embedding of w_{t-1}

Source sentence

$$\mathbf{h}_t = g(\mathbf{W}[\mathbf{h}_{t-1}; \mathbf{w}_{t-1}] + \mathbf{s} + \mathbf{b}])$$
$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}'$$

Learnt bias

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}) = \text{softmax}(\mathbf{u}_t)$$

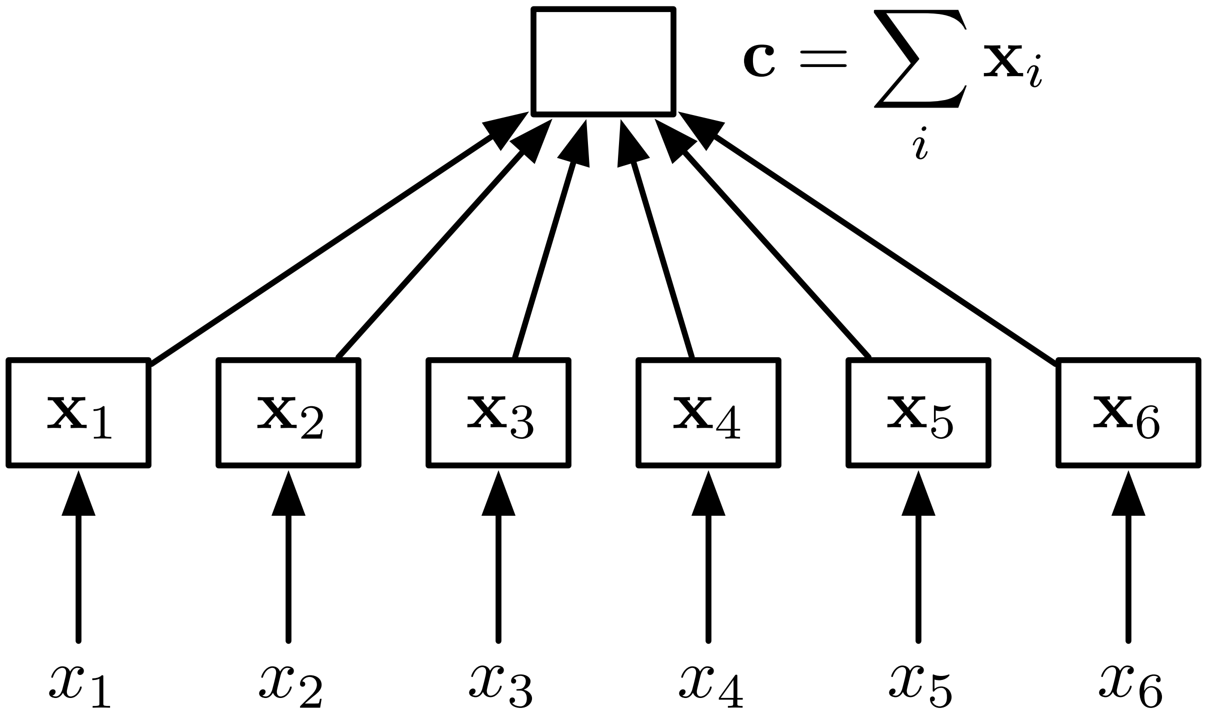
Recall unconditional RNN

$$\mathbf{h}_t = g(\mathbf{W}[\mathbf{h}_{t-1}; \mathbf{w}_{t-1}] + \mathbf{b}])$$

K&B 2013: Encoder

How should we define $\mathbf{c} = \text{embed}(\boldsymbol{x})$?

The simplest model possible:

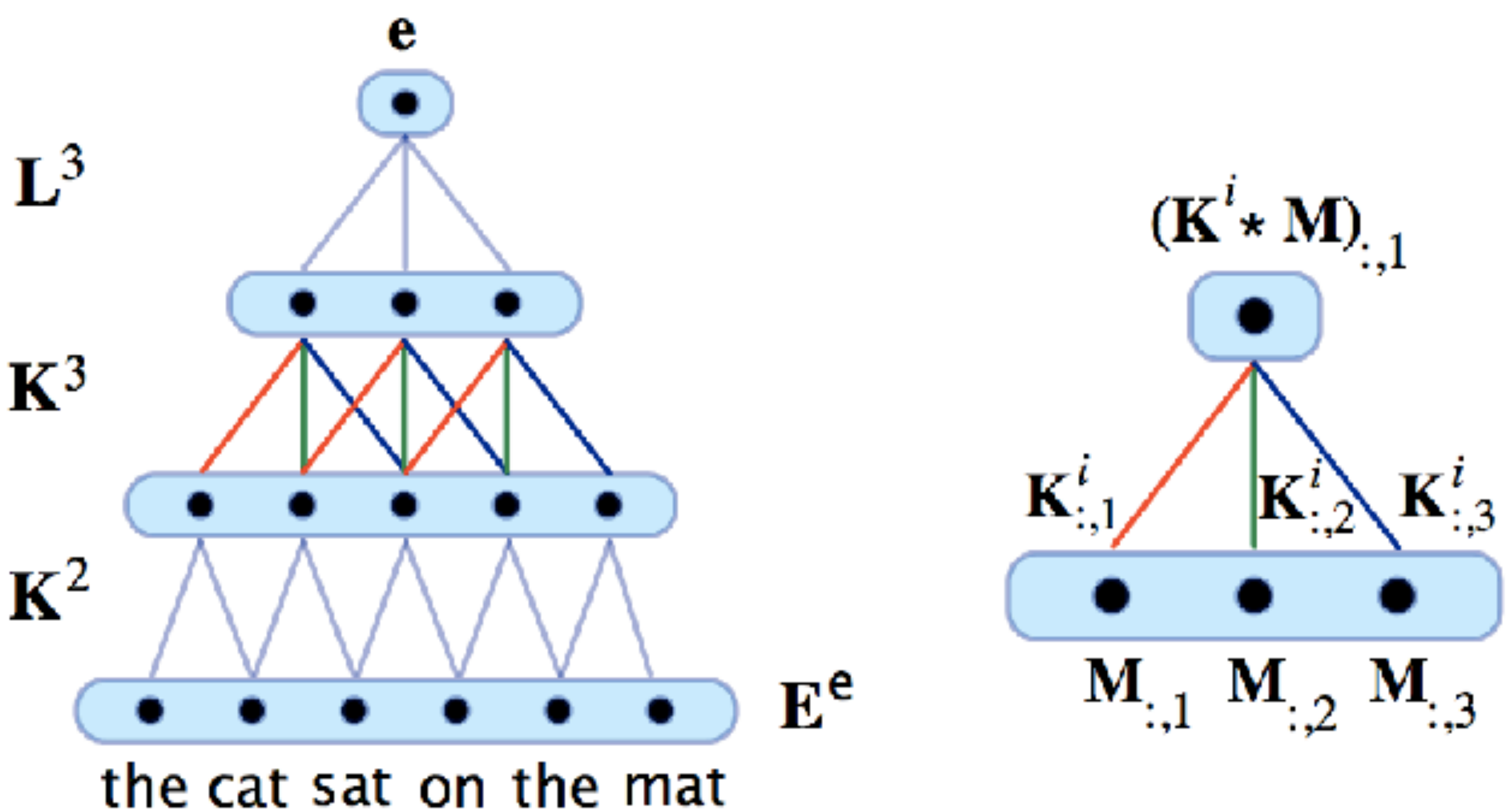


What do you think of this model?

K&B 2013: CSM Encoder

How should we define $\mathbf{c} = \text{embed}(\boldsymbol{x})$?

Convolutional sentence model (CSM)



K&B 2013: CSM Encoder

- **Good**
  - Convolutions learn interactions among features in a local context
  - By stacking them, longer range dependencies can be learnt
  - Deep ConvNets have a branching structure similar to trees, but no parser is required
- **Bad**
  - Sentences have different lengths, need different depth trees; convnets are not usually so dynamic, but see*

* Kalchbrenner et al. (2014). A convolutional neural network for modelling sentences. In *Proc. ACL*.

K&B 2013: RNN Decoder

Encoder

$$\mathbf{c} = \text{embed}(\boldsymbol{x})$$

$$\mathbf{s} = \mathbf{V}\mathbf{c}$$

Recurrent decoder

Recurrent connection

Embedding of w_{t-1}

Source sentence

$$\mathbf{h}_t = g(\mathbf{W}[\mathbf{h}_{t-1}; \mathbf{w}_{t-1}] + \mathbf{s} + \mathbf{b}])$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}'$$

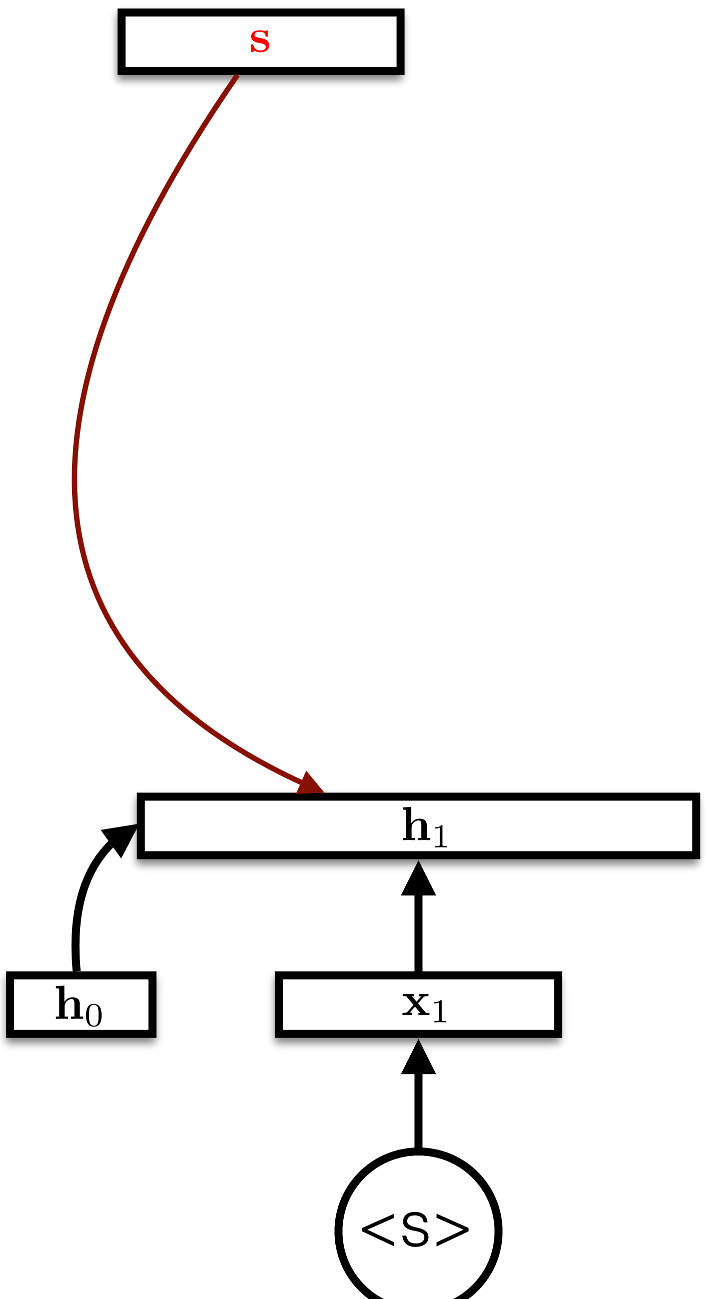
Learnt bias

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}) = \text{softmax}(\mathbf{u}_t)$$

