Conditional Language Modeling

Chris Dyer

DeepMind

Carnegie Mellon University

Unconditional LMs

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

It is convenient to decompose this probability using the **chain rule**, as follows:

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

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

Evaluating unconditional LMs

How good is our unconditional language model?

1. Held-out per-word **cross entropy** or **perplexity**

$$H = -\frac{1}{|\boldsymbol{w}|} \sum_{i=1}^{|\boldsymbol{w}|} \log_2 p(w_i \mid \boldsymbol{w}_{<i})$$

(units: bits per word)

$$ppl = b^{-\frac{1}{|\boldsymbol{w}|} \sum_{i=1}^{|\boldsymbol{w}|} \log_b p(w_i|\boldsymbol{w}_{<i})}$$

(units: uncertainty per word)

Same as training criterion. *How uncertain is the model at each time position, an average?*

2. Task-based evaluation
Use in a task-model that uses a language model in place of some other language model. Does it improve?

History-based LMs

A common strategy is to make a **Markov assumption**, which is a conditional independence assumption.

$$
\begin{aligned}
p(\boldsymbol{w}) = \; & p(w_1)\times \\
& p(w_2 \mid w_1)\times \\
& p(w_3 \mid w_1, w_2)\times \\
& p(w_4 \mid w_1, w_2, w_3)\times \\
& \ldots
\end{aligned}
$$

History-based LMs

A common strategy is to make a **Markov assumption**, which is a conditional independence assumption.

$$
\begin{aligned}
p(\boldsymbol{w}) = {} & p(w_1)\times \\
& p(w_2 \mid w_1)\times \\
& p(w_3 \mid \cancel{w_1}, w_2)\times \\
& p(w_4 \mid \cancel{w_1, w_2}, w_3)\times \\
& \ldots
\end{aligned}
$$

Markov: forget the distant past.

Is this valid for language? No…

Is it practical? Often!

History-based LMs

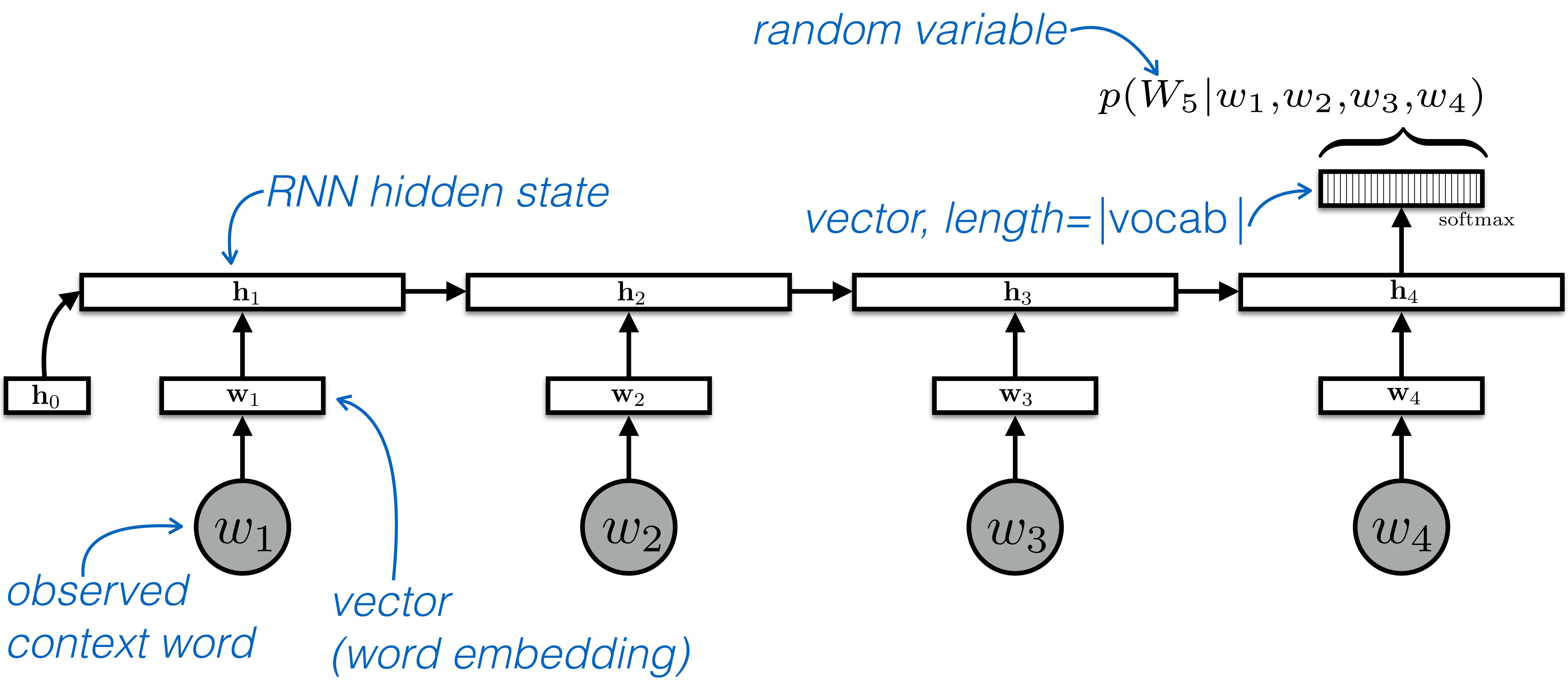
A common strategy is to make a **Markov assumption**, which is a conditional independence assumption.

$$
\begin{aligned}
p(\boldsymbol{w}) = p(w_1) &\times \\
p(w_2 \mid w_1) &\times \\
p(w_3 \mid \cancel{w_1}, w_2) &\times \\
p(w_4 \mid \cancel{w_1, w_2}, w_3) &\times \\
\cdots
\end{aligned}
$$

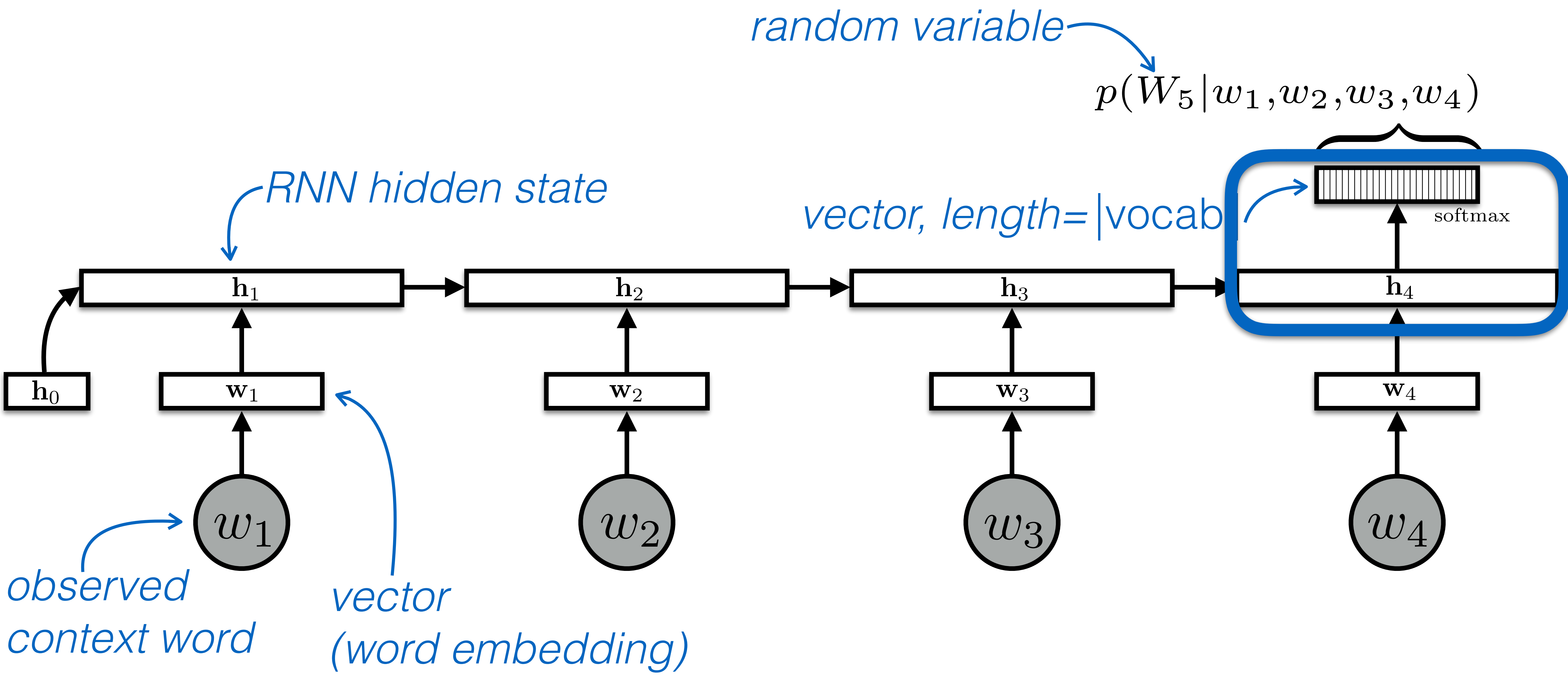
Markov: forget the distant past.
Is this valid for language? No…
Is it practical? Often!

Why RNNs are great for language: no more Markov assumptions!

History-based LMs with RNNs

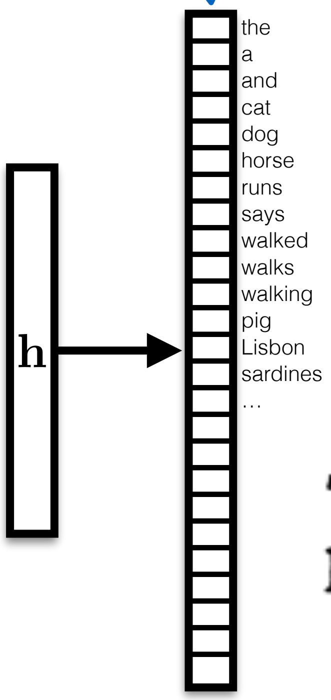


History-based LMs with RNNs



Distributions over words

Each dimension corresponds to a word in a closed vocabulary, V.



softmax

$$\mathbf{u} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

$$p_i = \frac{\exp u_i}{\sum_j \exp u_j}$$

The p_i's form a distribution, i.e.

$$p_i > 0 \quad \forall i, \quad \sum_i p_i = 1$$

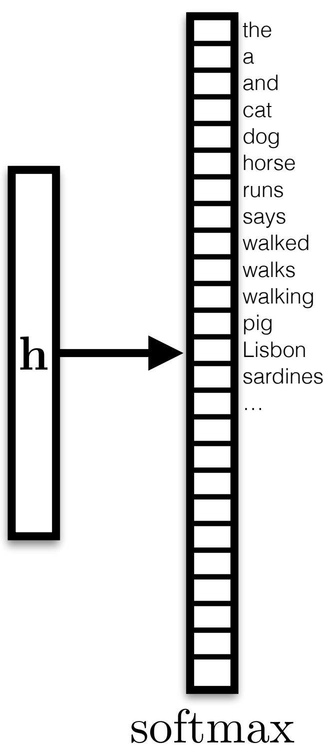
To enforce this stochastic constraint, we suggest a *normalised exponential* output non-linearity,

$$O_j = e^{I_j} \Big/ \sum_k e^{I_k}.$$

This "**softmax**" function is a generalisation of the logistic to multiple inputs. It also generalises maximum picking, or "Winner-Take-All", in the sense that that the outputs change smoothly, and equal inputs produce equal outputs. Although it looks rather cumbersome, and perhaps not really in the spirit of neural networks, those familiar with Markov random fields or statistical mechanics will know that it has convenient mathematical properties. Circuit designers will enjoy the simple transistor circuit which implements it.

Bridle. (1990) Probabilistic interpretation of feedforward classification…

Distributions over words

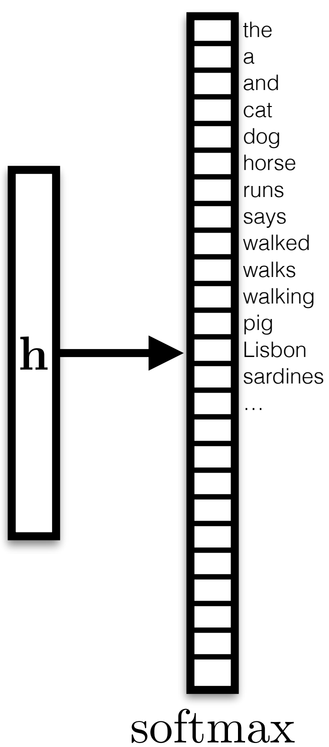


$$\mathbf{u} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

$$p_i = \frac{\exp u_i}{\sum_j \exp u_j}$$

$$\begin{aligned} p(\boldsymbol{w}) = {} & p(w_1) \times \\ & p(w_2 \mid w_1) \times \\ & p(w_3 \mid w_1, w_2) \times \\ & p(w_4 \mid w_1, w_2, w_3) \times \\ & \ldots \end{aligned}$$

Distributions over words



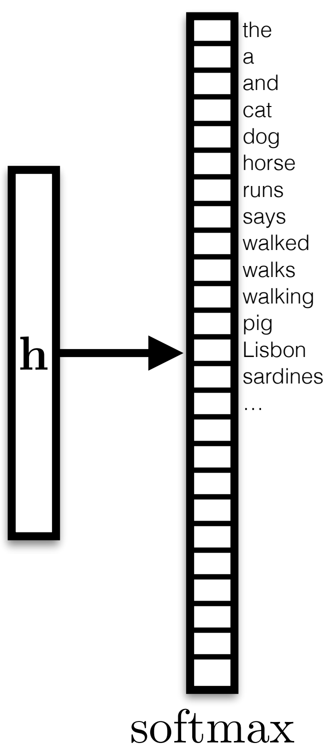
$$\mathbf{u} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

$$p_i = \frac{\exp u_i}{\sum_j \exp u_j}$$

$$\begin{aligned} p(\boldsymbol{w}) = p(w_1) \times \\ p(w_2 \mid w_1) \times \\ p(w_3 \mid w_1, w_2) \times \\ p(w_4 \mid w_1, w_2, w_3) \times \\ \cdots \end{aligned}$$

histories are sequences of words…

Distributions over words



$$\mathbf{u} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

$$p_i = \frac{\exp u_i}{\sum_j \exp u_j}$$

$\mathbf{h} \in \mathbb{R}^d$

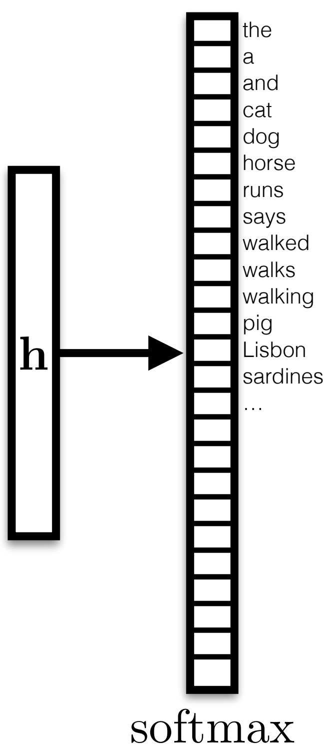
$|V| = 100,000$

What are the dimensions of $\mathbf{b}$?

$$\begin{aligned} p(\boldsymbol{w}) = p(w_1) \times \\ p(w_2 \mid w_1) \times \\ p(w_3 \mid w_1, w_2) \times \\ p(w_4 \mid w_1, w_2, w_3) \times \\ \cdots \end{aligned}$$

histories are sequences of words…

Distributions over words



$$\mathbf{u} = \mathbf{W}\mathbf{h} + \mathbf{b}$$

$$p_i = \frac{\exp u_i}{\sum_j \exp u_j}$$

$\mathbf{h} \in \mathbb{R}^d$

$|V| = 100,000$

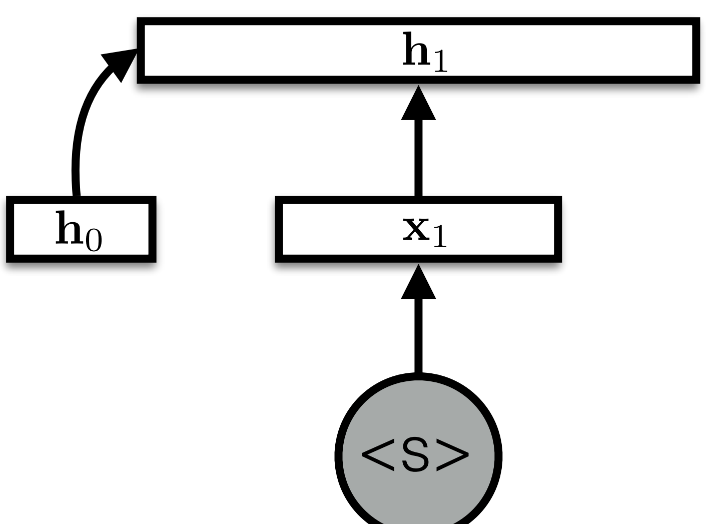
What are the dimensions of $\mathbf{W}$?