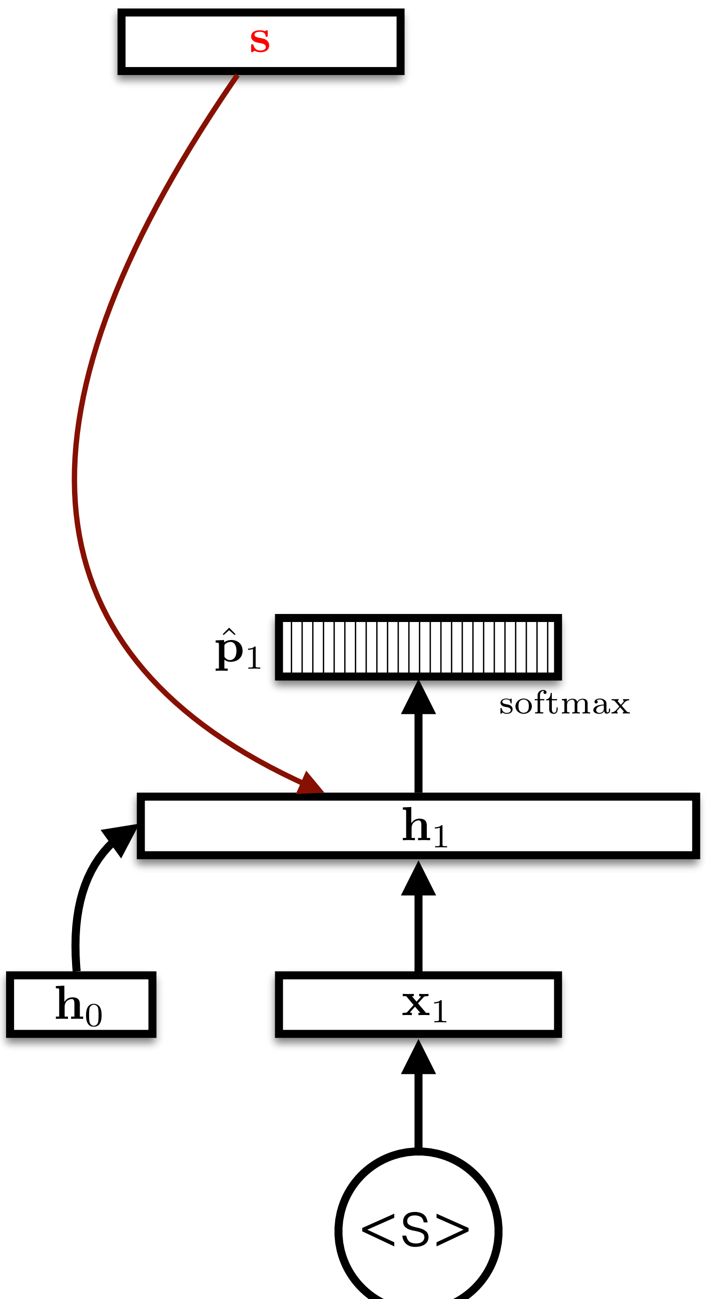
Recall unconditional RNN

$$\mathbf{h}_t = g(\mathbf{W}[\mathbf{h}_{t-1}; \mathbf{w}_{t-1}] + \mathbf{b}])$$