$$\begin{aligned} p(\boldsymbol{w}) = p(w_1) \times \\ p(w_2 \mid w_1) \times \\ p(w_3 \mid w_1, w_2) \times \\ p(w_4 \mid w_1, w_2, w_3) \times \\ \cdots \end{aligned}$$

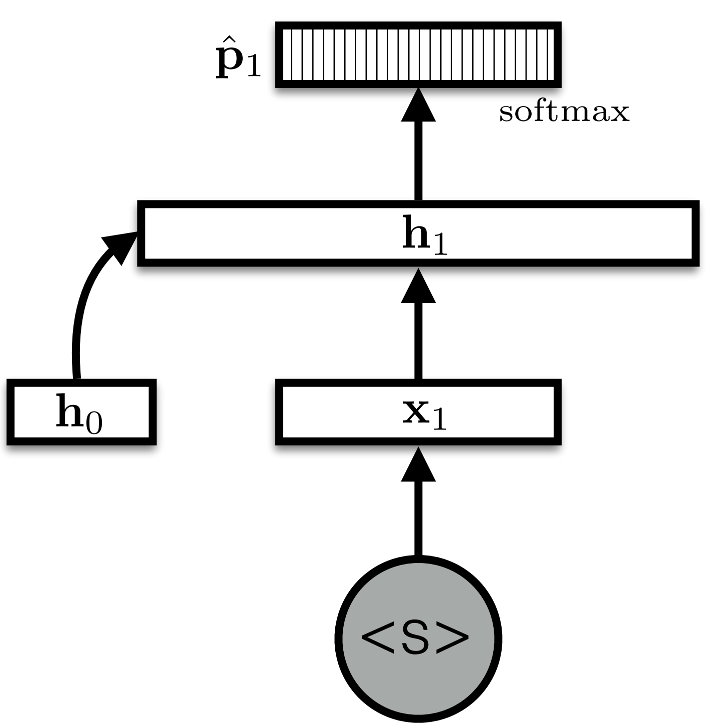
histories are sequences of words…

RNN language models

RNN language models

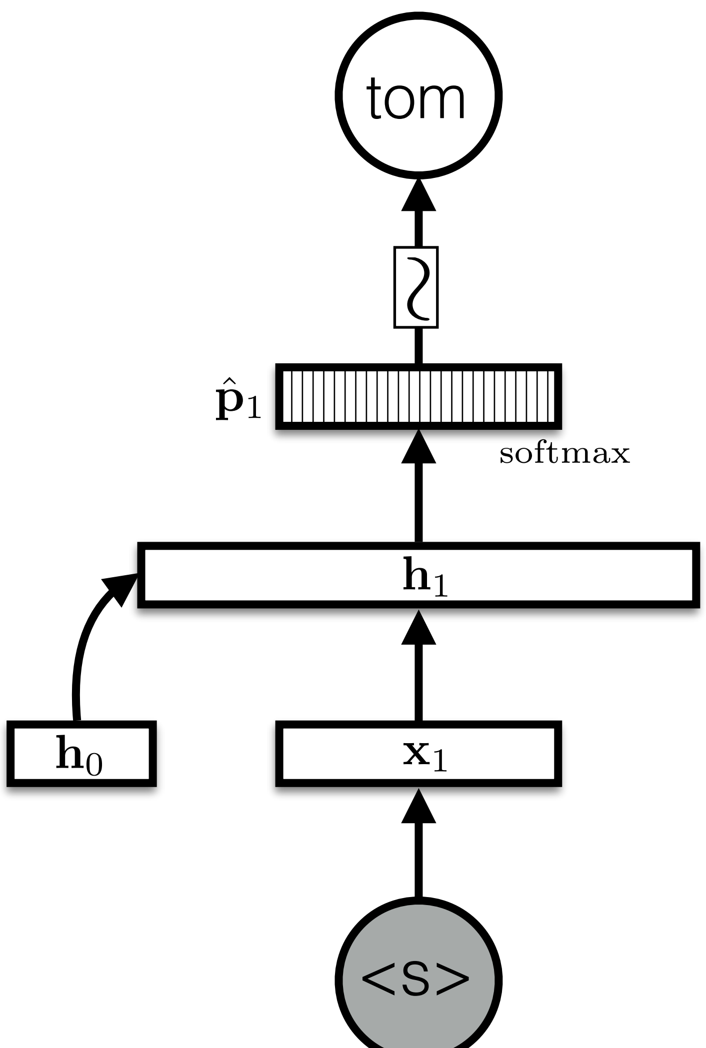


RNN language models



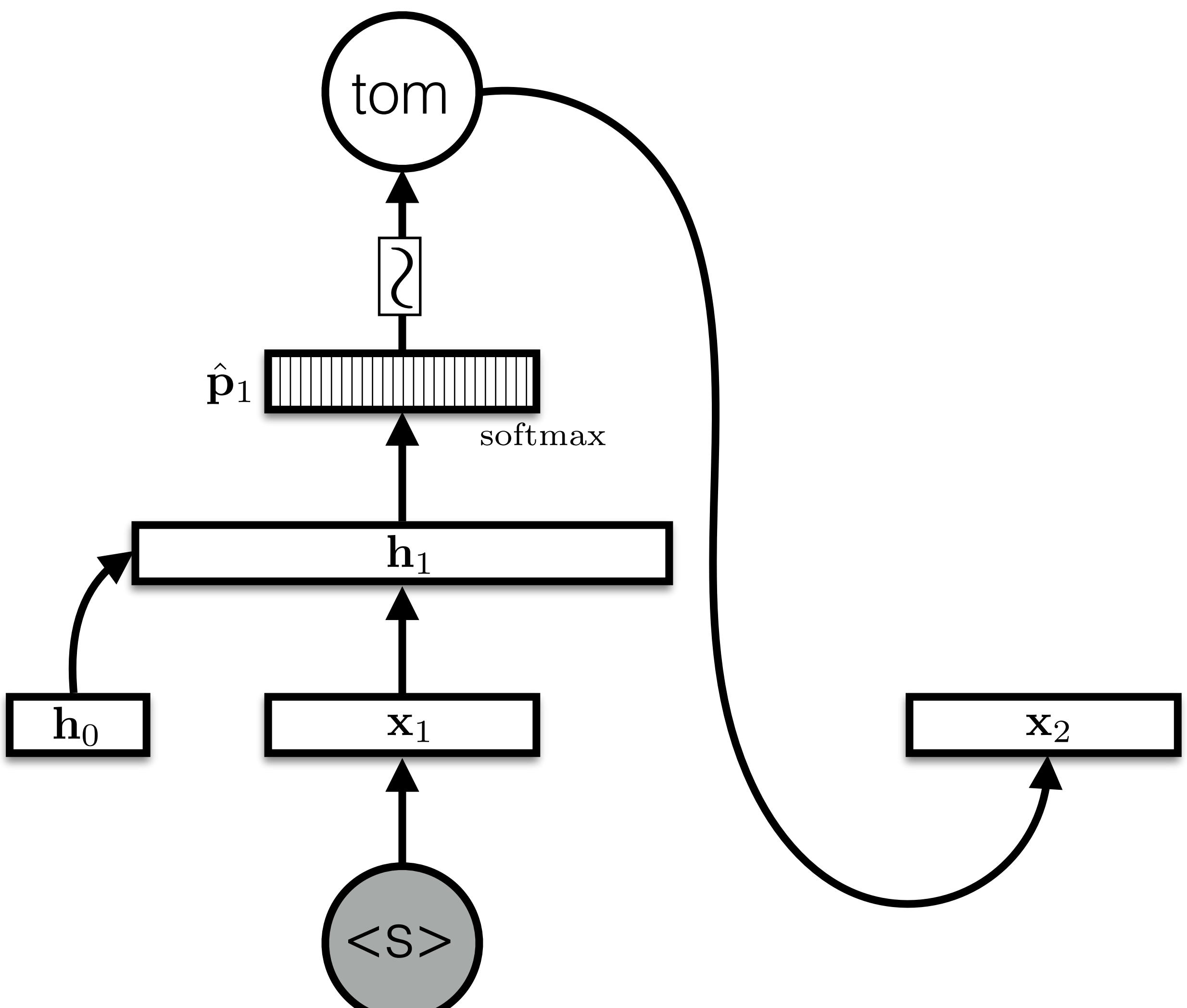
RNN language models

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



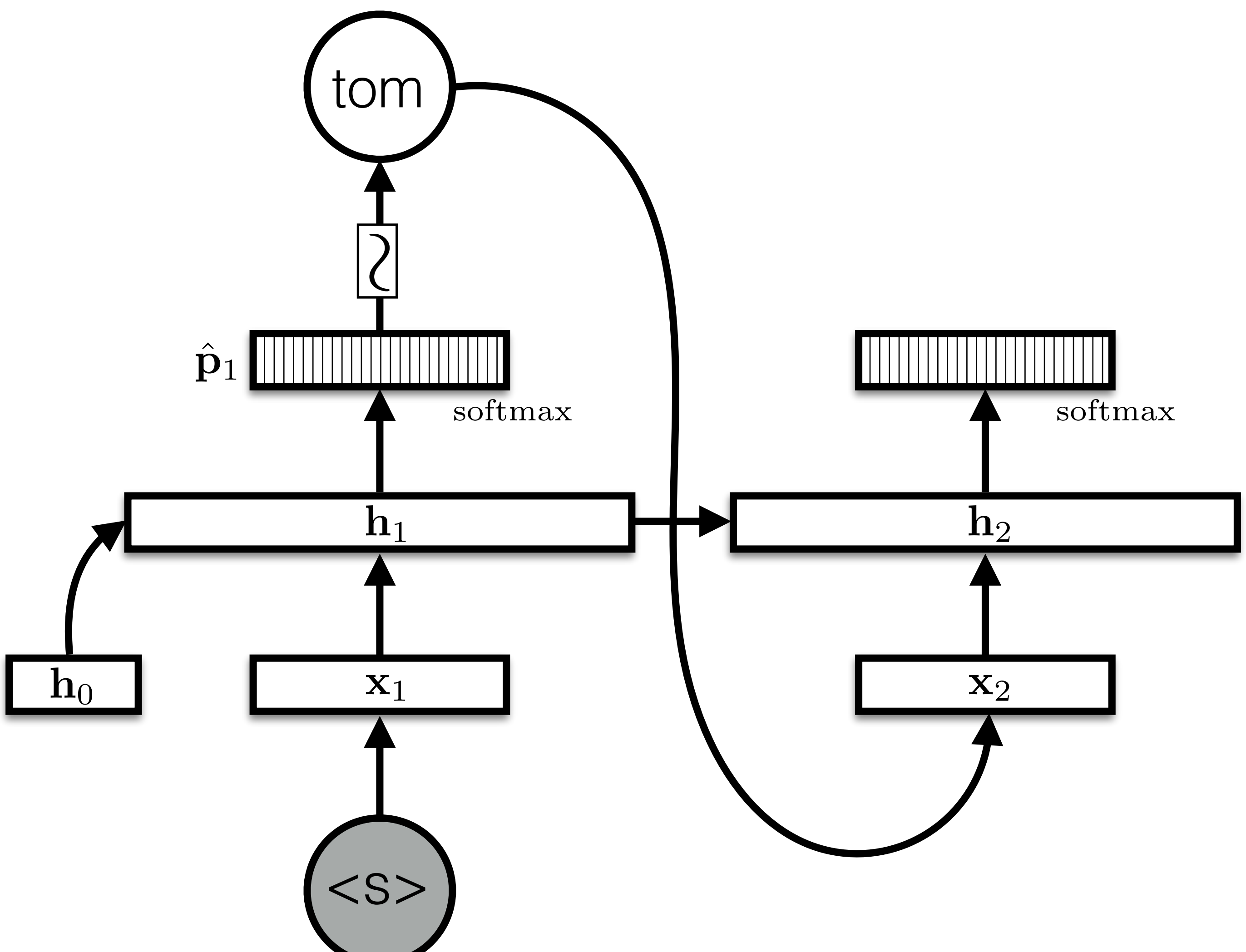
RNN language models

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



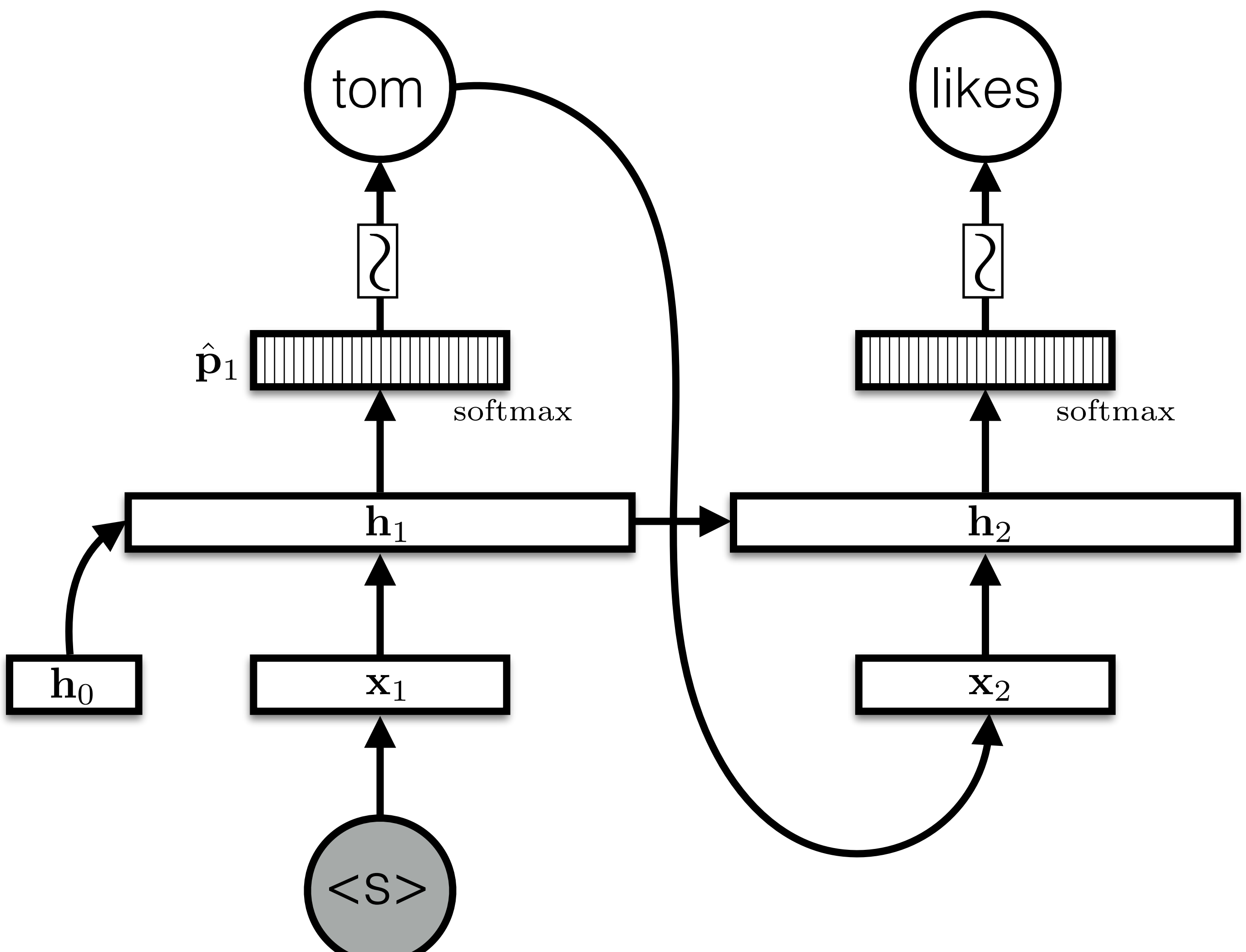
RNN language models

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



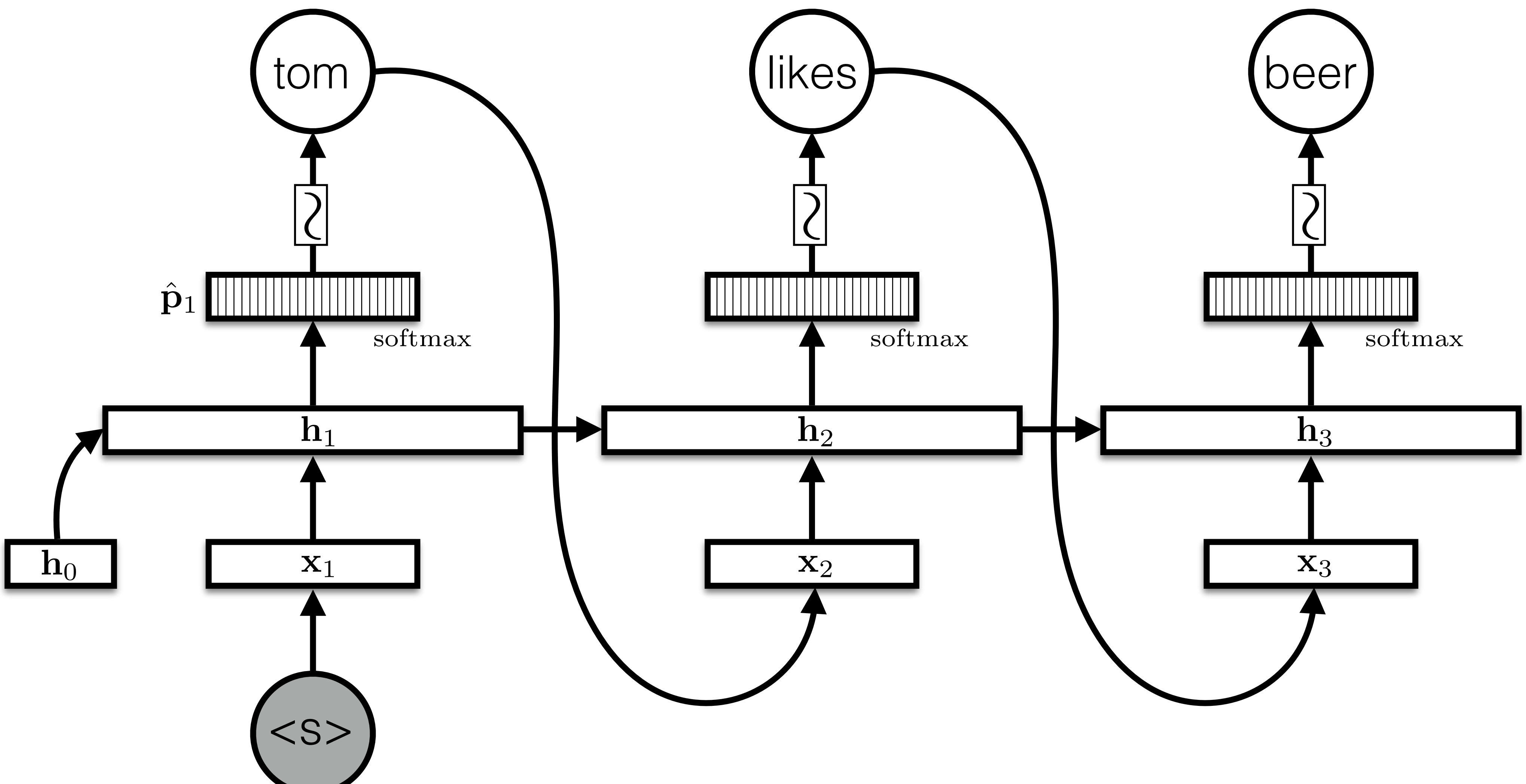
RNN language models

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



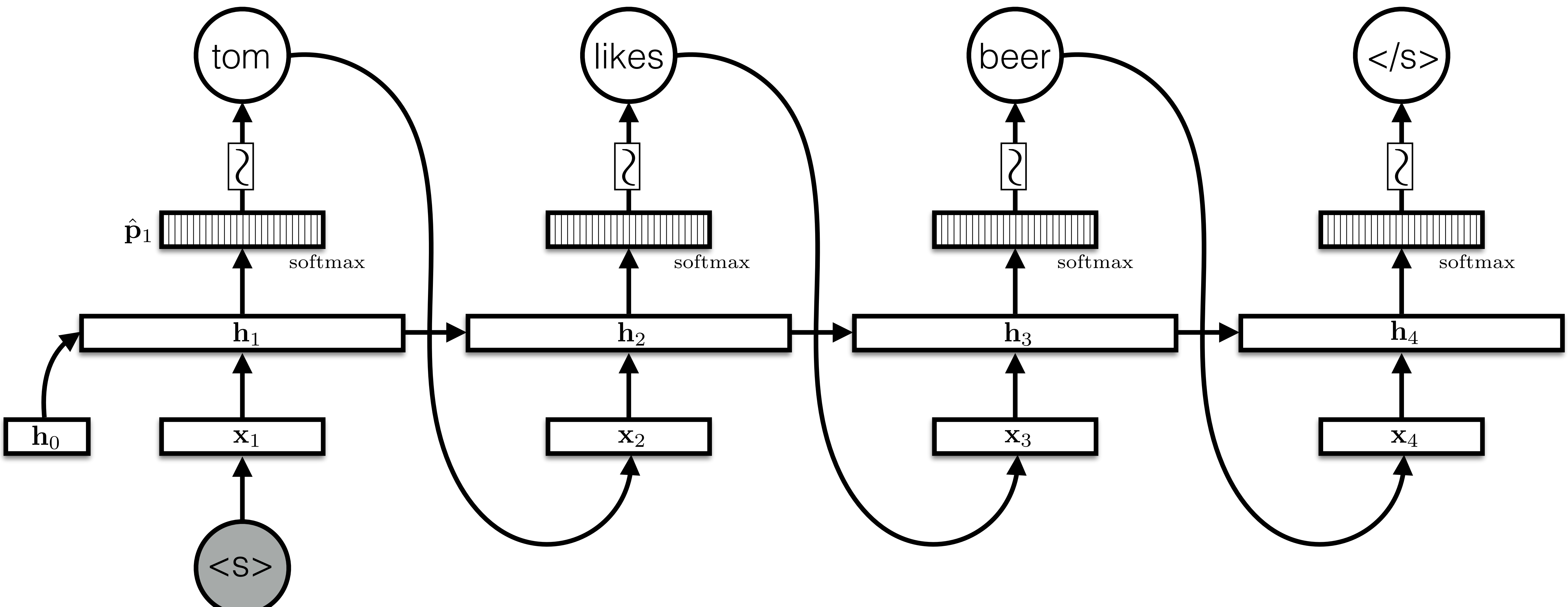
RNN language models

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

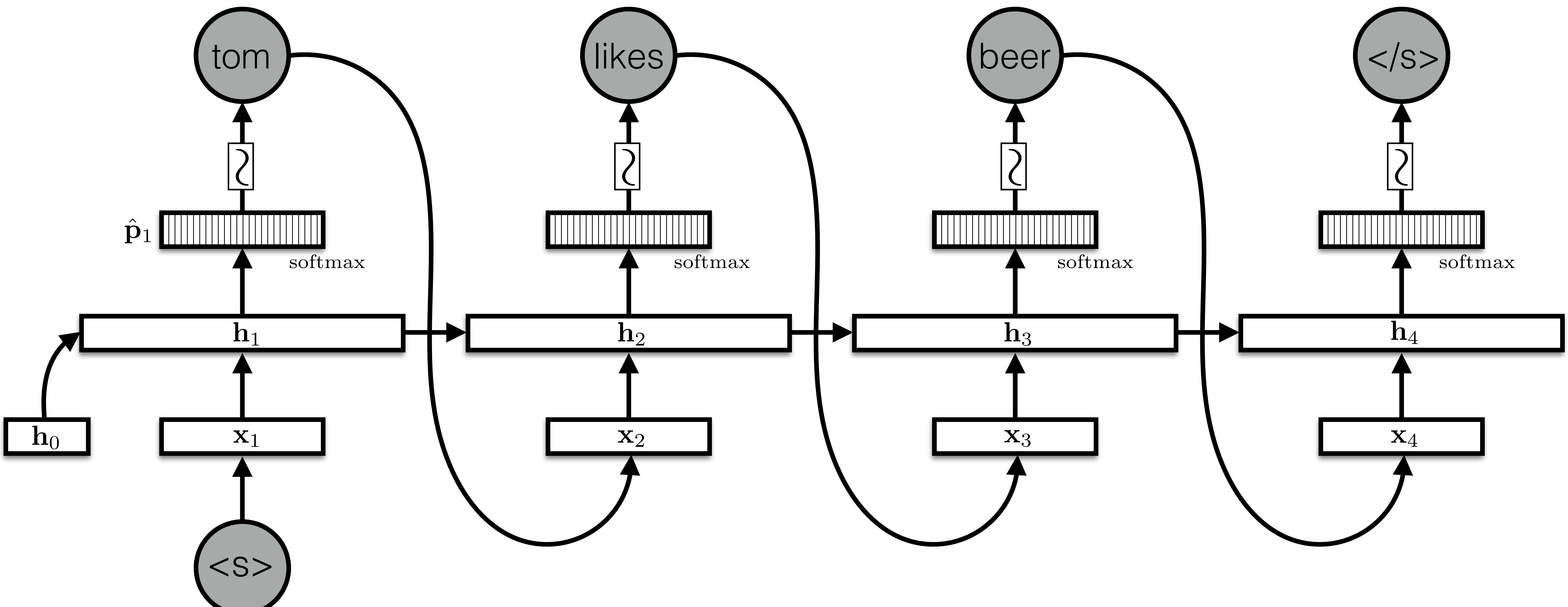


RNN language models

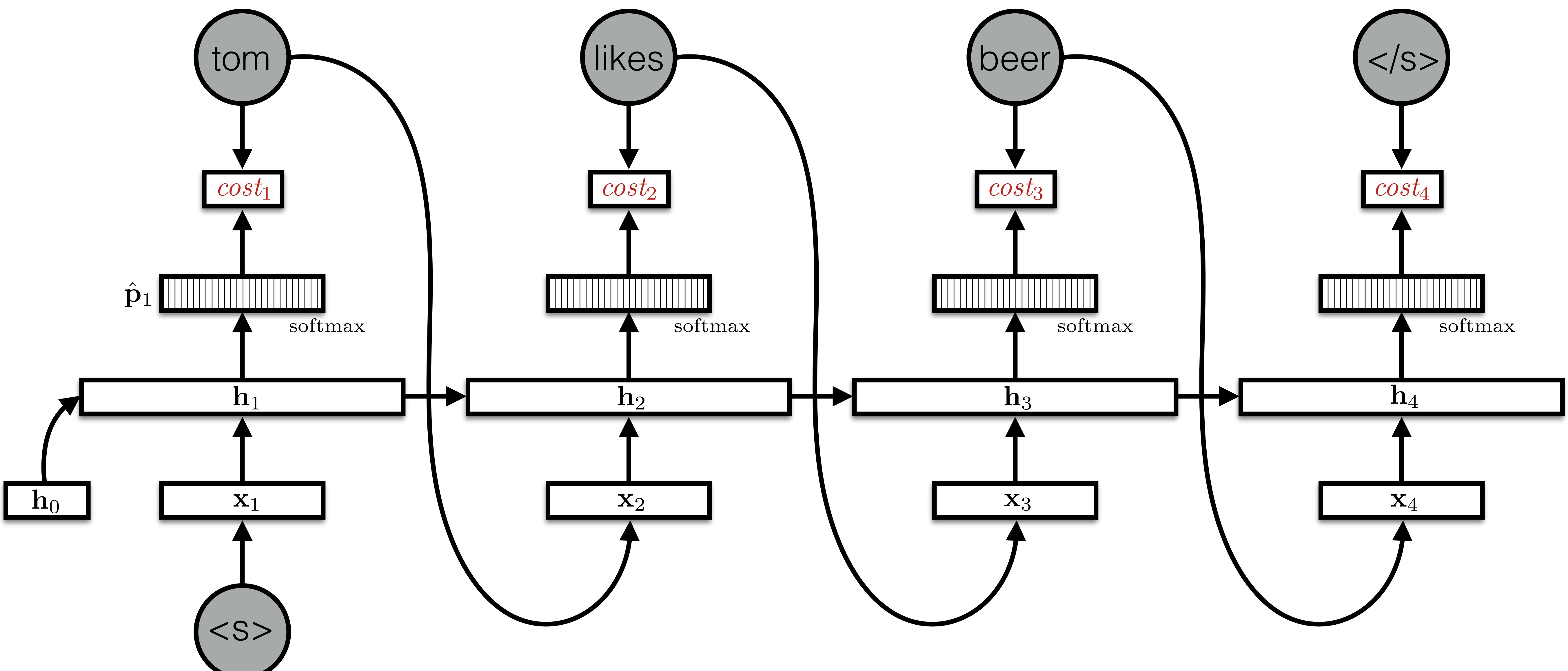
$$p(tom \mid \langle \mathbf{s} \rangle) \times p(likes \mid \langle \mathbf{s} \rangle, tom)$$
$$\times p(beer \mid \langle \mathbf{s} \rangle, tom, likes)$$
$$\times p(\langle / \mathbf{s} \rangle \mid \langle \mathbf{s} \rangle, tom, likes, beer)$$