K&B 2013: RNN Decoder

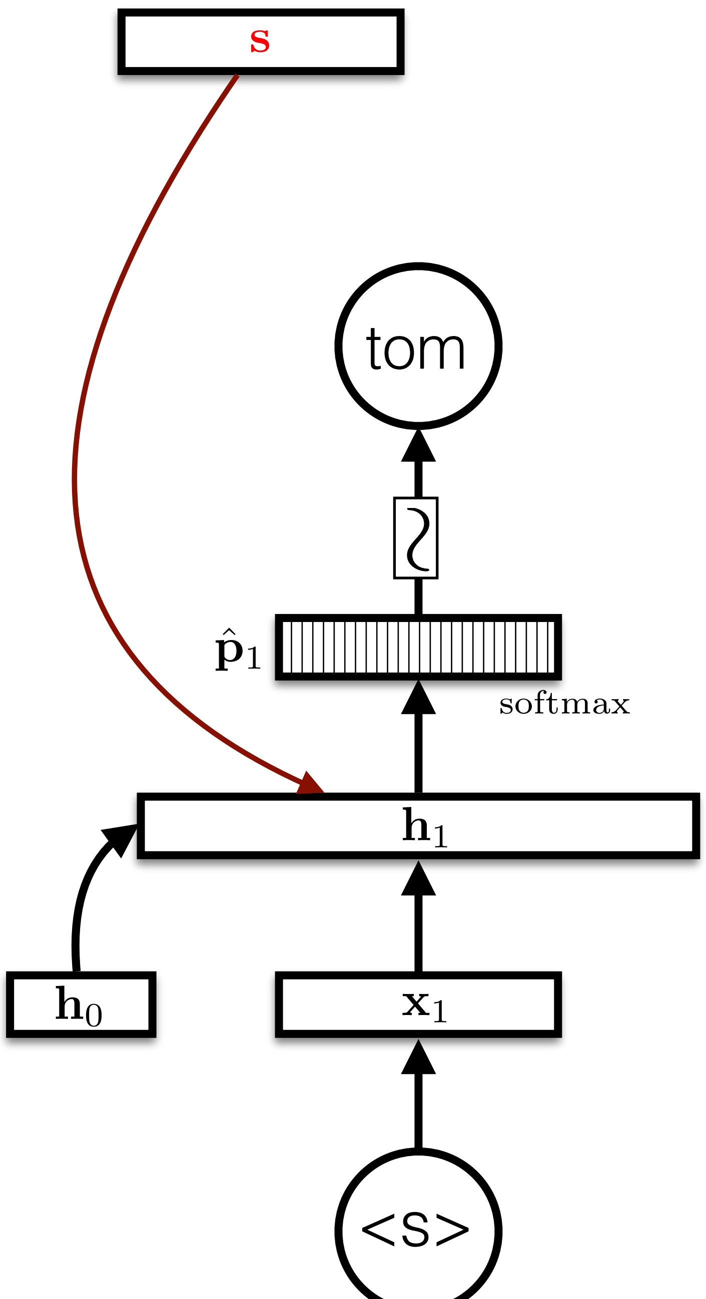


K&B 2013: RNN Decoder



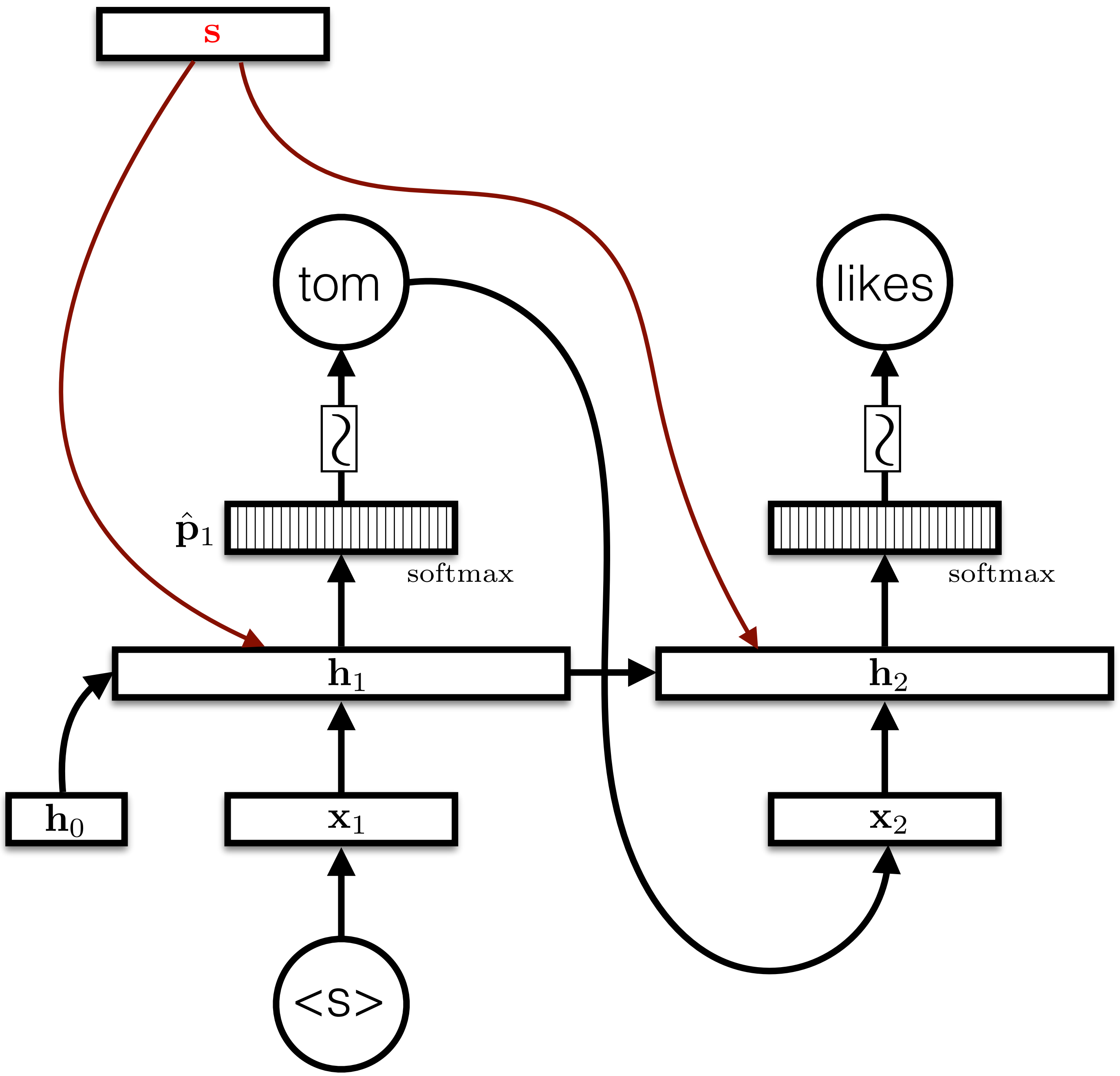
K&B 2013: RNN Decoder

$$p(tom \mid \mathbf{s}, \langle \mathbf{s} \rangle)$$



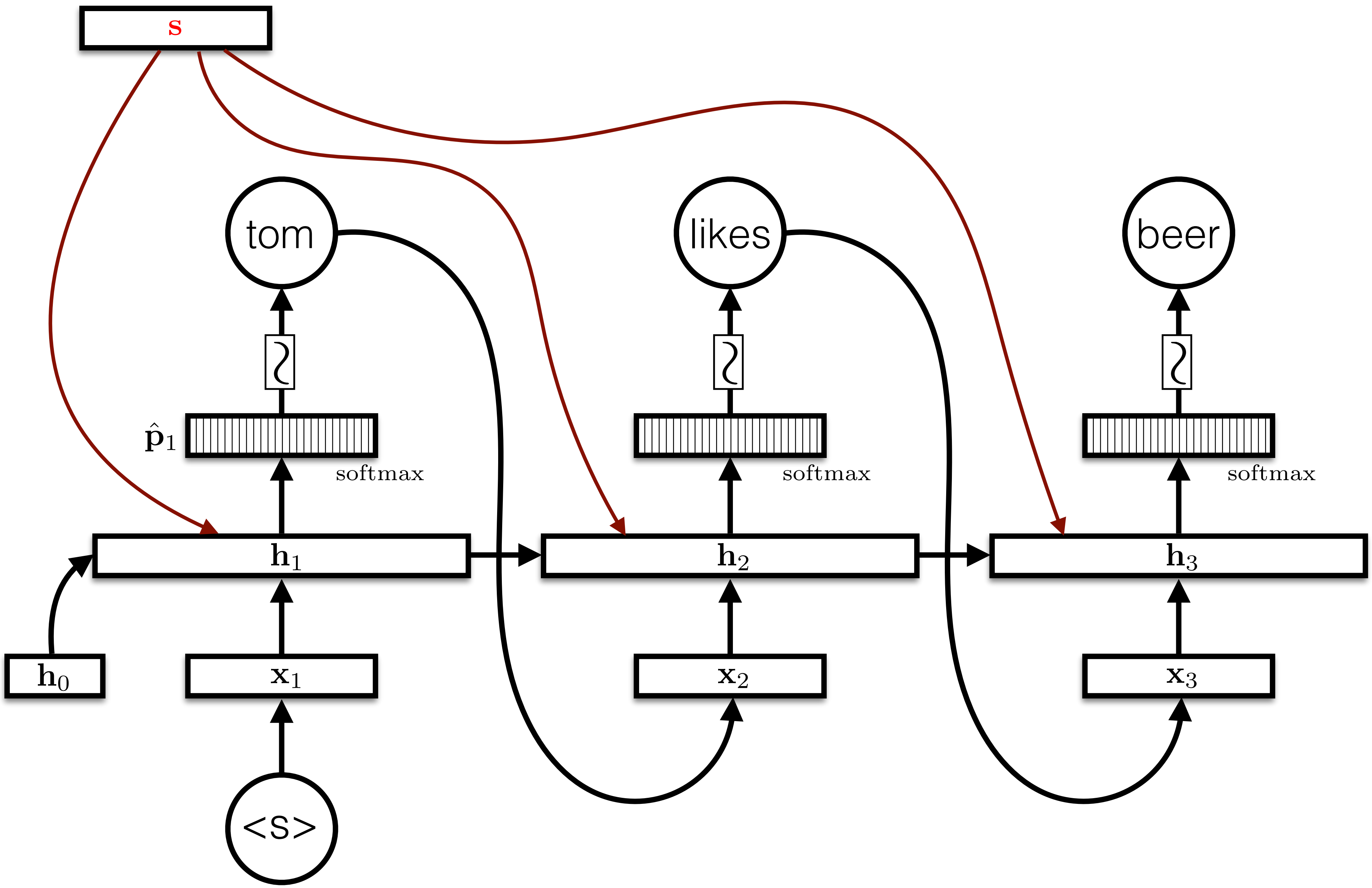
K&B 2013: RNN Decoder

$$p(tom \mid \mathbf{s}, \langle \mathbf{s} \rangle) \times p(likes \mid \mathbf{s}, \langle \mathbf{s} \rangle, tom)$$



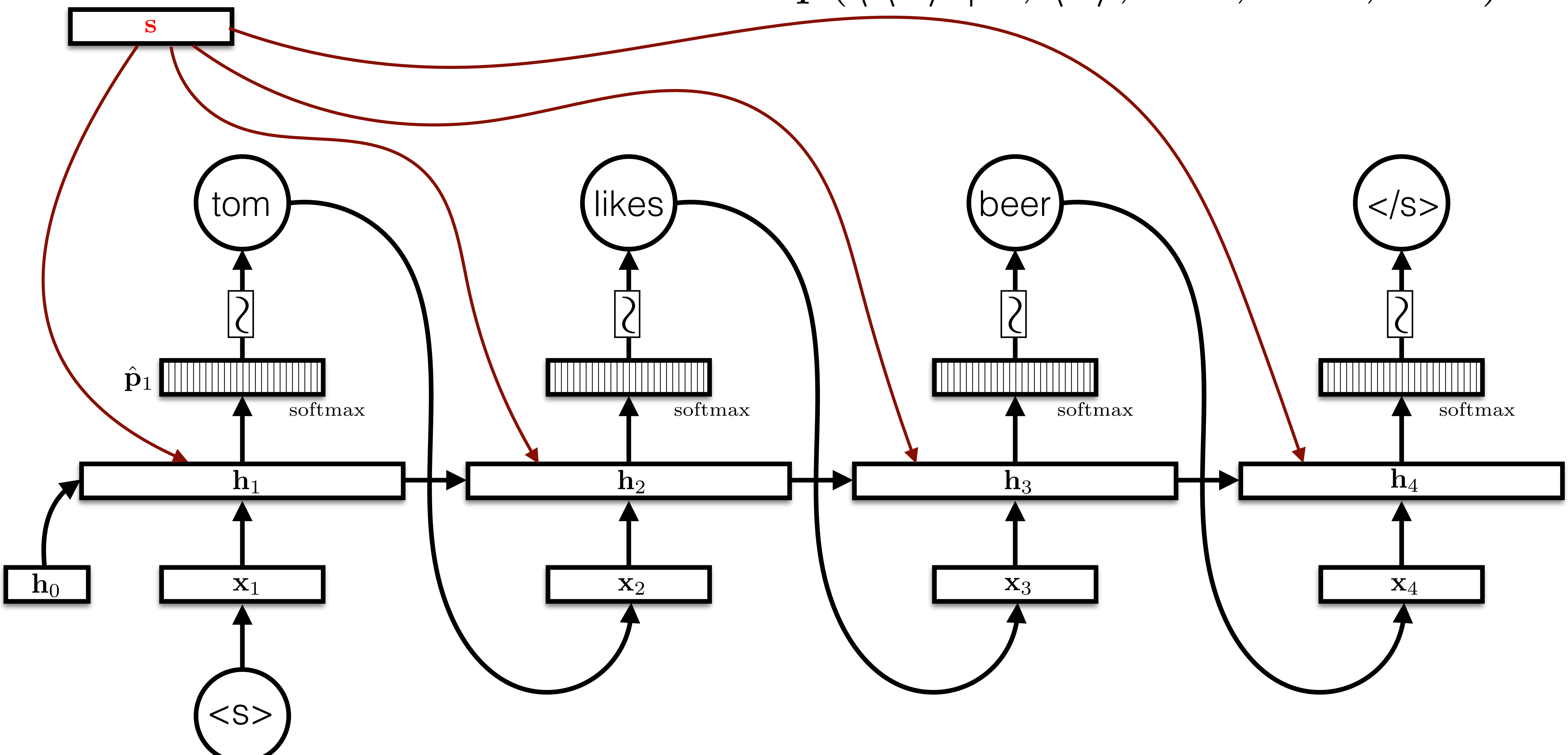
K&B 2013: RNN Decoder

$$p(tom \mid \mathbf{s}, \langle \mathbf{s} \rangle) \times p(likes \mid \mathbf{s}, \langle \mathbf{s} \rangle, tom)$$
$$\times p(beer \mid \mathbf{s}, \langle \mathbf{s} \rangle, tom, likes)$$



K&B 2013: RNN Decoder

$$p(tom \mid \mathbf{s}, \langle \mathrm{s} \rangle) \times p(likes \mid \mathbf{s}, \langle \mathrm{s} \rangle, tom) \times p(beer \mid \mathbf{s}, \langle \mathrm{s} \rangle, tom, likes) \times p(\langle \backslash \mathrm{s} \rangle \mid \mathbf{s}, \langle \mathrm{s} \rangle, tom, likes, beer)$$



Sutskever et al. (2014)

LSTM encoder

$$(\mathbf{c}_0, \mathbf{h}_0) \text{ are parameters}$$

$$(\mathbf{c}_i, \mathbf{h}_i) = \mathrm{LSTM}(x_i, \mathbf{c}_{i-1}, \mathbf{h}_{i-1})$$

The encoding is $(\mathbf{c}_\ell, \mathbf{h}_\ell)$ where $\ell = |\boldsymbol{x}|$.

LSTM decoder

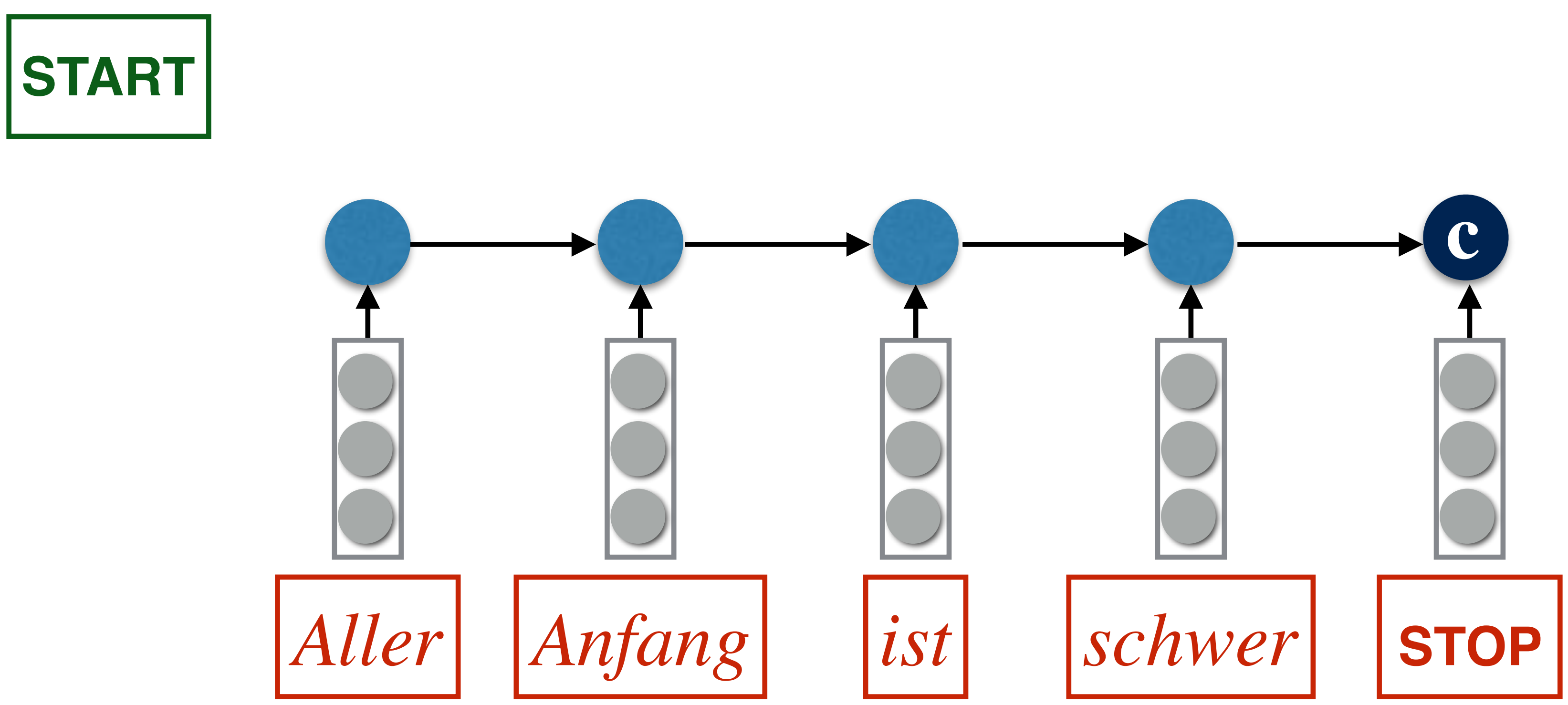
$$w_0 = \langle \mathbf{s} \rangle$$

$$(\mathbf{c}_{t+\ell}, \mathbf{h}_{t+\ell}) = \mathrm{LSTM}(w_{t-1}, \mathbf{c}_{t+\ell-1}, \mathbf{h}_{t+\ell-1})$$

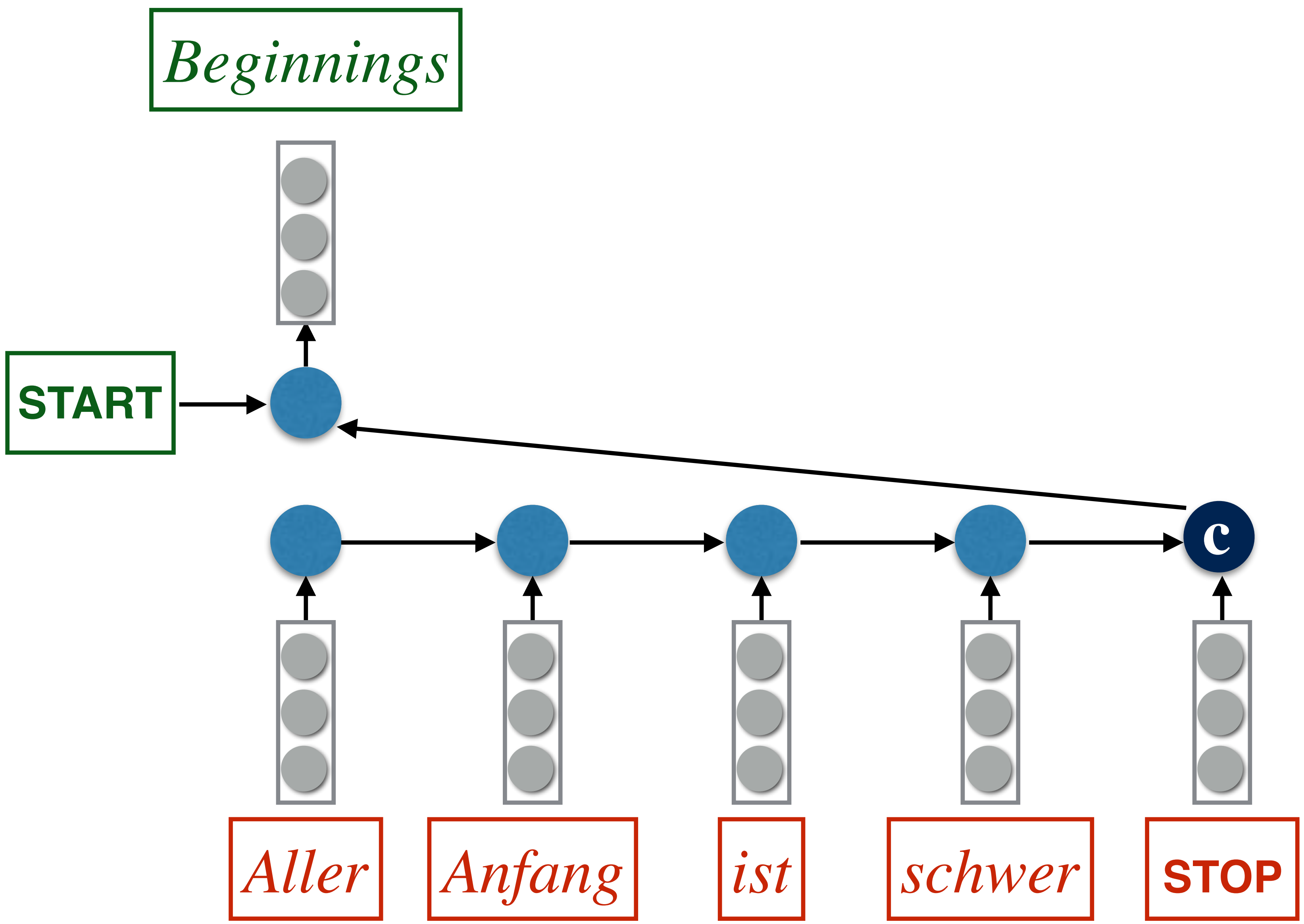
$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_{t+\ell} + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}) = \mathrm{softmax}(\mathbf{u}_t)$$

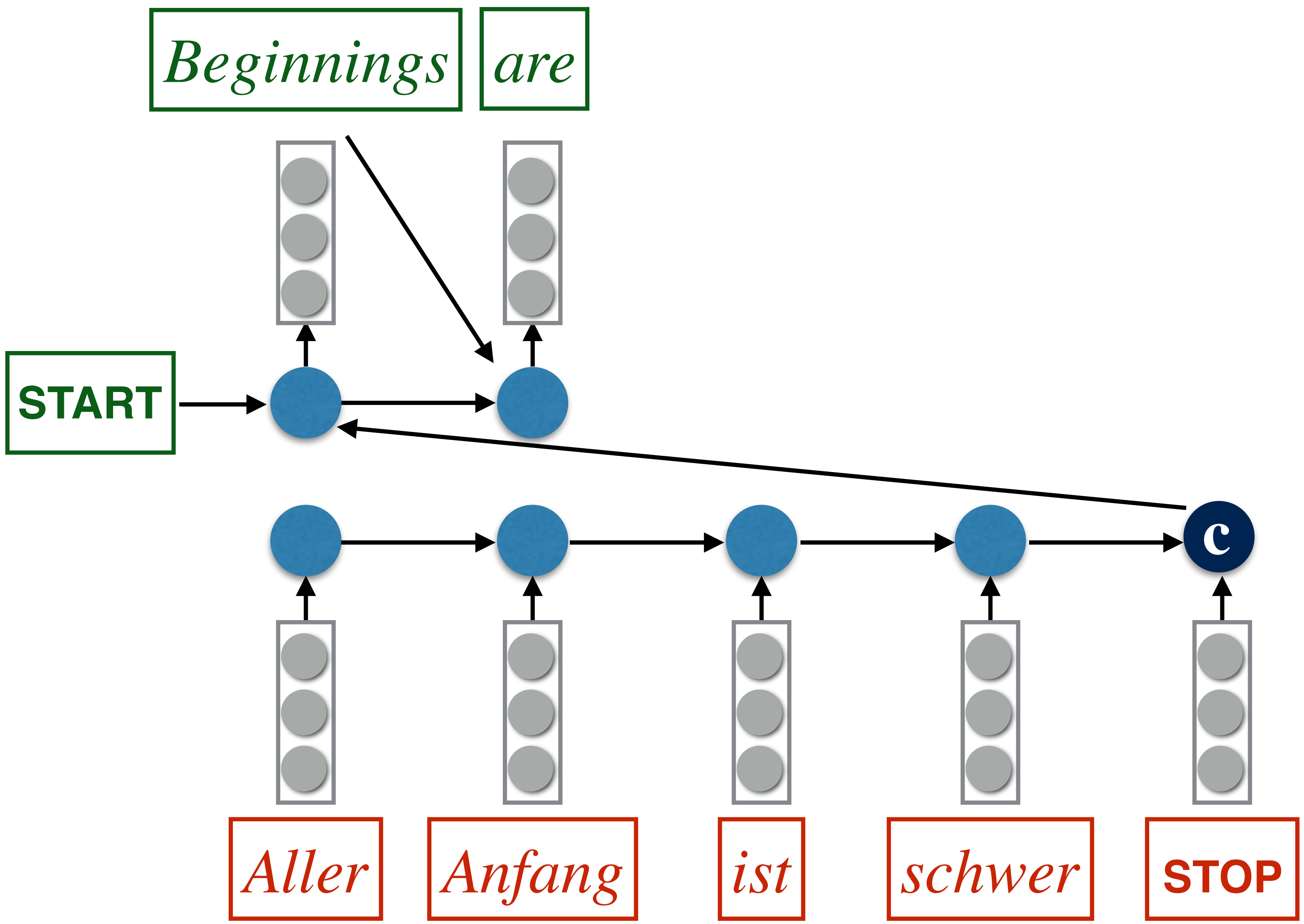
Sutskever et al. (2014)



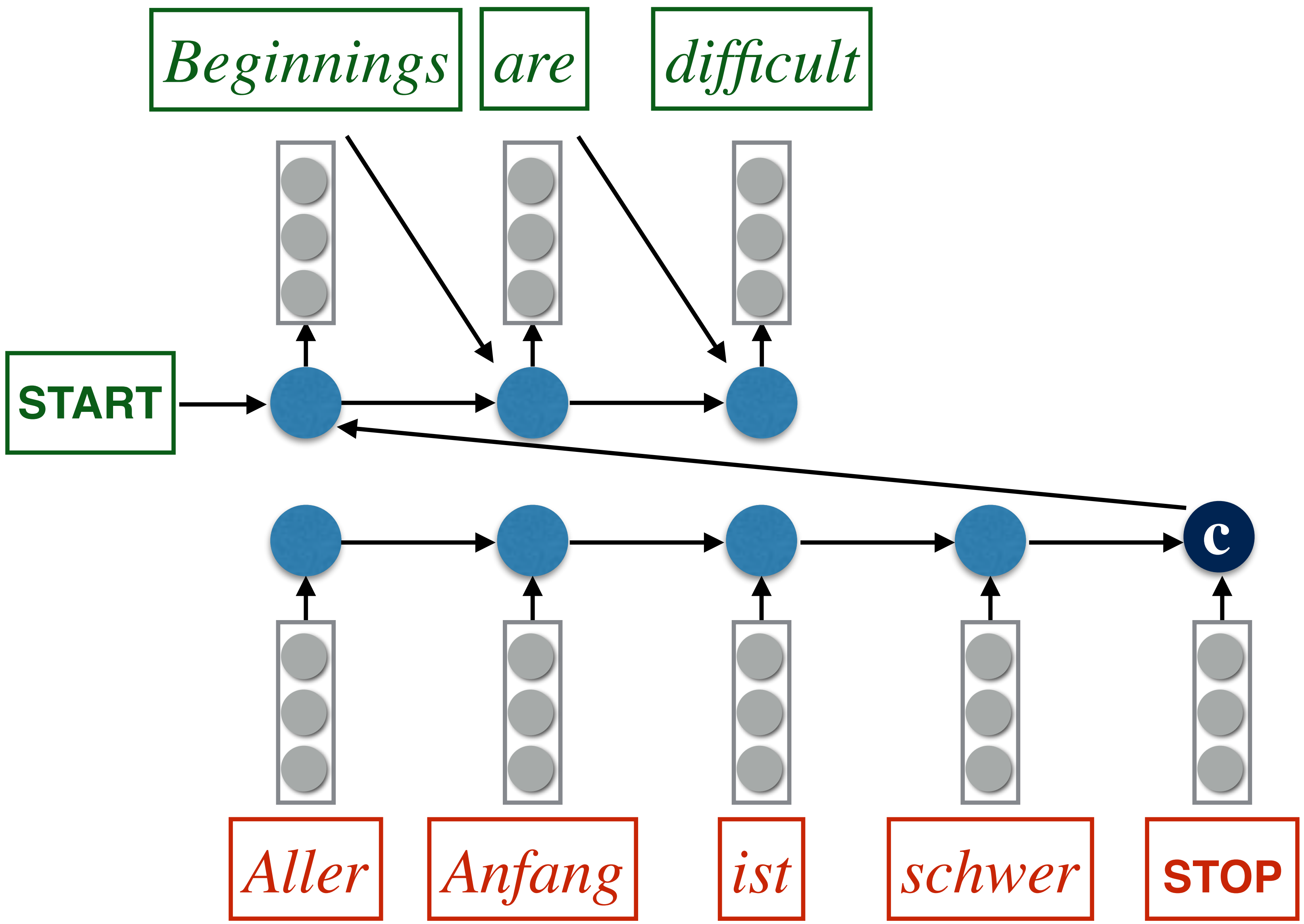
Sutskever et al. (2014)