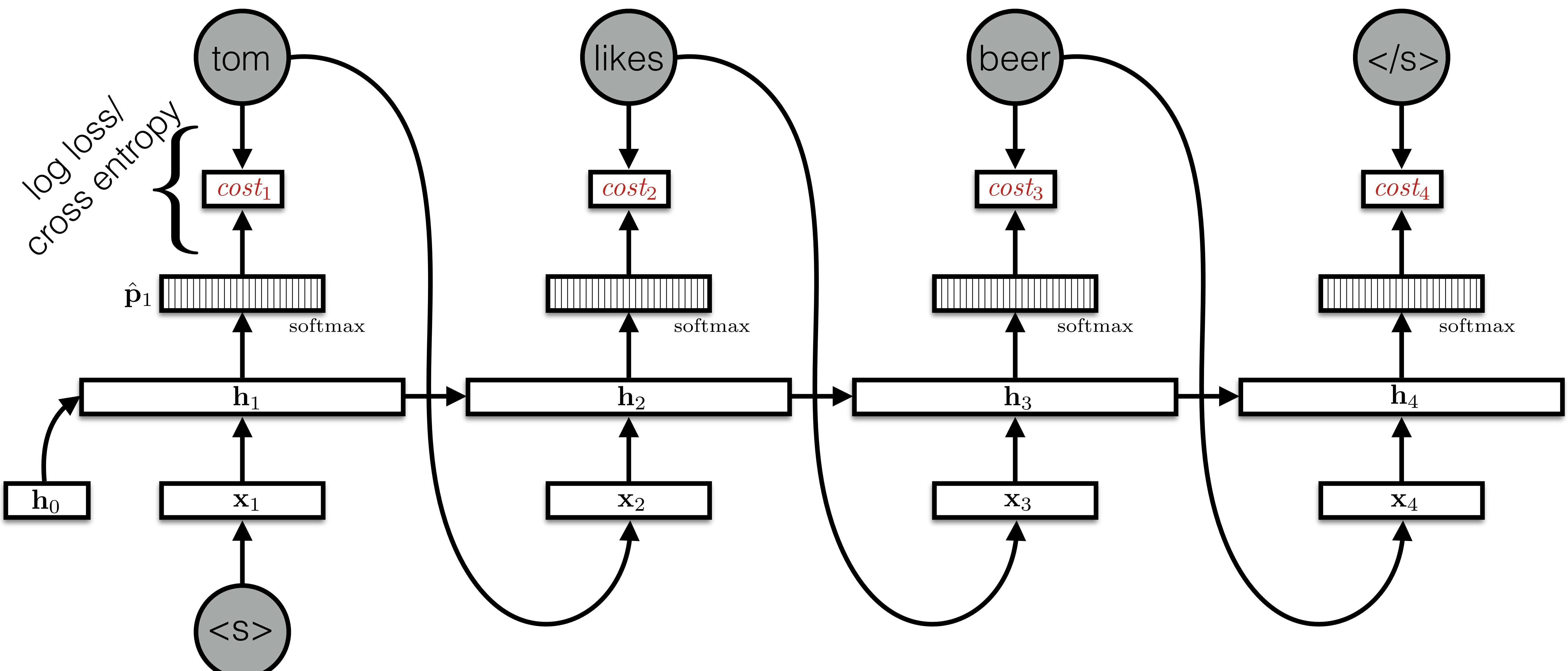
Training RNN language models



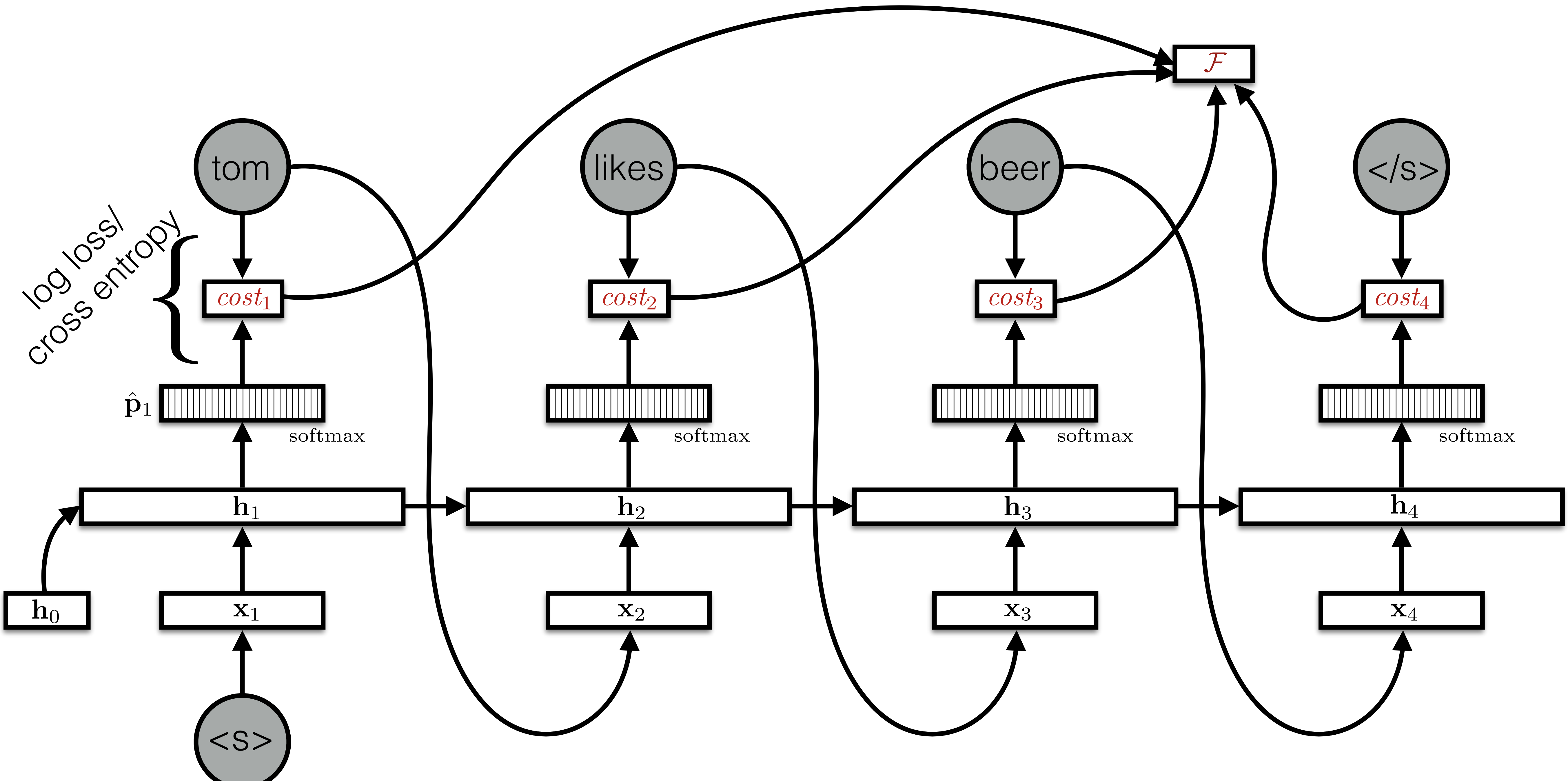
Training RNN language models



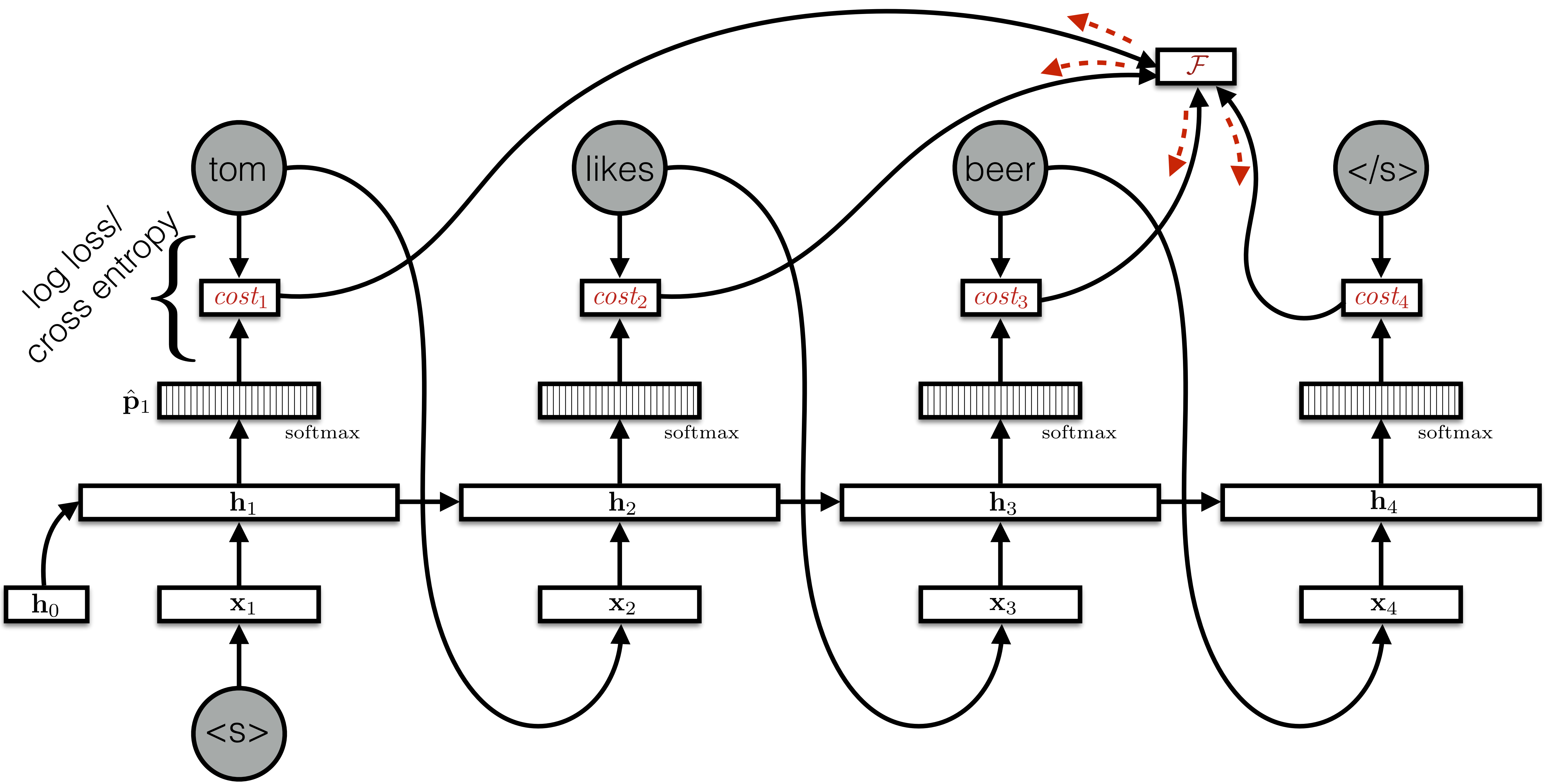
Training RNN language models



Training RNN language models



Training RNN language models



Training RNN language models

The cross-entropy objective seeks the **maximum likelihood** (MLE) objective.

“Find the parameters that make the training data most likely.”

Training RNN language models

The cross-entropy objective seeks the **maximum likelihood** (MLE) objective.

“Find the parameters that make the training data most likely.”

You *will* overfit.

1. Stop training early, based on a validation set
2. Weight decay / other regularizers
3. “Dropout” during training.

In contrast to count-based models, zeroes aren’t a problem.

RNN language models

- Unlike Markov (*n*-gram) models, RNNs never forget
    - However, they don't always remember so well (recall Felix's lectures on RNNs vs. LSTMs)
- Algorithms
    - Sample a sequence from the probability distribution defined by the RNN
    - Train the RNN to minimize cross entropy (aka MLE)
    - What about: what is the most probable sequence?

How well do RNN LMs do?

	perplexity
order=5 Markov Kneser-Ney freq. est.	221
RNN 400 hidden	171
3xRNN interpolation	**151**

Mikolov et al. (2010 Interspeech) "Recurrent neural network based language model"

How well do RNN LMs do?

	perplexity	Word Error Rate (WER)
order=5 Markov Kneser-Ney freq. est.	221	13.5
RNN 400 hidden	171	12.5
3xRNN interpolation	**151**	**11.6**

Mikolov et al. (2010 Interspeech) “Recurrent neural network based language model”

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

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

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

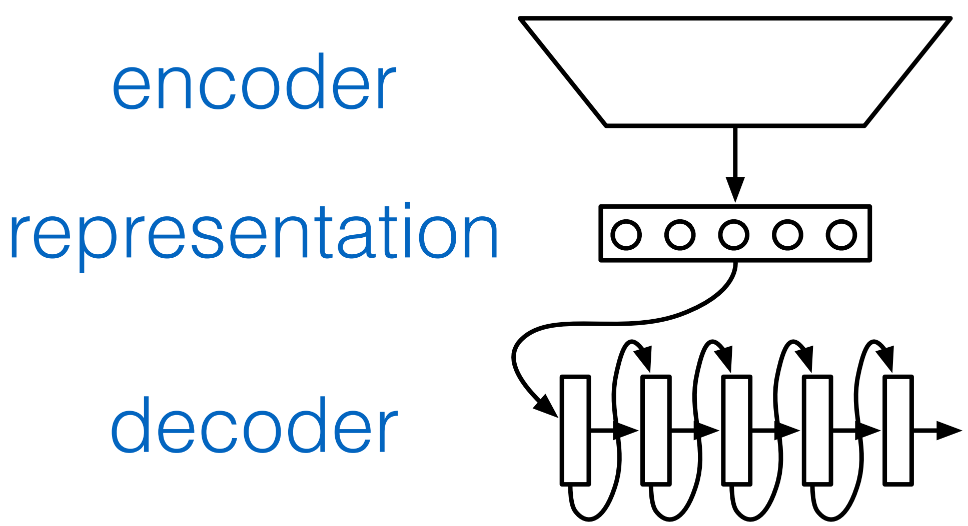
Human evaluation.

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, $\boldsymbol{w}$.

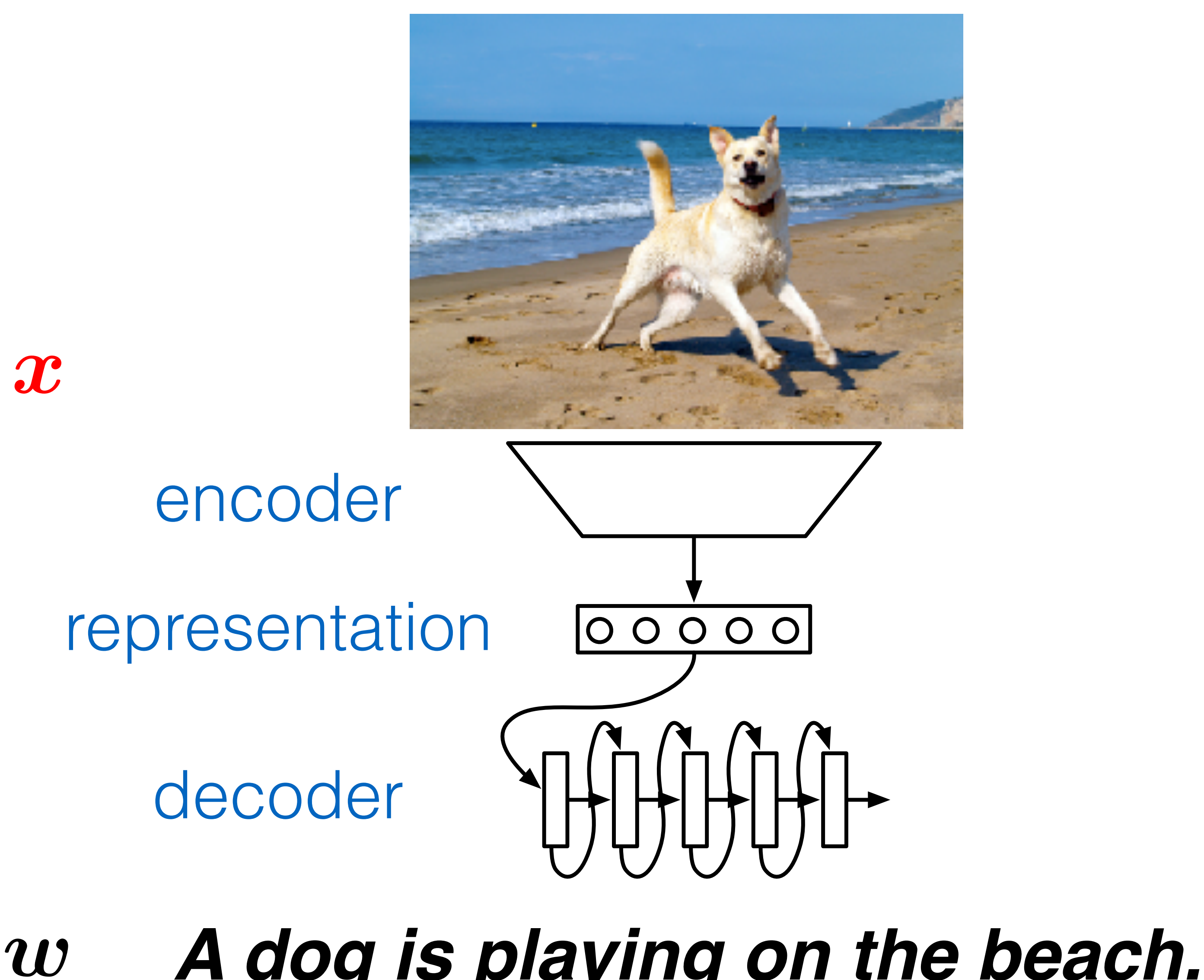
$\boldsymbol{x}$ ***Kunst kann nicht gelehrt werden…***



$\boldsymbol{w}$ ***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, $\boldsymbol{w}$.



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 one standard solution: RNNs

Kalchbrenner and Blunsom 2013

Encoder

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

Kalchbrenner and Blunsom 2013

Encoder

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

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

Recurrent decoder

Recurrent connection

Embedding of w_{t-1}

Source sentence

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

Learnt bias

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

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

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

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

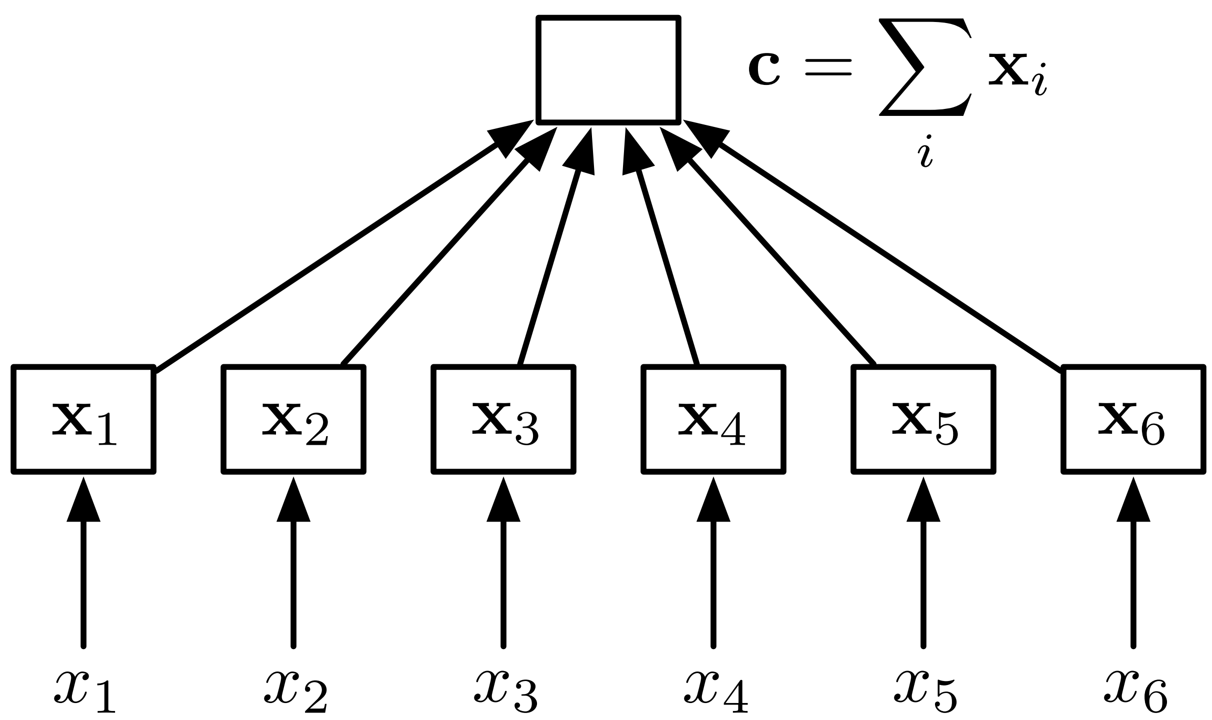
Recall unconditional RNN

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

K&B 2013: Encoder

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

The simplest model possible:



What do you think of this model?