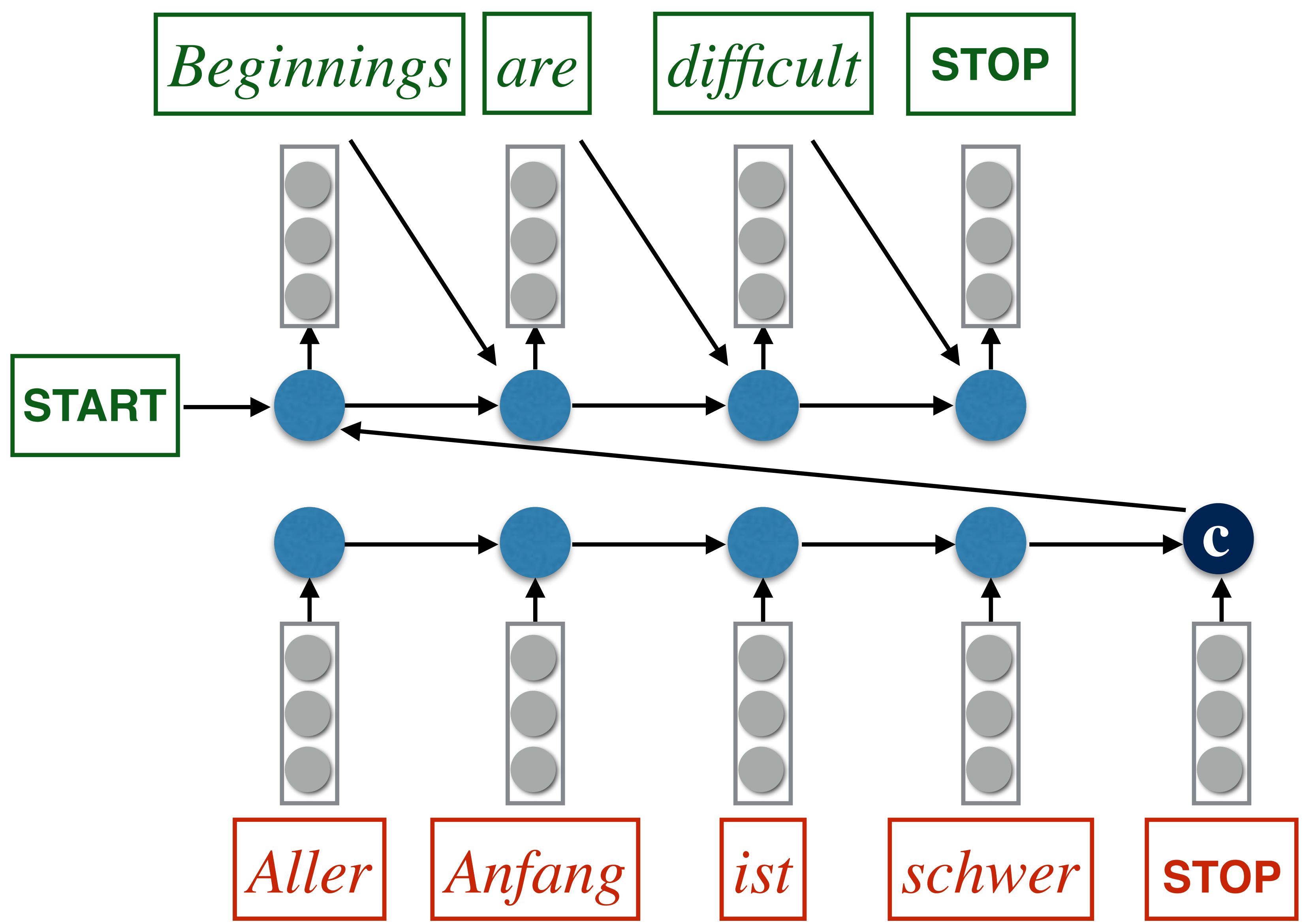
Sutskever et al. (2014)



Sutskever et al. (2014)



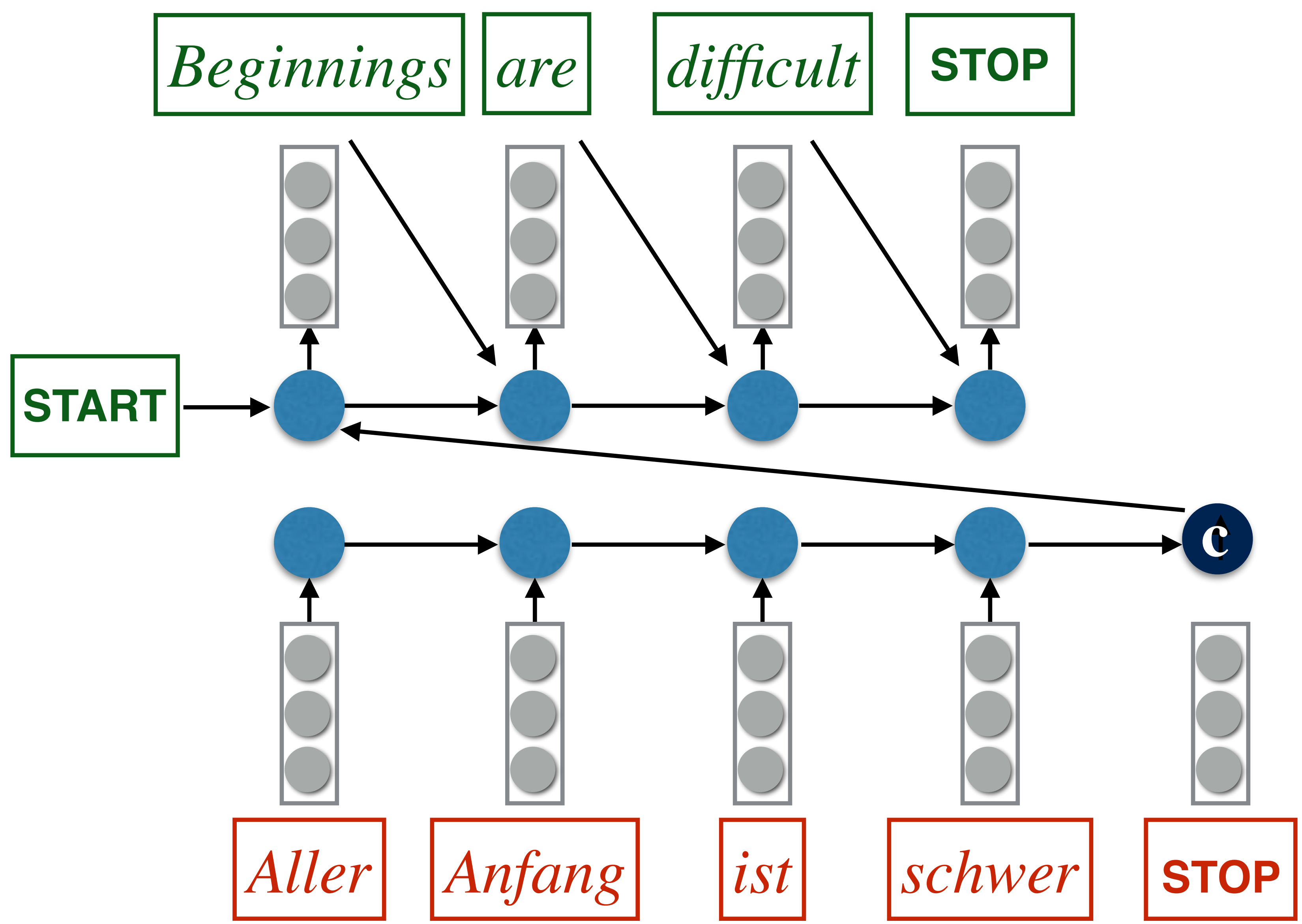
Sutskever et al. (2014)



Sutskever et al. (2014)

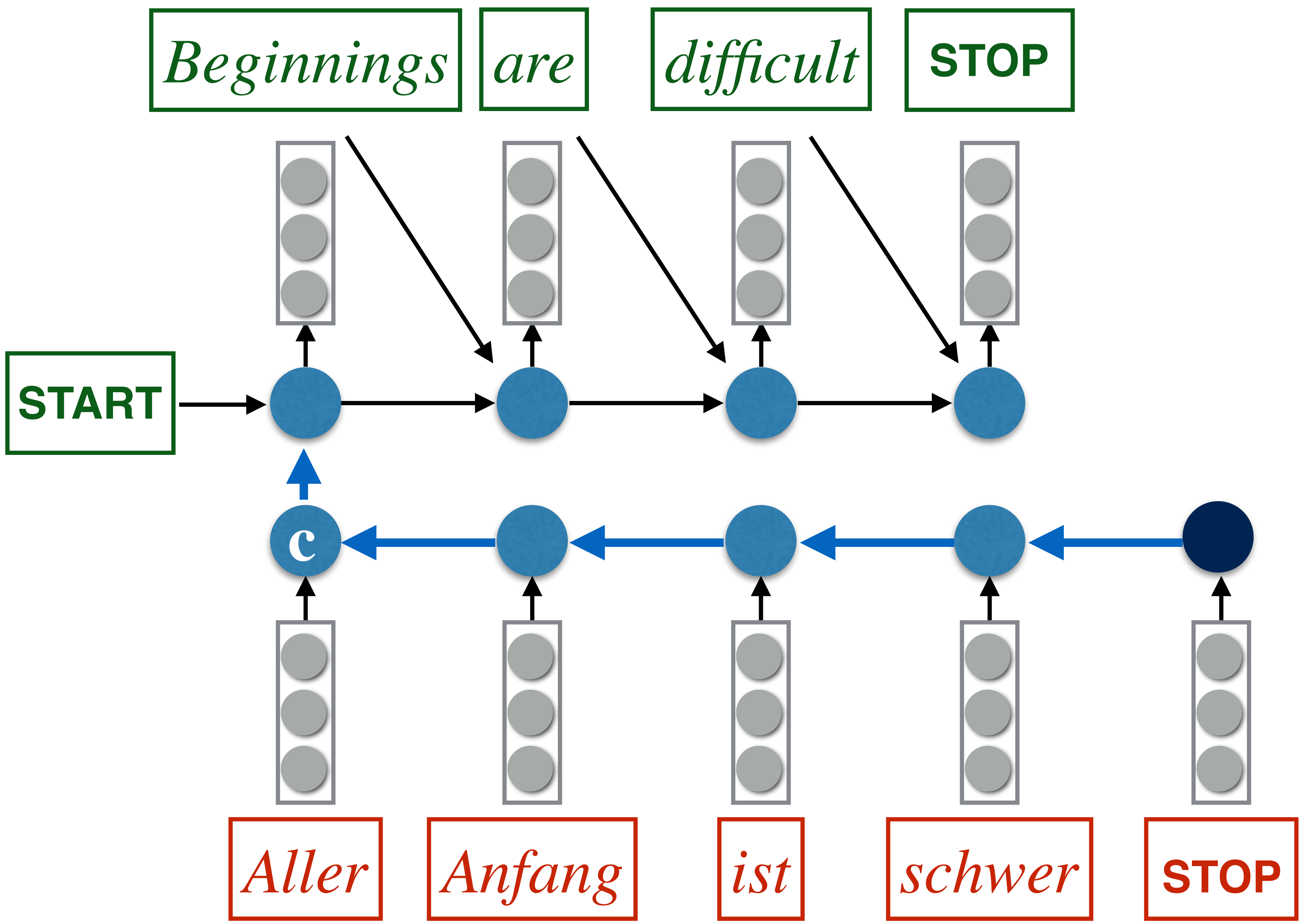
- **Good**
  - RNNs deal naturally with sequences of various lengths
  - LSTMs in principle can propagate gradients a long distance
  - Very simple architecture!
- **Bad**
  - The hidden state has to remember a lot of information! (We will return to this problem on Thursday.)

Sutskever et al. (2014): Tricks



Sutskever et al. (2014): Tricks

Read the input sequence "backwards": **+4 BLEU**



Sutskever et al. (2014): Tricks

Use an ensemble of J **independently trained** models.

Ensemble of 2 models: **+3 BLEU**

Ensemble of 5 models: **+4.5 BLEU**

Decoder:

$$(\mathbf{c}_{t+\ell}^{(j)}, \mathbf{h}_{t+\ell}^{(j)}) = \mathrm{LSTM}^{(j)}(w_{t-1}, \mathbf{c}_{t+\ell-1}^{(j)}, \mathbf{h}_{t+\ell-1}^{(j)})$$

$$\mathbf{u}_t^{(j)} = \mathbf{P}\mathbf{h}_t^{(j)} + \mathbf{b}^{(j)}$$

$$\mathbf{u}_t = \frac{1}{J}\sum_{j'=1}^{J} \mathbf{u}^{(j')}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}) = \mathrm{softmax}(\mathbf{u}_t)$$

A word about decoding

In general, we want to find the most probable (MAP) output given the input, i.e.

$$\begin{aligned} \boldsymbol{w}^* &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{w}} \sum_{t=1}^{|\boldsymbol{w}|} \log p(w_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}) \end{aligned}$$

A word about decoding

In general, we want to find the most probable (MAP) output given the input, i.e.

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x})$$
$$= \arg\max_{\boldsymbol{w}} \sum_{t=1}^{|\boldsymbol{w}|} \log p(w_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t})$$

This is, for general RNNs, a hard problem. We therefore approximate it with a **greedy search**:

$$w_1^* = \arg\max_{w_1} p(w_1 \mid \boldsymbol{x})$$
$$w_2^* = \arg\max_{w_2} p(w_2 \mid \boldsymbol{x}, w_1^*)$$
$$\vdots$$
$$w_t^* = \arg\max_{w_2} p(w_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}^*)$$

A word about decoding

In general, we want to find the most probable (MAP) output given the input, i.e.

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} p(\boldsymbol{w} \mid \boldsymbol{x})$$
$$= \arg\max_{\boldsymbol{w}} \sum_{t=1}^{|\boldsymbol{w}|} \log p(w_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t})$$

This is, for general RNNs, a ~~hard~~ **undecidable :(** problem. We therefore approximate it with a **greedy search**:

$$w_1^* = \arg\max_{w_1} p(w_1 \mid \boldsymbol{x})$$
$$w_2^* = \arg\max_{w_2} p(w_2 \mid \boldsymbol{x}, w_1^*)$$
$$\vdots$$
$$w_t^* = \arg\max_{w_2} p(w_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}^*)$$

A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$$\boldsymbol{x} = Bier\ trinke\ ich$$

beer **drink** **I**

$\langle \mathtt{s} \rangle$ logprob=0

w_0 w_1 w_2 w_3

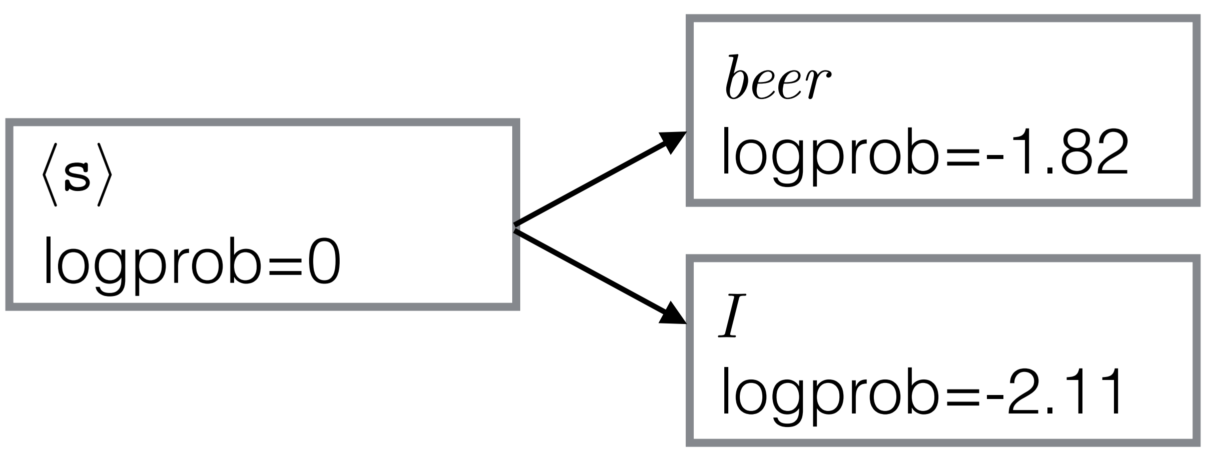
A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$\boldsymbol{x} = Bier\ trinke\ ich$

beer **drink** **I**



w_0 w_1 w_2 w_3

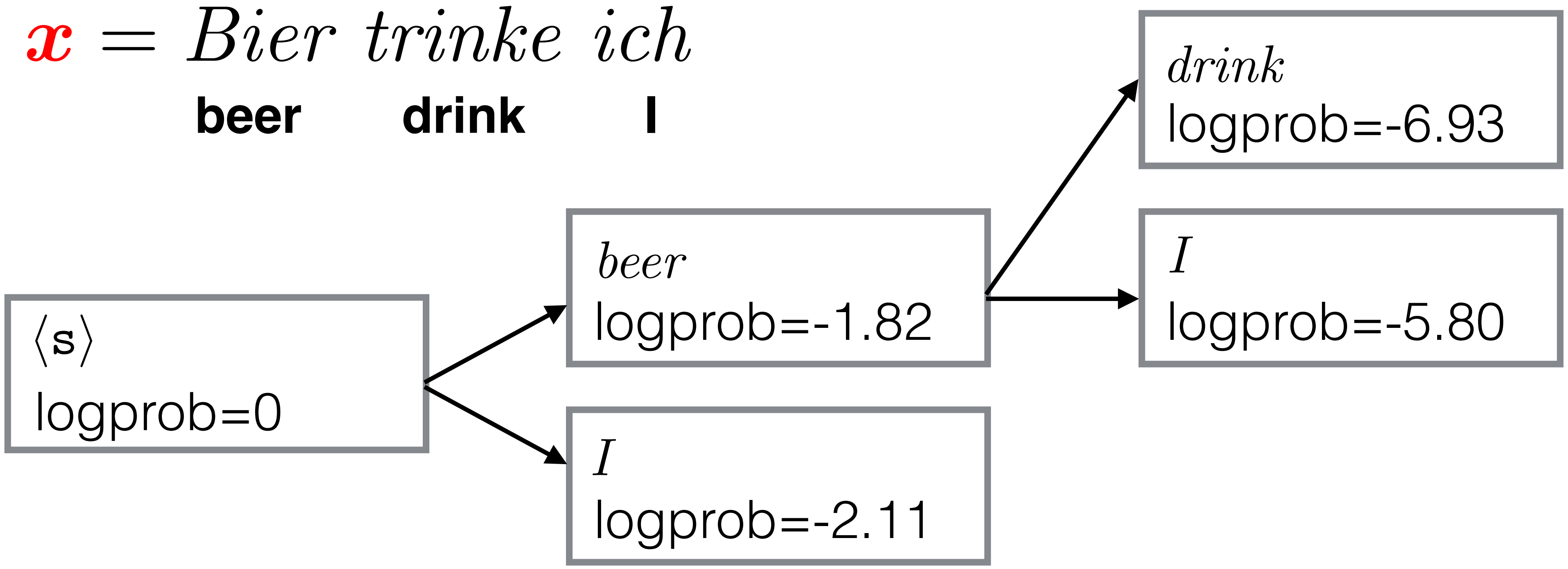
A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$\boldsymbol{x} = Bier\ trinke\ ich$