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

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

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

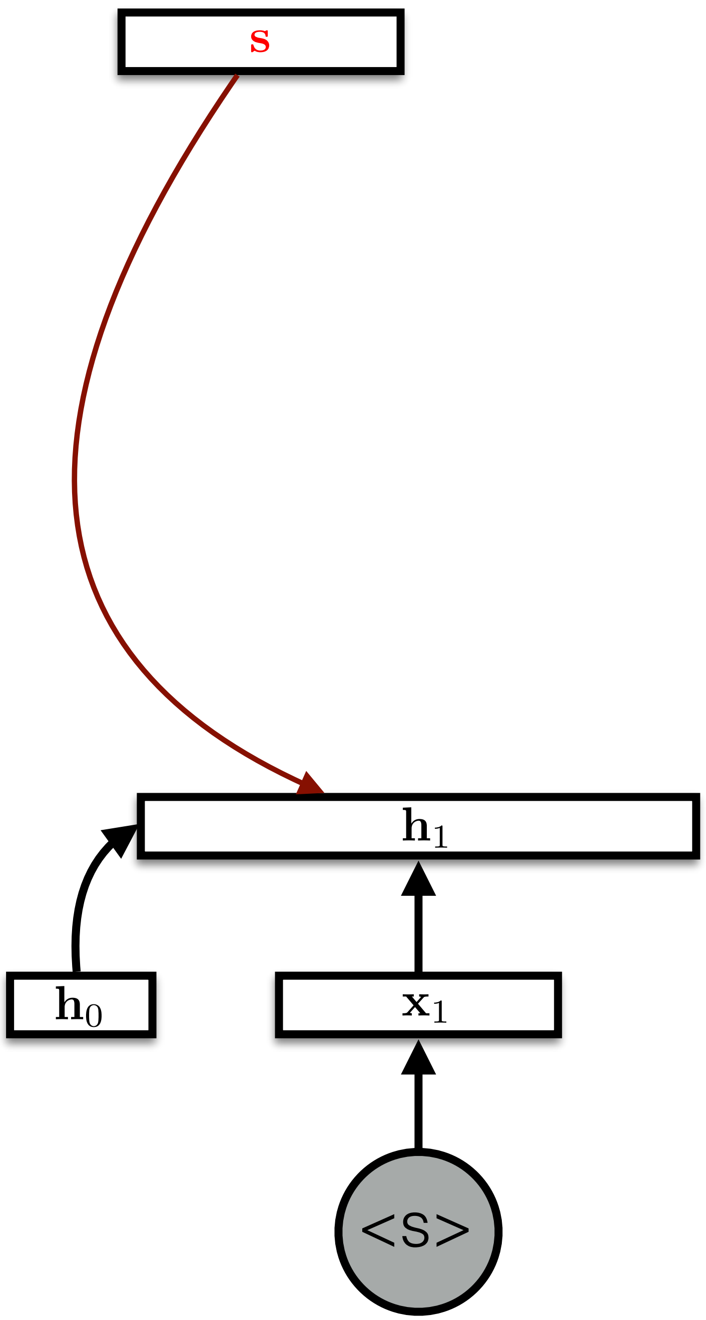
Learnt bias

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

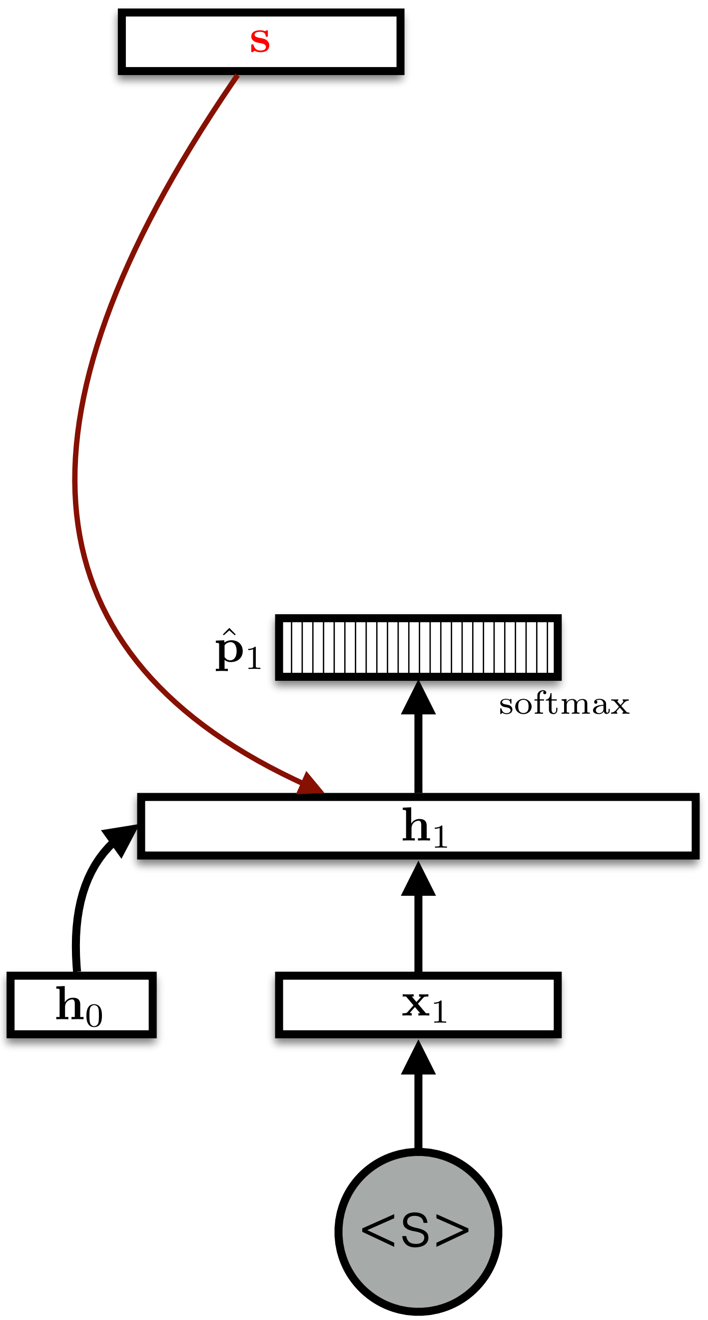
Recall unconditional RNN

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

K&B 2013: RNN Decoder

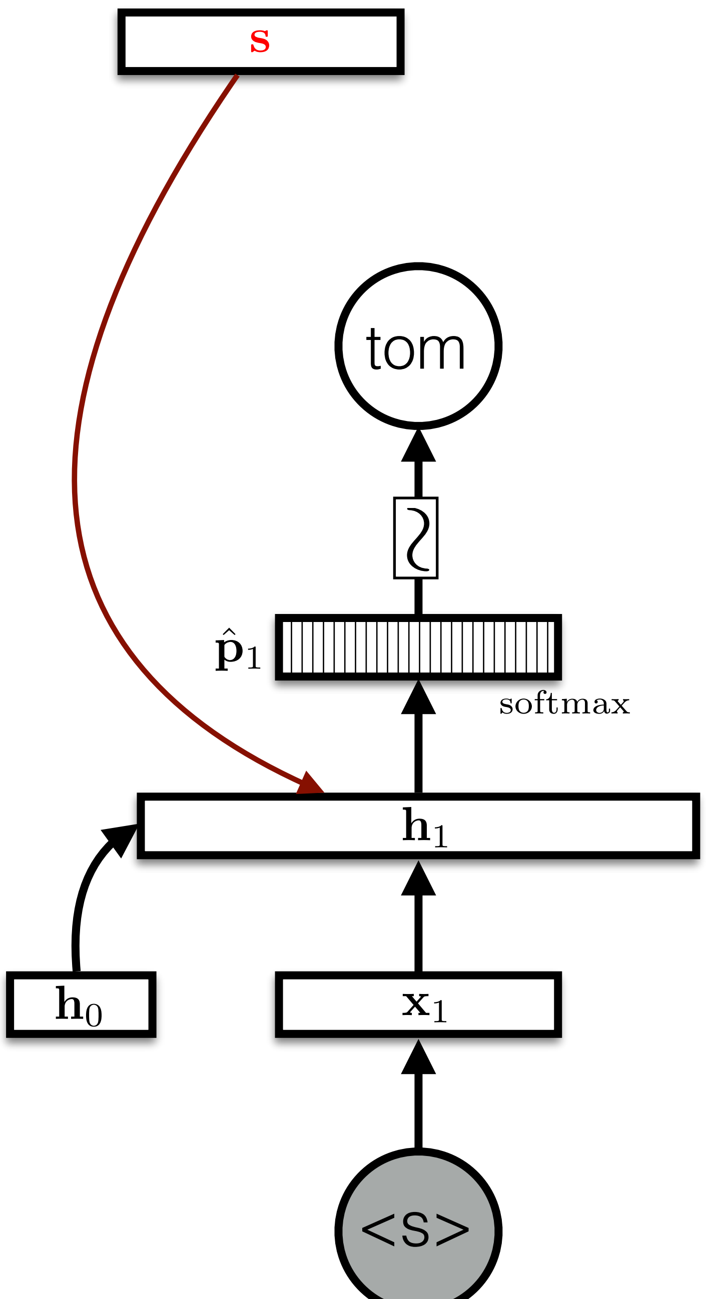


K&B 2013: RNN Decoder



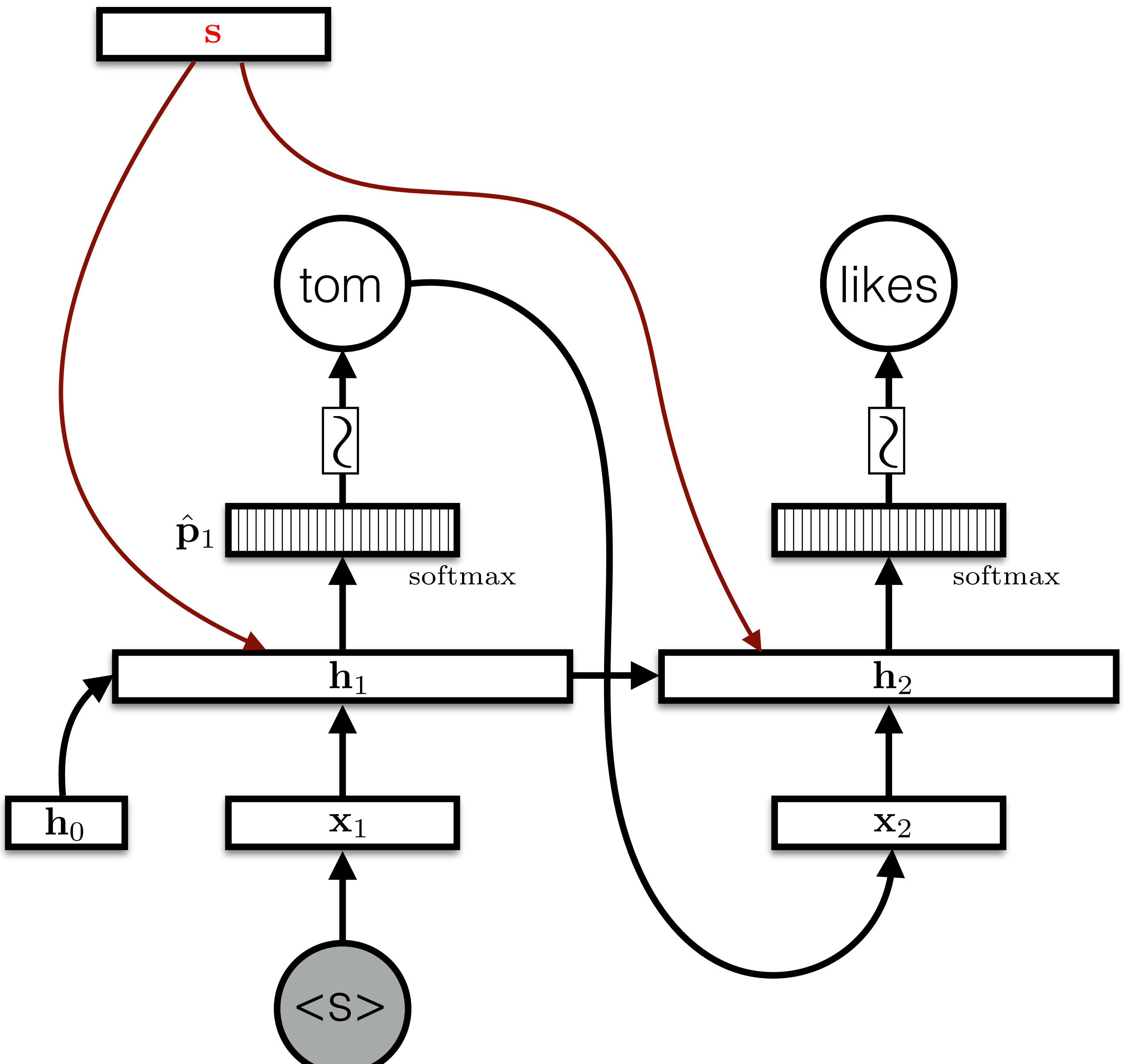
K&B 2013: RNN Decoder

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



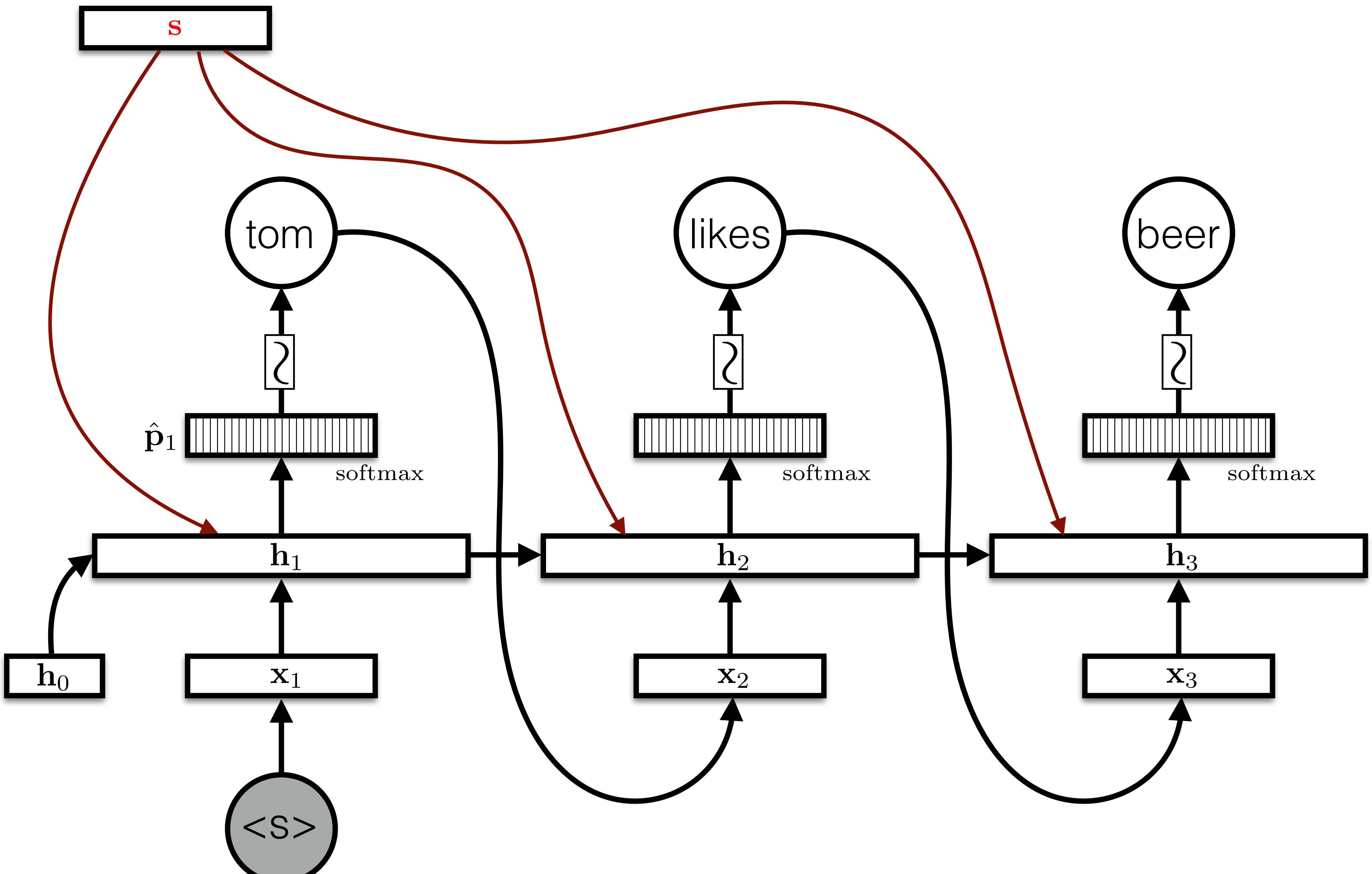
K&B 2013: RNN Decoder

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



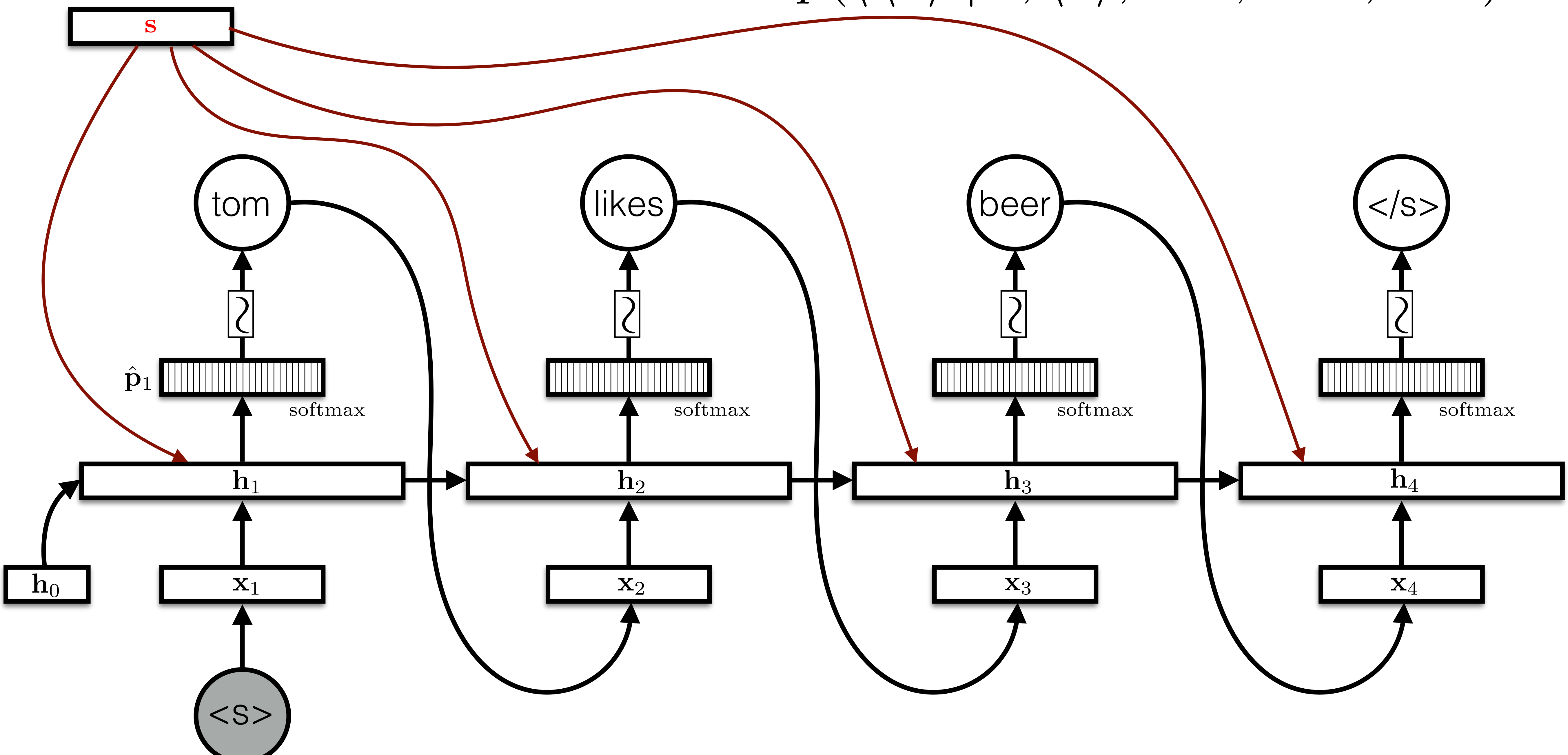
K&B 2013: RNN Decoder

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



K&B 2013: RNN Decoder

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



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.

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