beer **drink** **I**



w_0 w_1 w_2 w_3

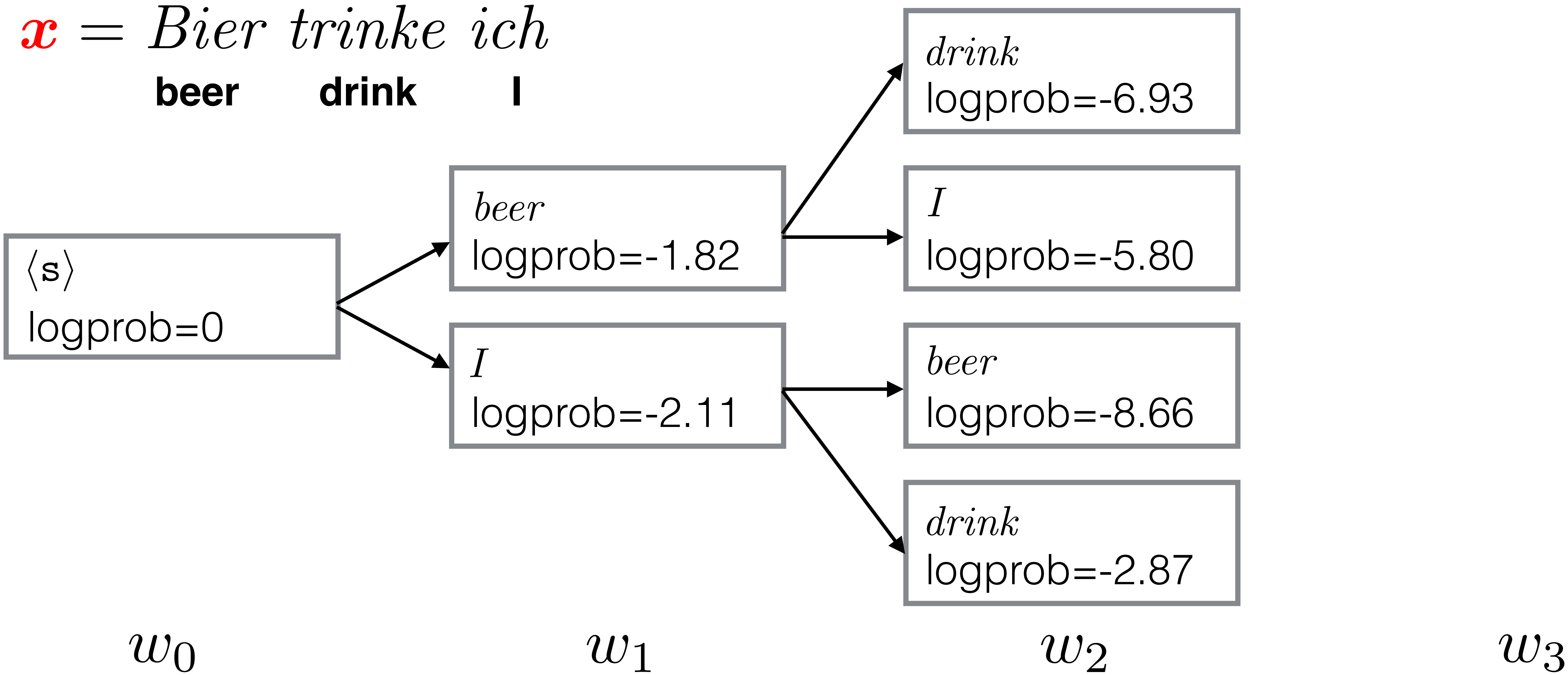
A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$\boldsymbol{x} = \textit{Bier trinke ich}$

beer **drink** **I**



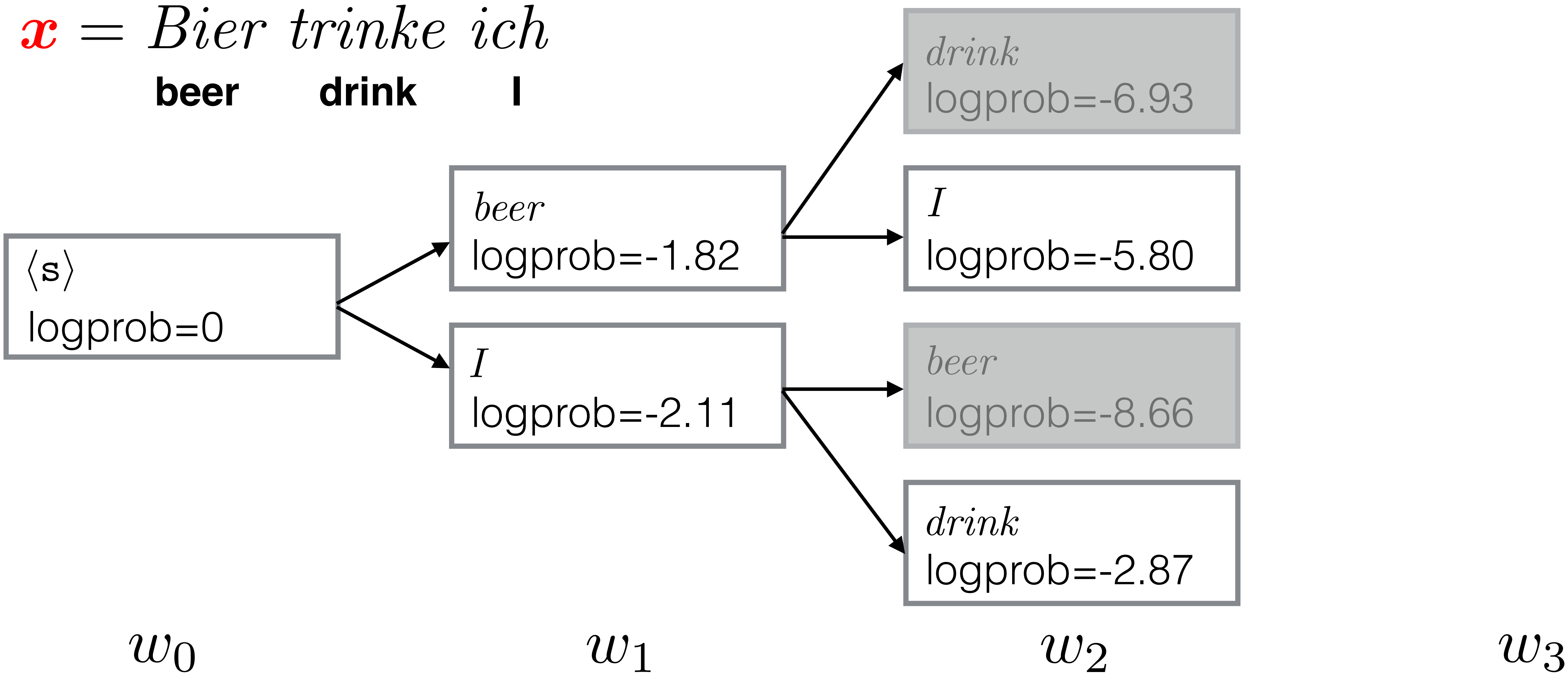
A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$\boldsymbol{x} = \textit{Bier trinke ich}$

beer **drink** **I**



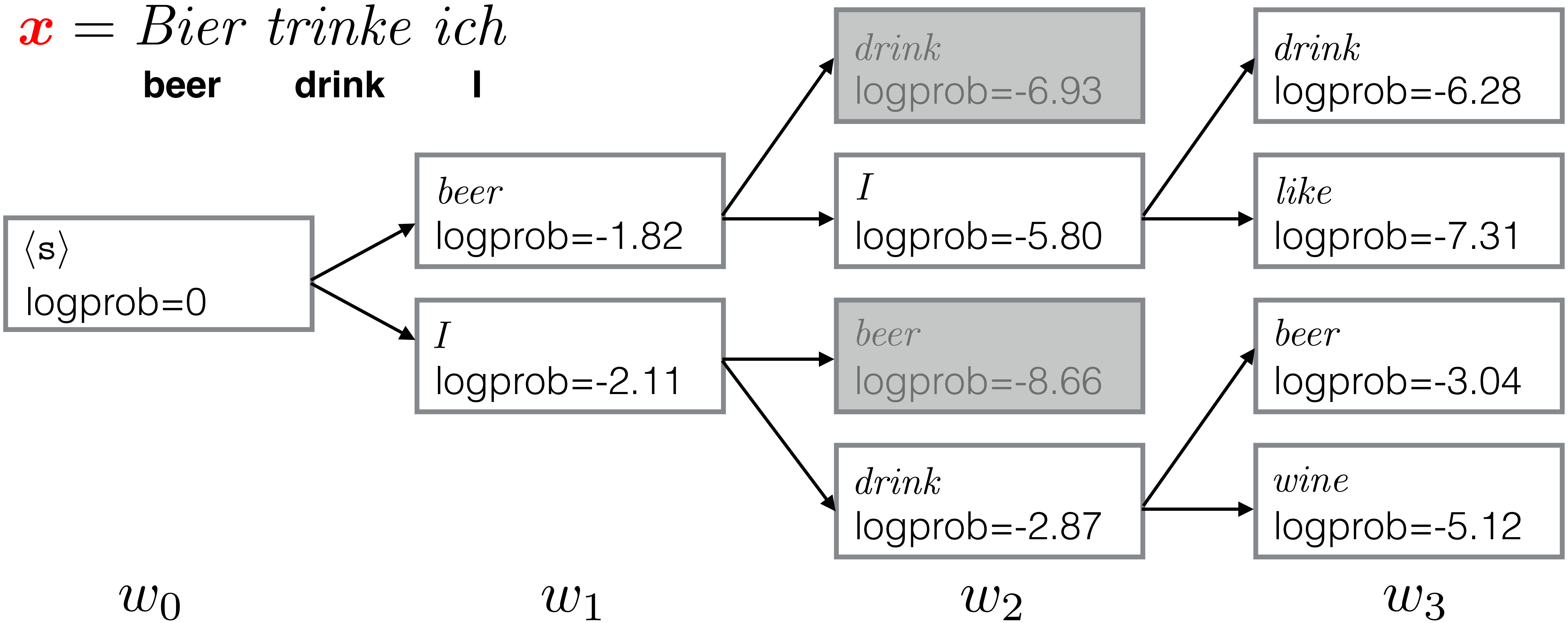
A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$\boldsymbol{x} = \textit{Bier trinke ich}$

beer **drink** **I**



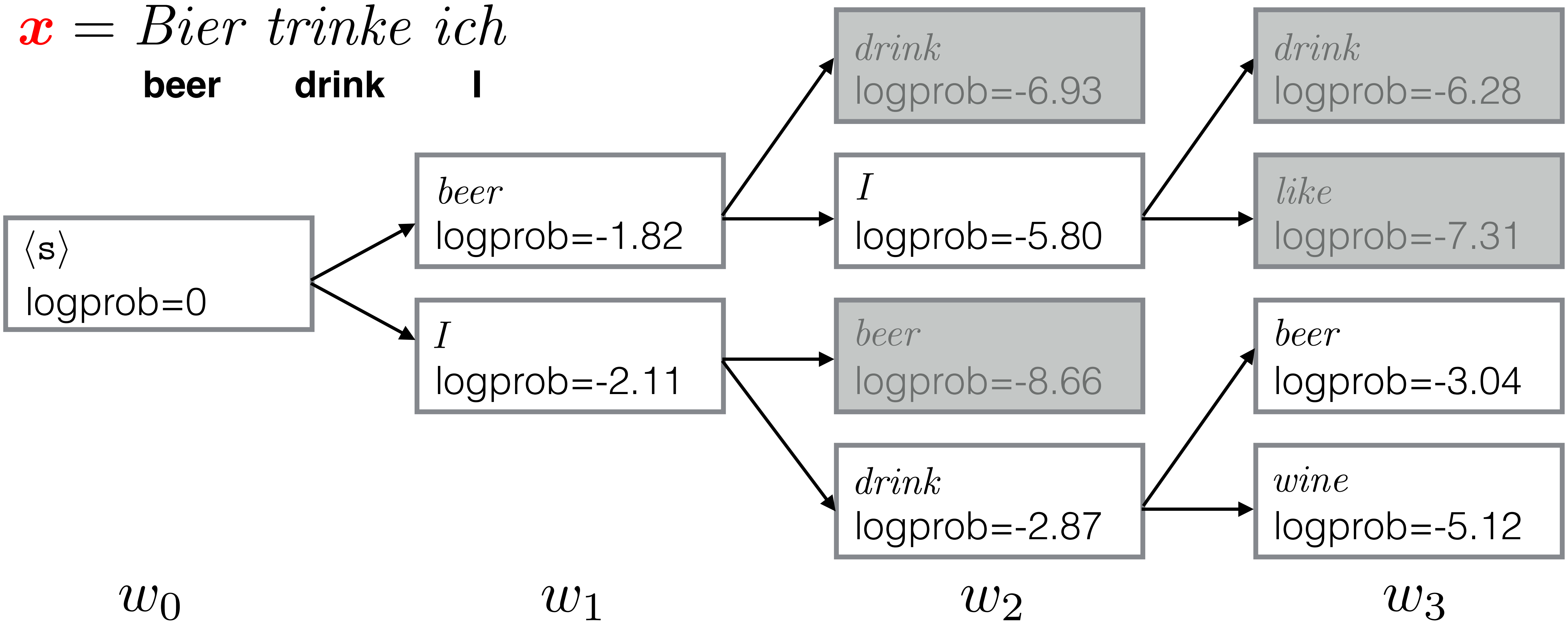
A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:

$\boldsymbol{x} = \textit{Bier trinke ich}$

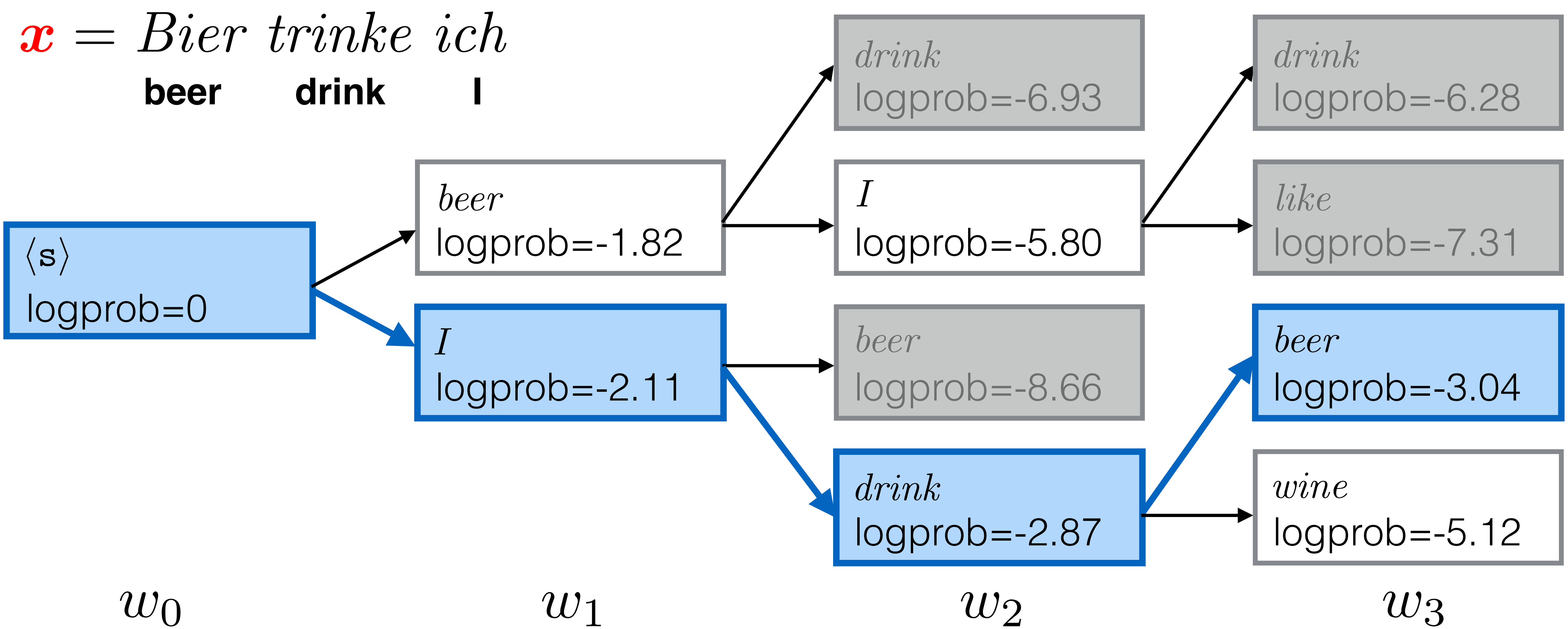
beer **drink** **I**



A word about decoding

A slightly better approximation is to use a **beam search** with beam size *b*. Key idea: keep track of top b hypothesis.

E.g., for *b*=2:



Sutskever et al. (2014): Tricks

Use beam search: **+1 BLEU**

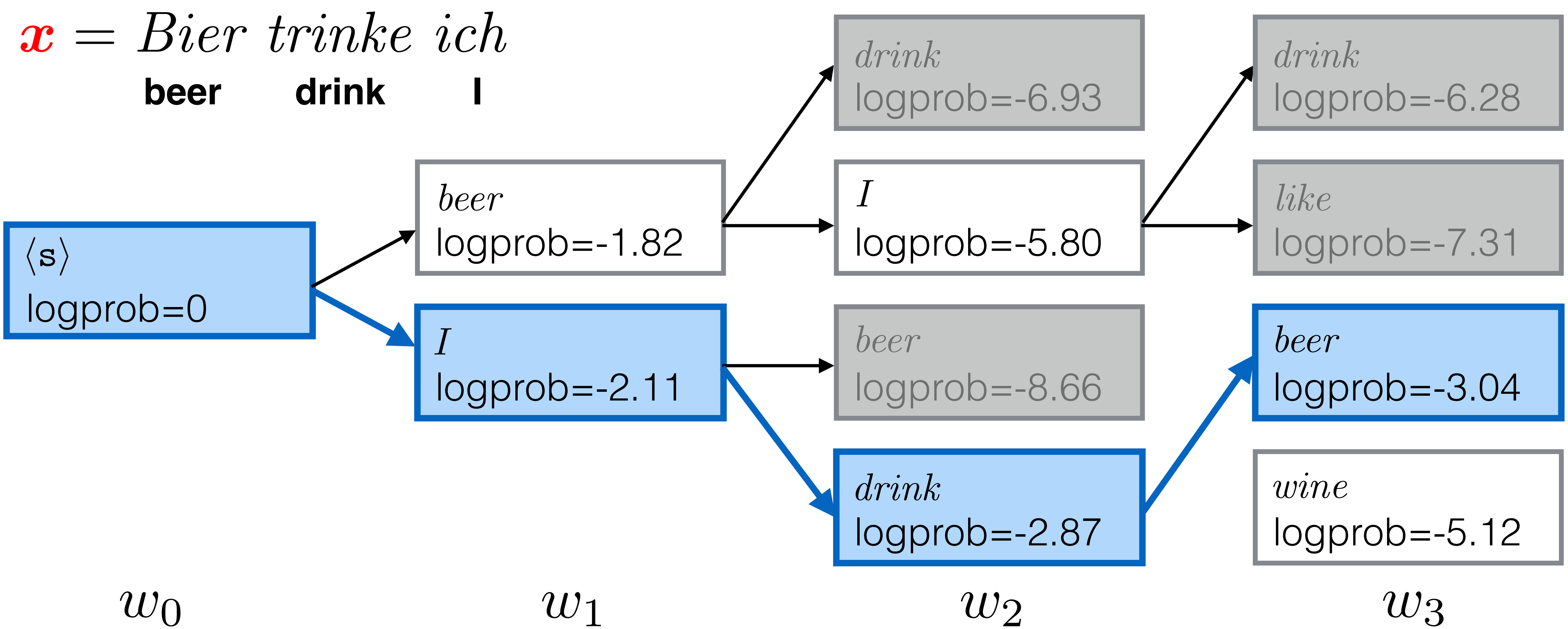


Image caption generation

- Neural networks are great for working with multiple modalities—**everything is a vector**!

- Image caption generation can therefore use the same techniques as translation modeling

- A word about data

  - Relatively few captioned images are available

  - Pre-train image embedding model using another task, like image identification (e.g., ImageNet)

Kiros et al. (2013)

- Looks a lot like Kalchbrenner and Blunsom (2013)
  - convolutional network on the input
  - n-gram language model on the output
- Innovation: **multiplicative interactions** in the decoder n-gram model

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Unconditional *n*-gram LM: *Embedding of* w_{t-1}

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}]$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{C}\mathbf{x}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{C}\mathbf{x}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$w_i = r_{i,w}$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$w_i = r_{i,w}$$

$$w_i = r_{i,j,w} x_j$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \mathrm{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \mathrm{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$w_i = r_{i,w}$$

how big is this tensor?

$$w_i = r_{i,j,w} x_j$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$w_i = r_{i,w}$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \mathrm{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{C}\mathbf{x}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \mathrm{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$\sout{w_i = r_{i,w}}$$

$$w_i = r_{i,j,w} x_j$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$\sout{w_i = r_{i,w}}$$

$$w_i = r_{i,j,w} x_j$$

what's the intuition here?

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$\sout{w_i = r_{i,w}}$$

$$w_i = r_{i,j,w} x_j$$

how big is this tensor?

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{C}\mathbf{x}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

~~$w_i = r_{i,w}$~~

$$w_i = r_{i,j,w} x_j$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \mathrm{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \mathrm{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$w_i = r_{i,w}$$

$$w_i = r_{i,j,w} x_j$$

$$w_i = u_{w,i} v_{i,j} \qquad (\mathbf{U} \in \mathbb{R}^{|V| \times d}, \quad \mathbf{V} \in \mathbb{R}^{d \times k})$$

$$\mathbf{r}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

Kiros et al. (2013)

Encoder $\mathbf{x} = \text{embed}(\boldsymbol{x})$

Simple conditional *n*-gram LM:

$$\mathbf{h}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{t-n+1}^{t-1}) = \text{softmax}(\mathbf{u}_t)$$

Multiplicative *n*-gram LM:

$$w_i = u_{w,i} v_{i,j} \qquad (\mathbf{U} \in \mathbb{R}^{|V| \times d}, \quad \mathbf{V} \in \mathbb{R}^{d \times k})$$

$$\mathbf{r}_t = \mathbf{W}[\mathbf{w}_{t-n+1}; \mathbf{w}_{t-n+2}; \ldots; \mathbf{w}_{t-1}] + \mathbf{Cx}$$

$$\mathbf{h}_t = (\mathbf{W}^{fr}\mathbf{r}_t) \odot (\mathbf{W}^{fx}\mathbf{x})$$

$$\mathbf{u}_t = \mathbf{P}\mathbf{h}_t + \mathbf{b}$$

$$p(W_t \mid \boldsymbol{x}, \boldsymbol{w}_{<t}) = \text{softmax}(\mathbf{u}_t)$$

Kiros et al. (2013)

- Two take-home messages:
  - Feed-forward n-gram models can be used in place of RNNs in conditional models
  - Modeling interactions between input modalities holds a lot of promise
    - Although MLP-type models can approximate higher order tensors, multiplicative models appear to make learning interactions easier

Questions?