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

undecidable :(

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

$$w_1^* \approx \arg\max_{w_1} p(w_1 \mid \boldsymbol{x})$$

$$w_2^* \approx \arg\max_{w_2} p(w_2 \mid \boldsymbol{x}, w_1^*)$$

$$\vdots$$

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

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:

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

beer **drink** **I**

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

w_0 w_1 w_2 w_3

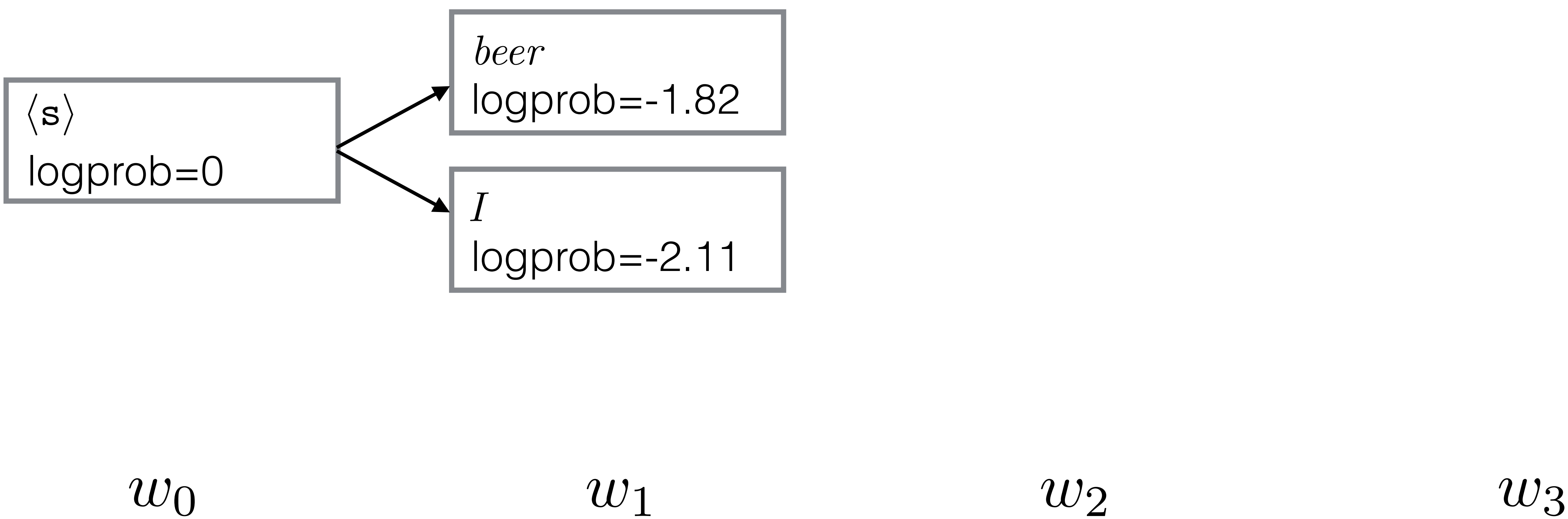
A word about decoding

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:

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

beer **drink** **I**



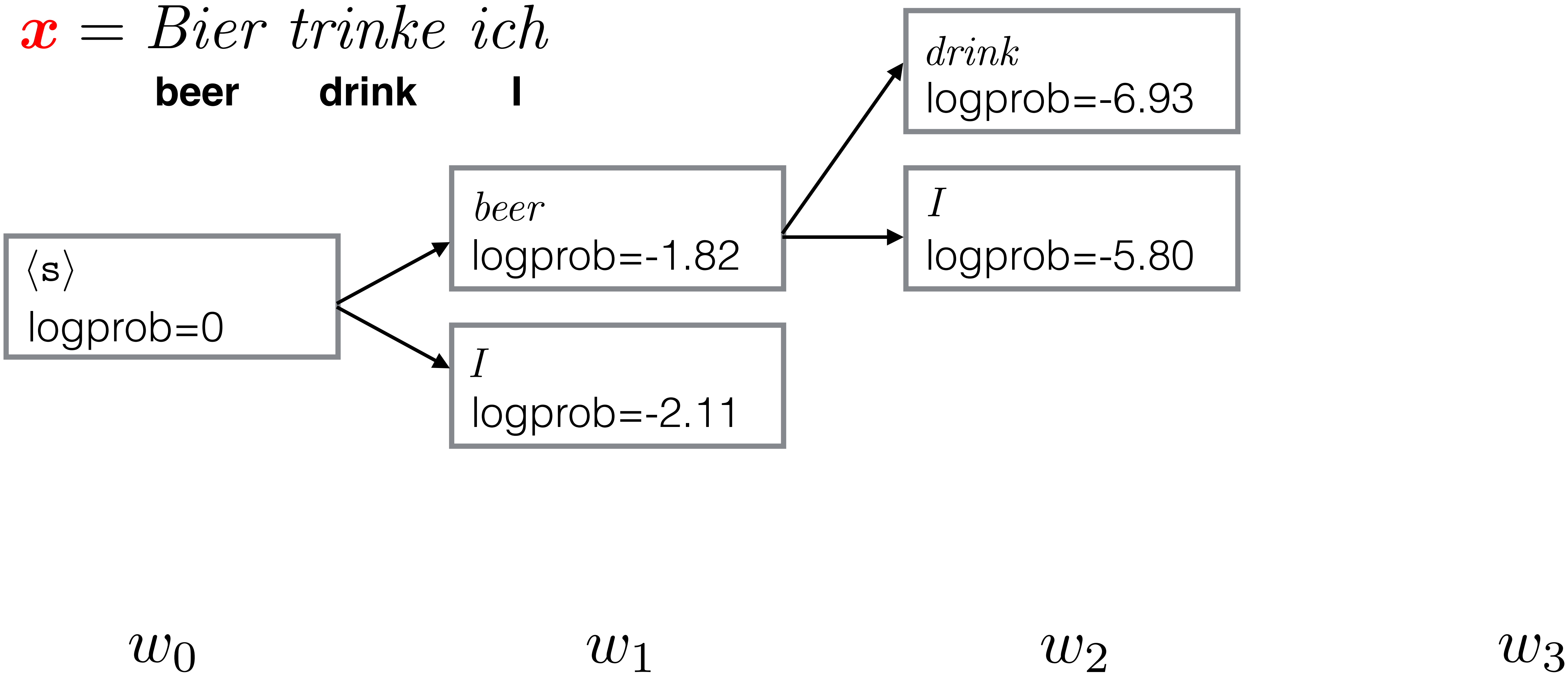
A word about decoding

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:

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

beer **drink** **I**



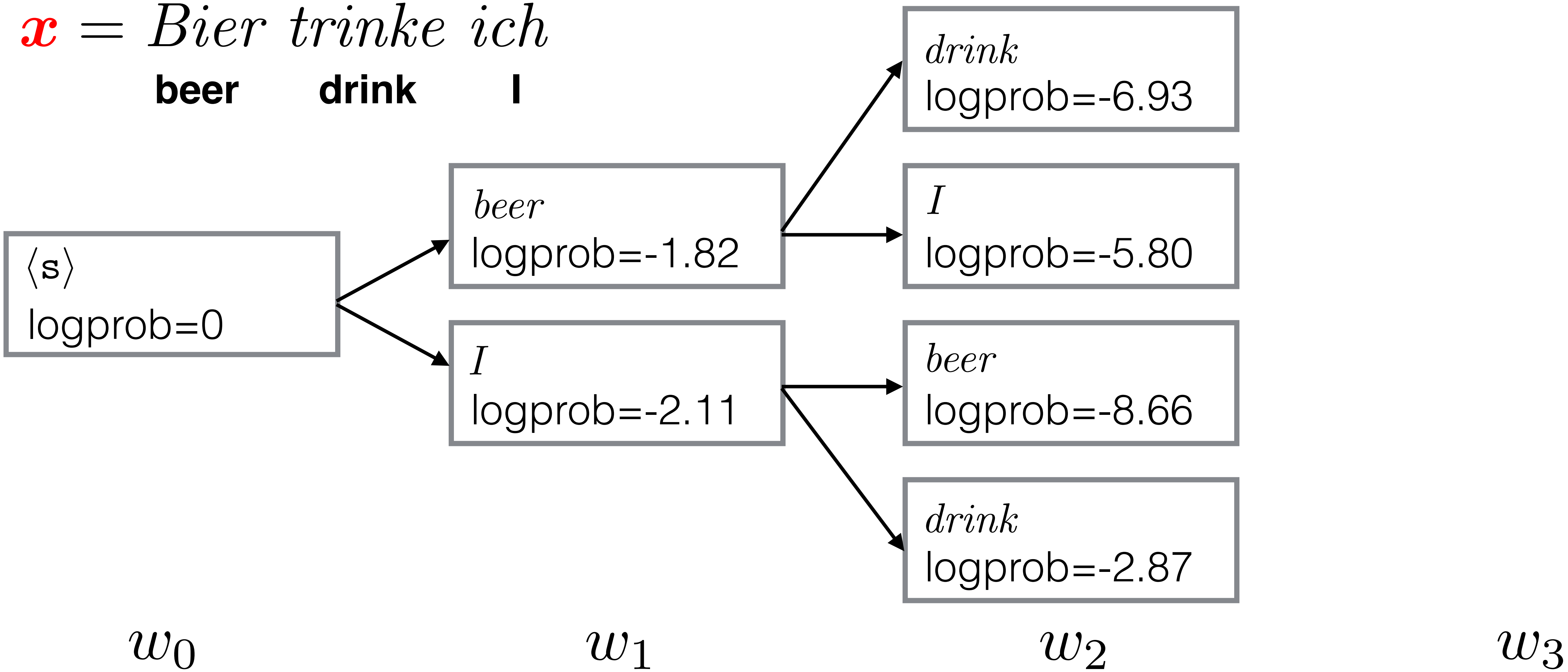
A word about decoding

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:

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

beer **drink** **I**



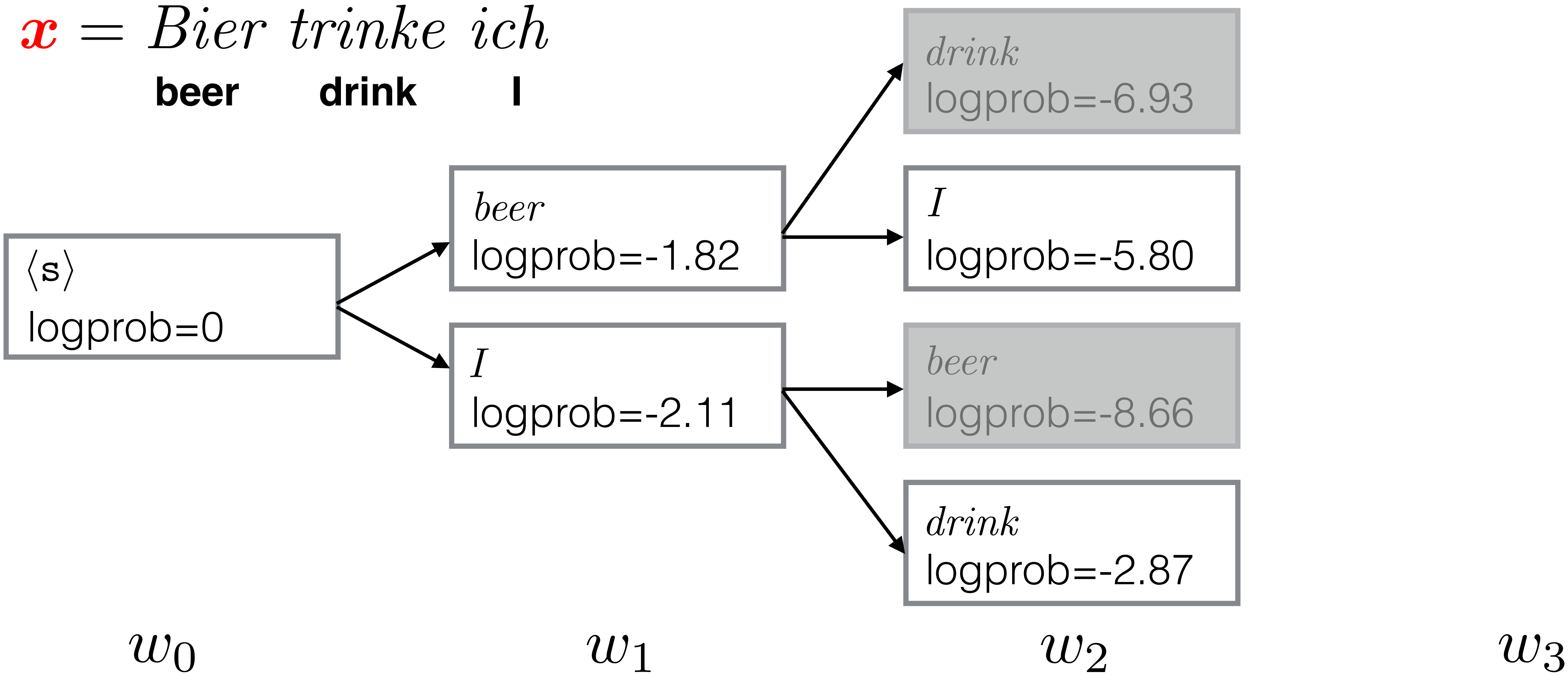
A word about decoding

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:

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

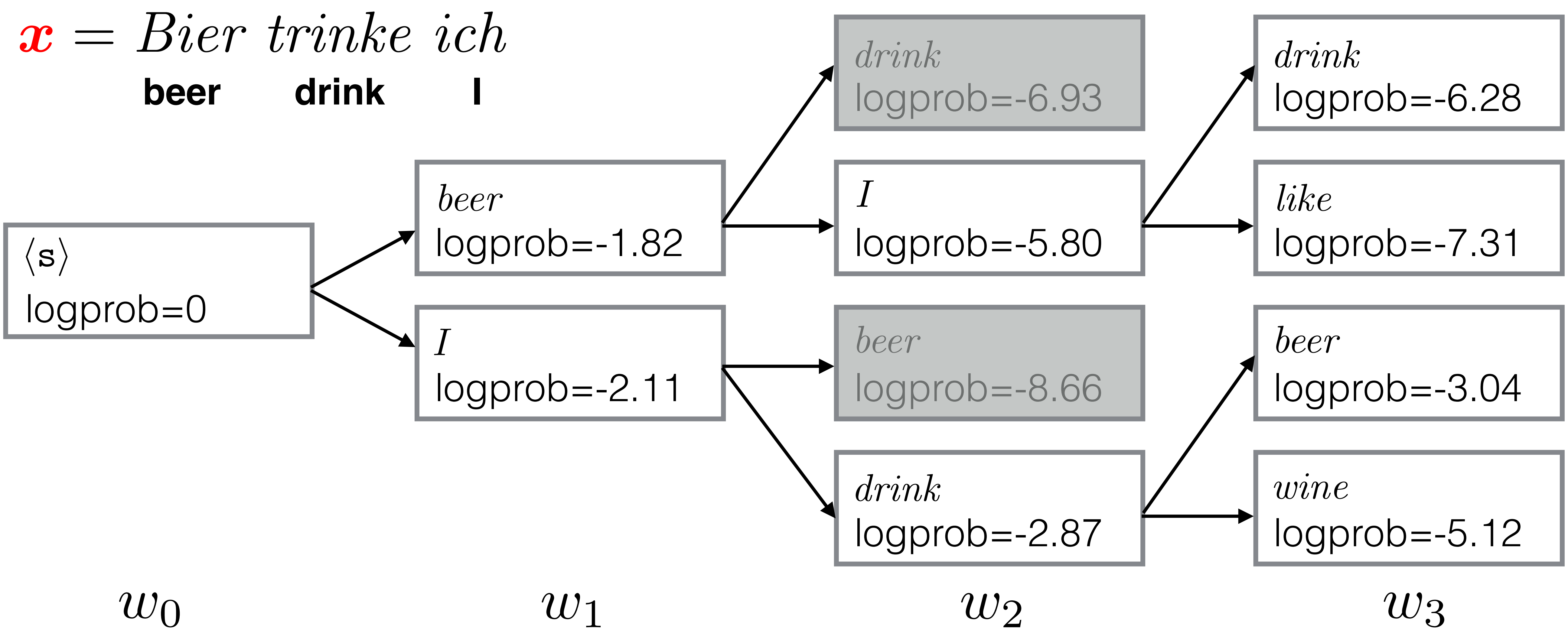
beer **drink** **I**



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.

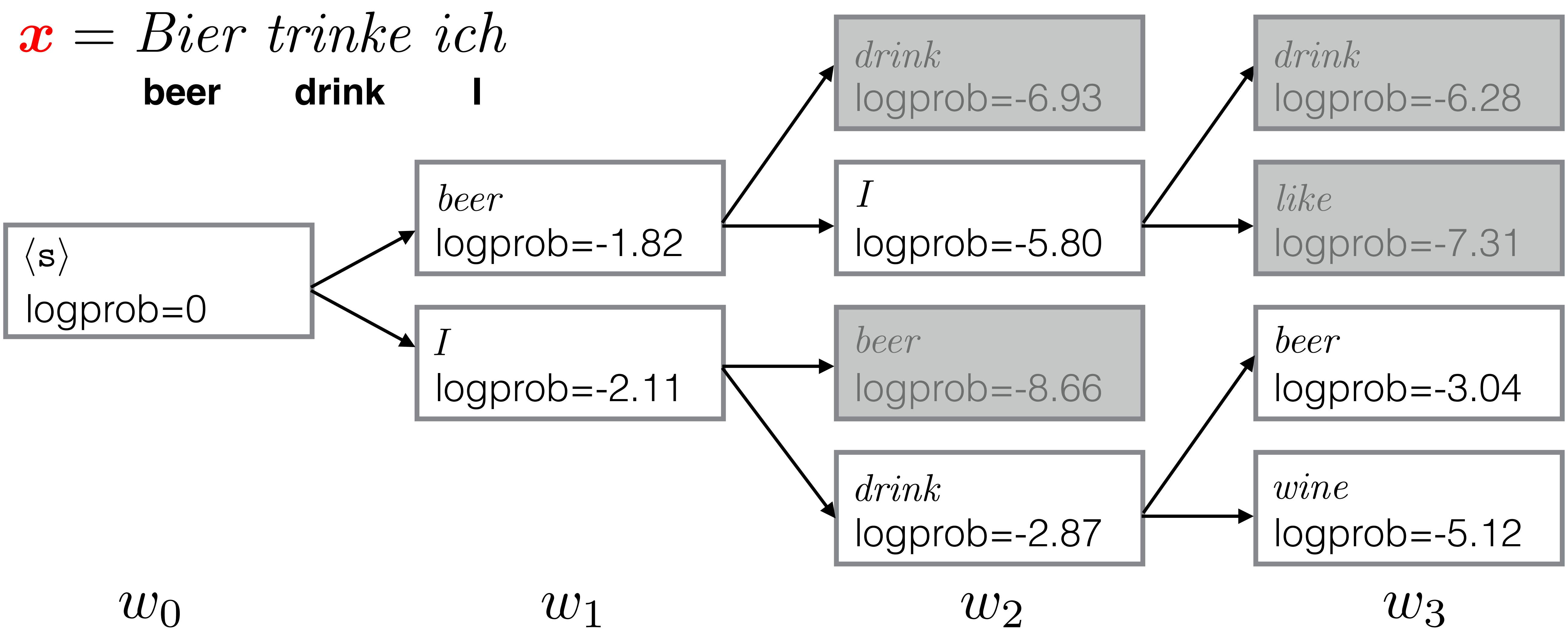
E.g., for *b*=2:



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.

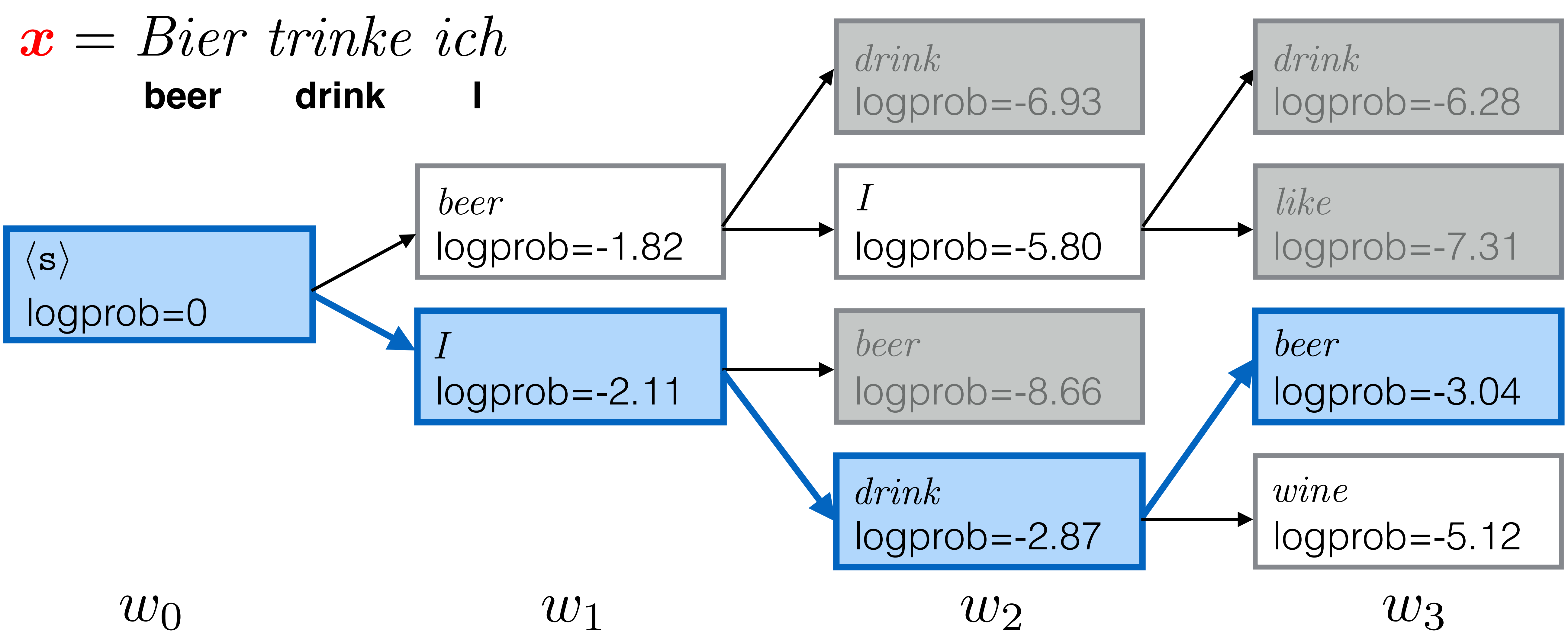
E.g., for *b*=2:



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:



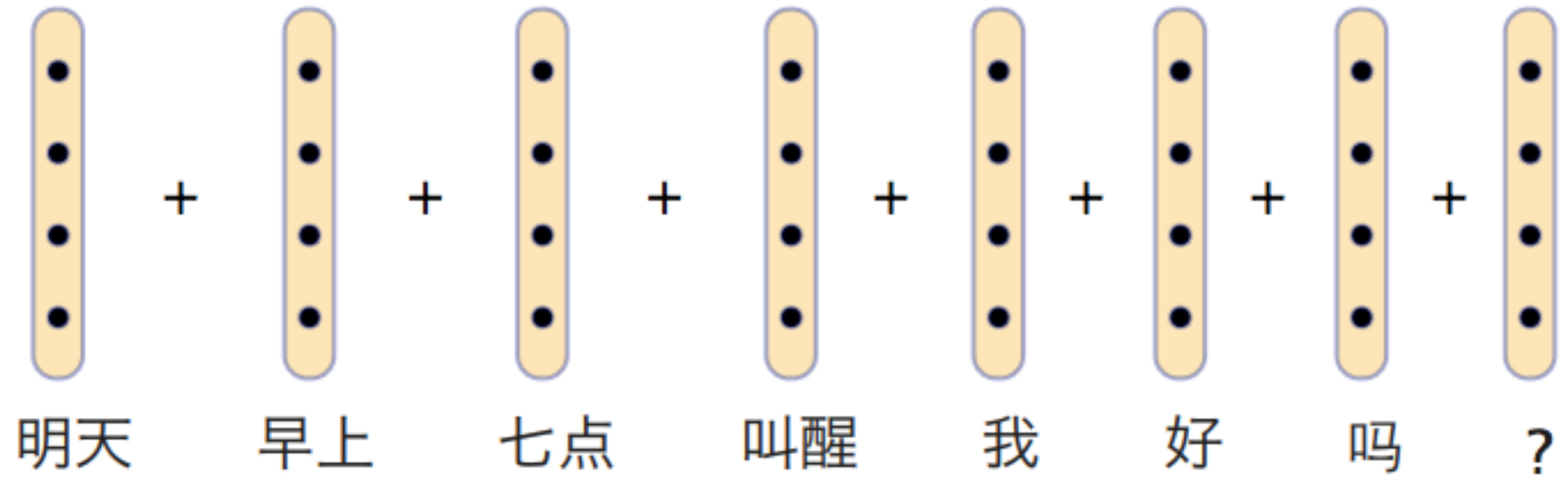
How well does this model do?

	perplexity (2011)	perplexity (2012)
order=5 Markov Kneser-Ney freq. est.	222	225
RNN LM	178	181
RNN LM + x	**140**	**142**

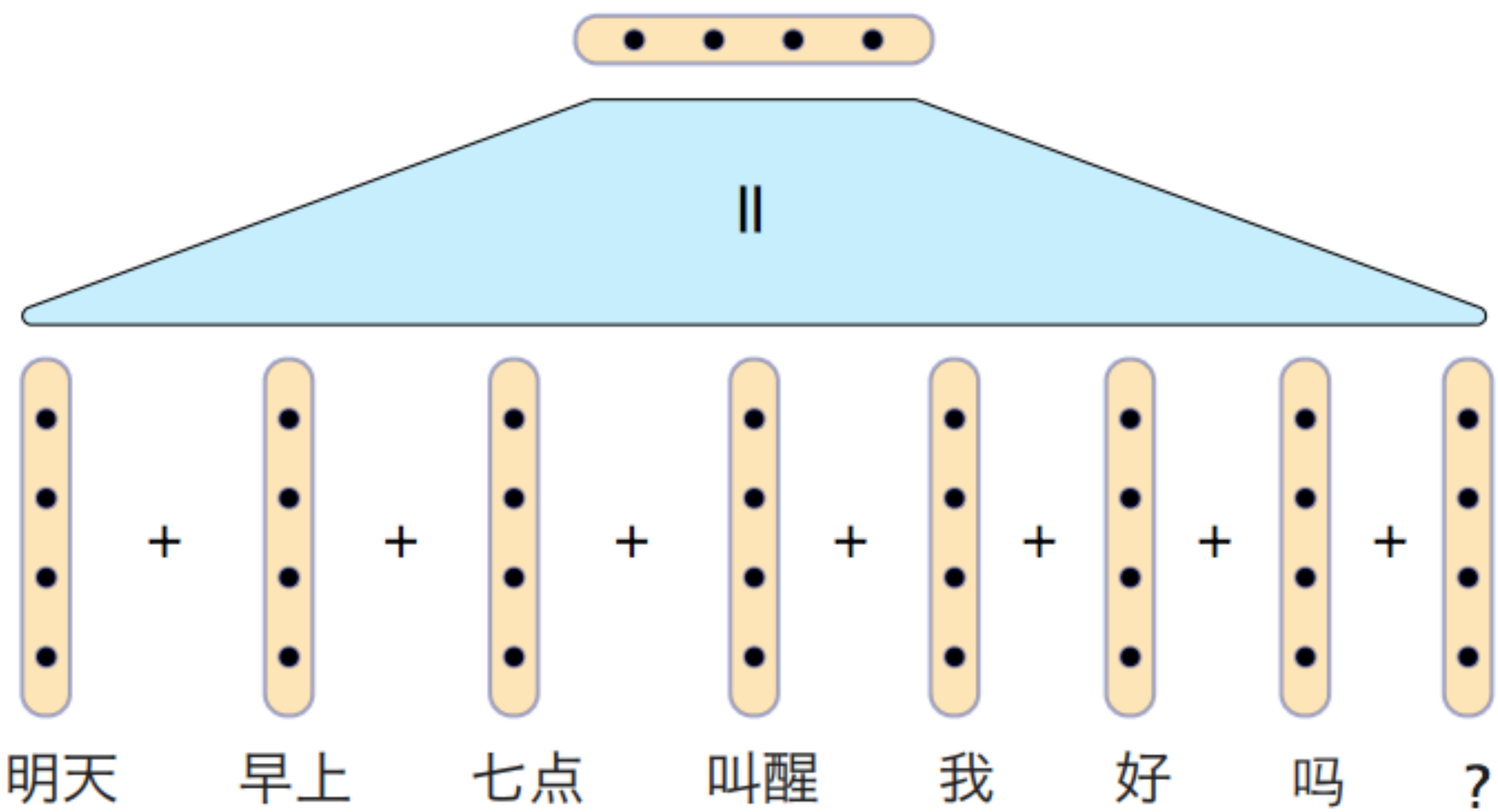
How well does this model do?

明天　早上　七点　叫醒　我　好　吗　?

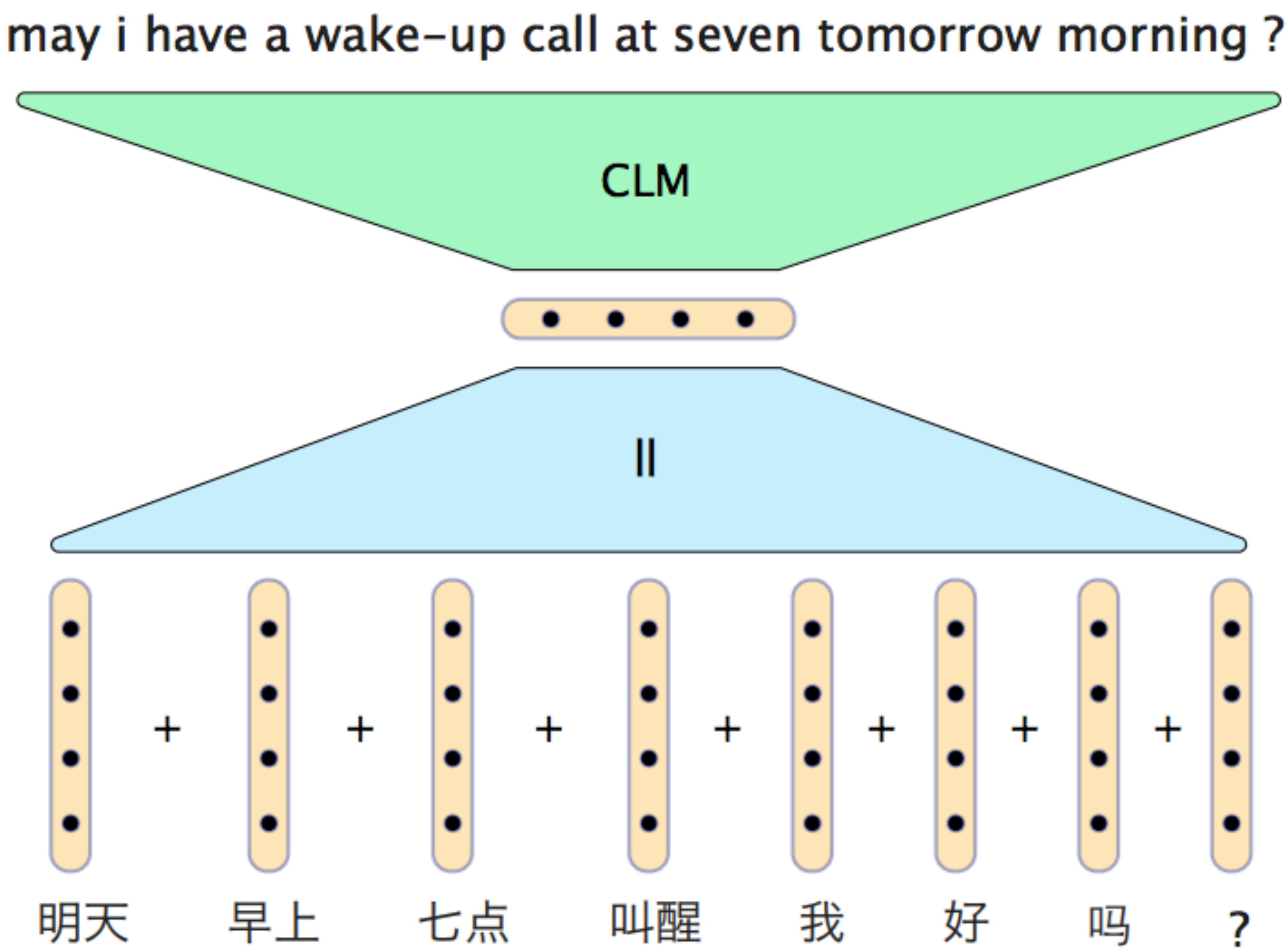
How well does this model do?



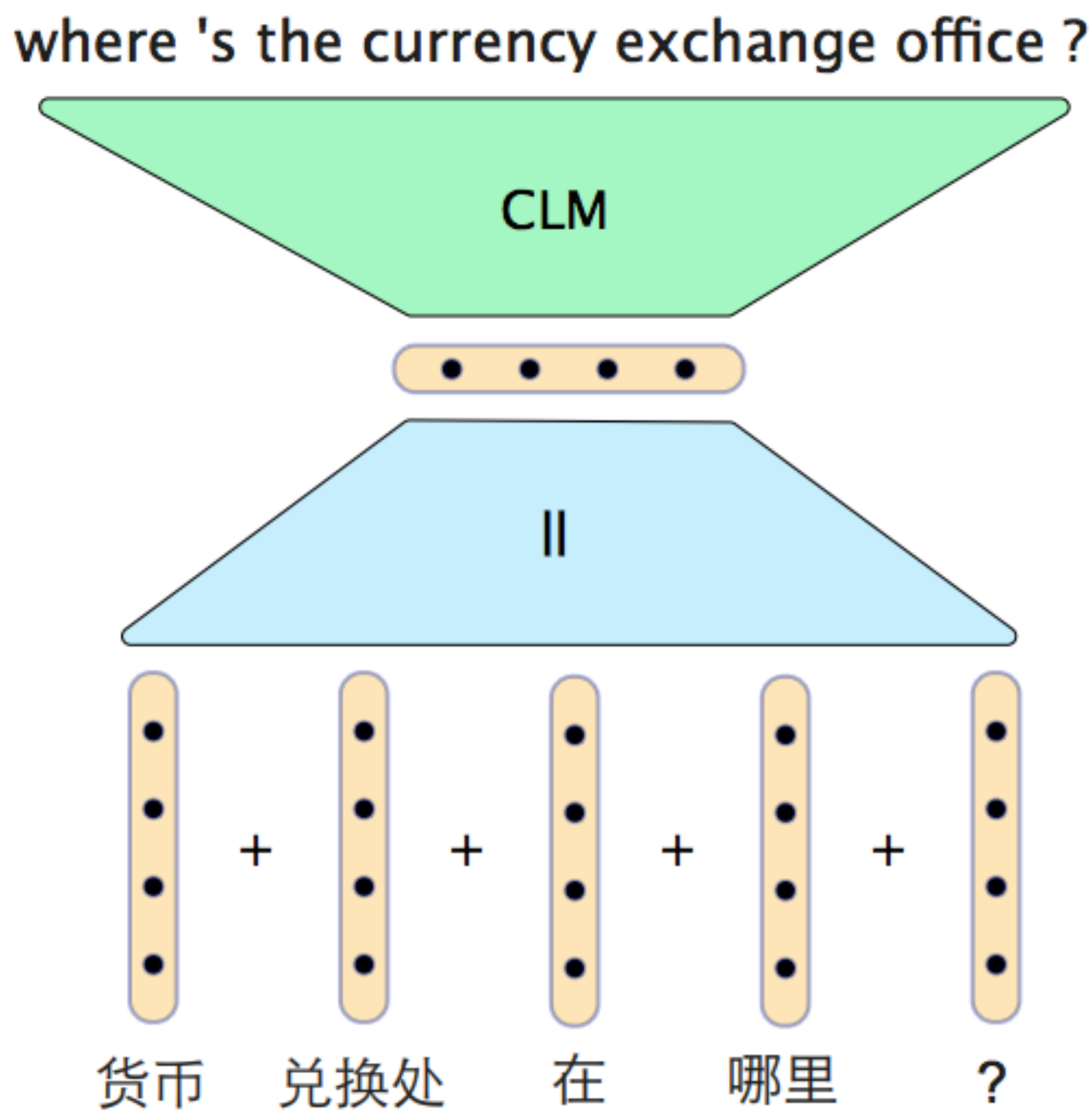
How well does this model do?



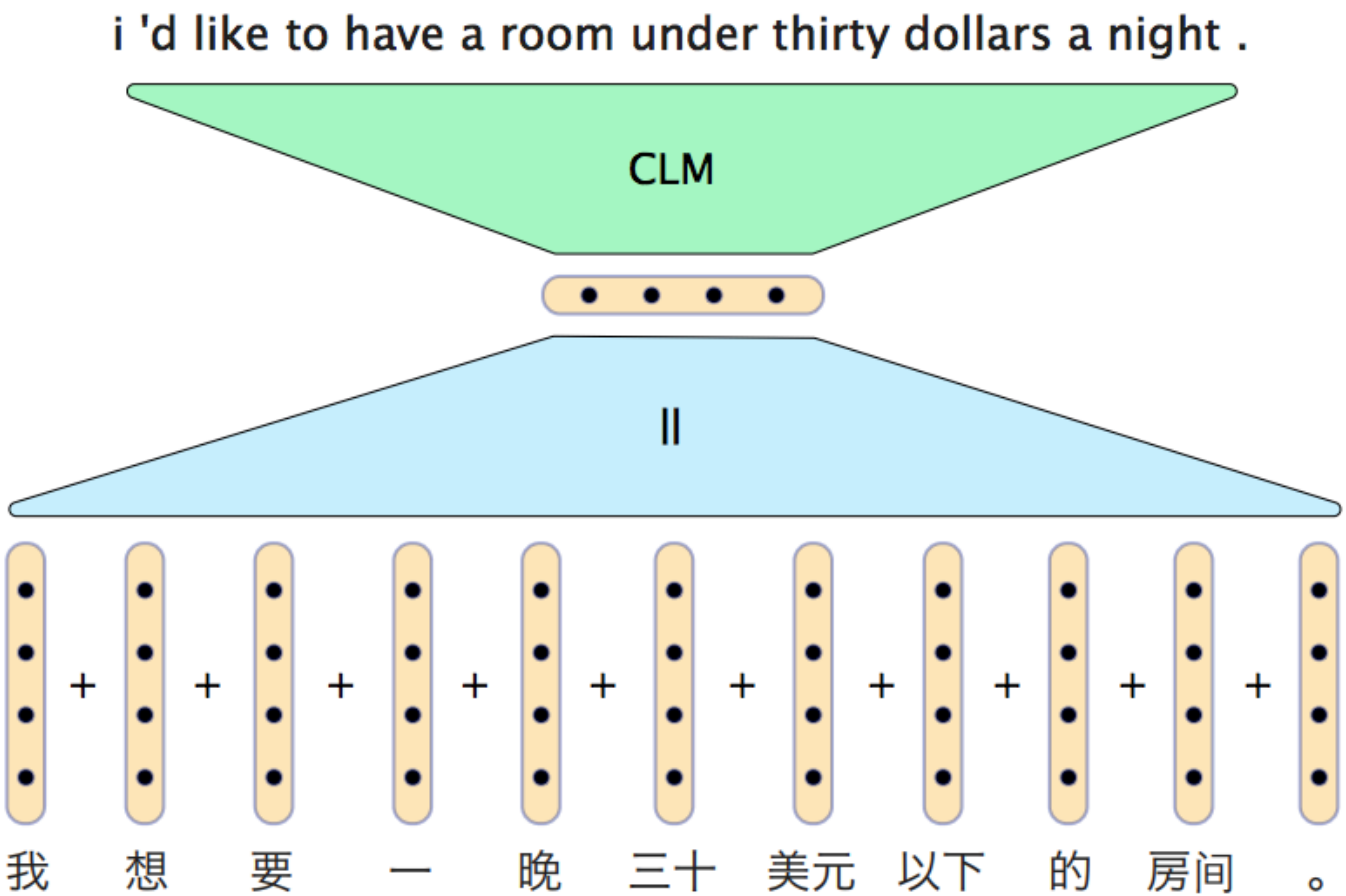
How well does this model do?



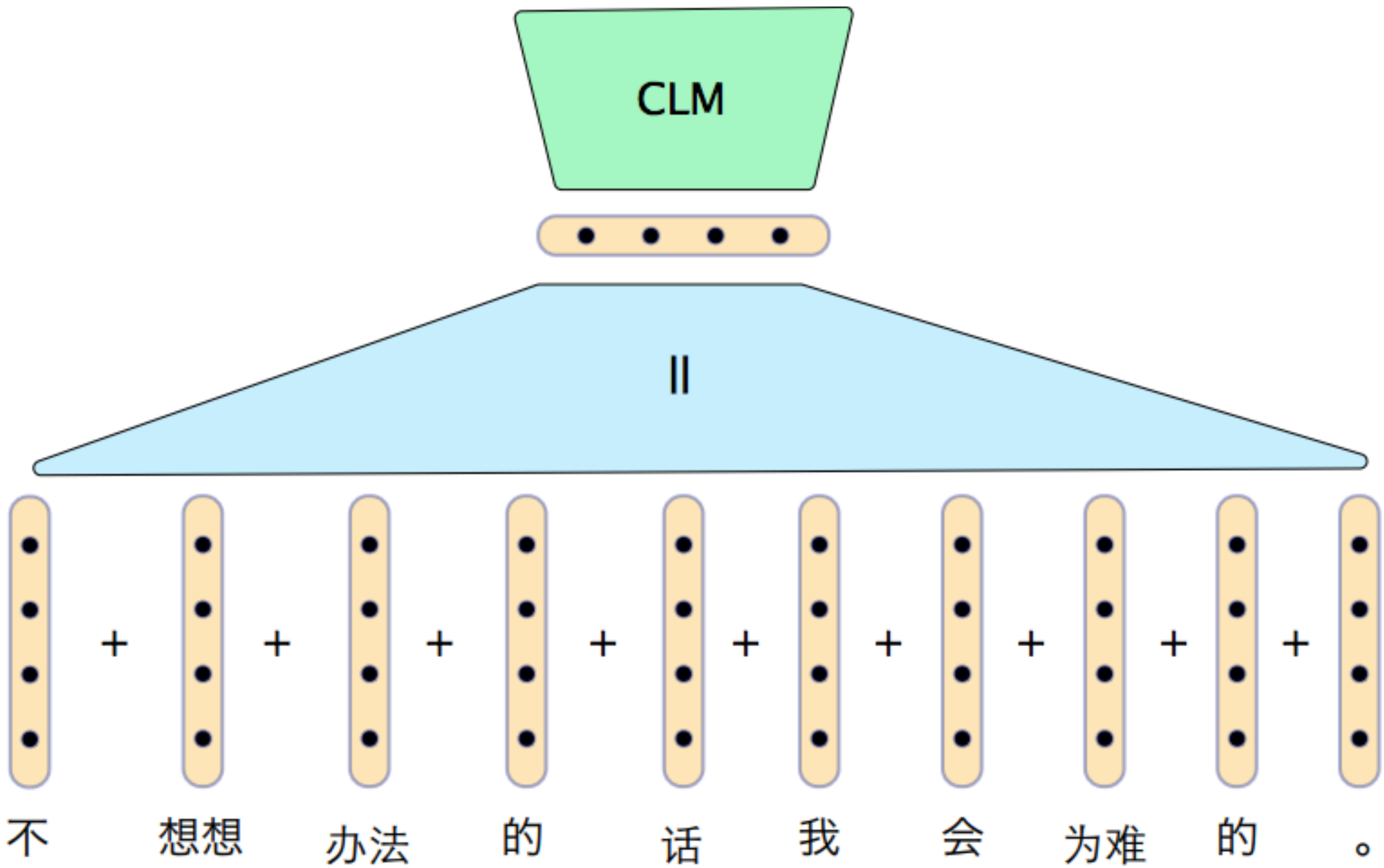
How well does this model do?



How well does this model do?

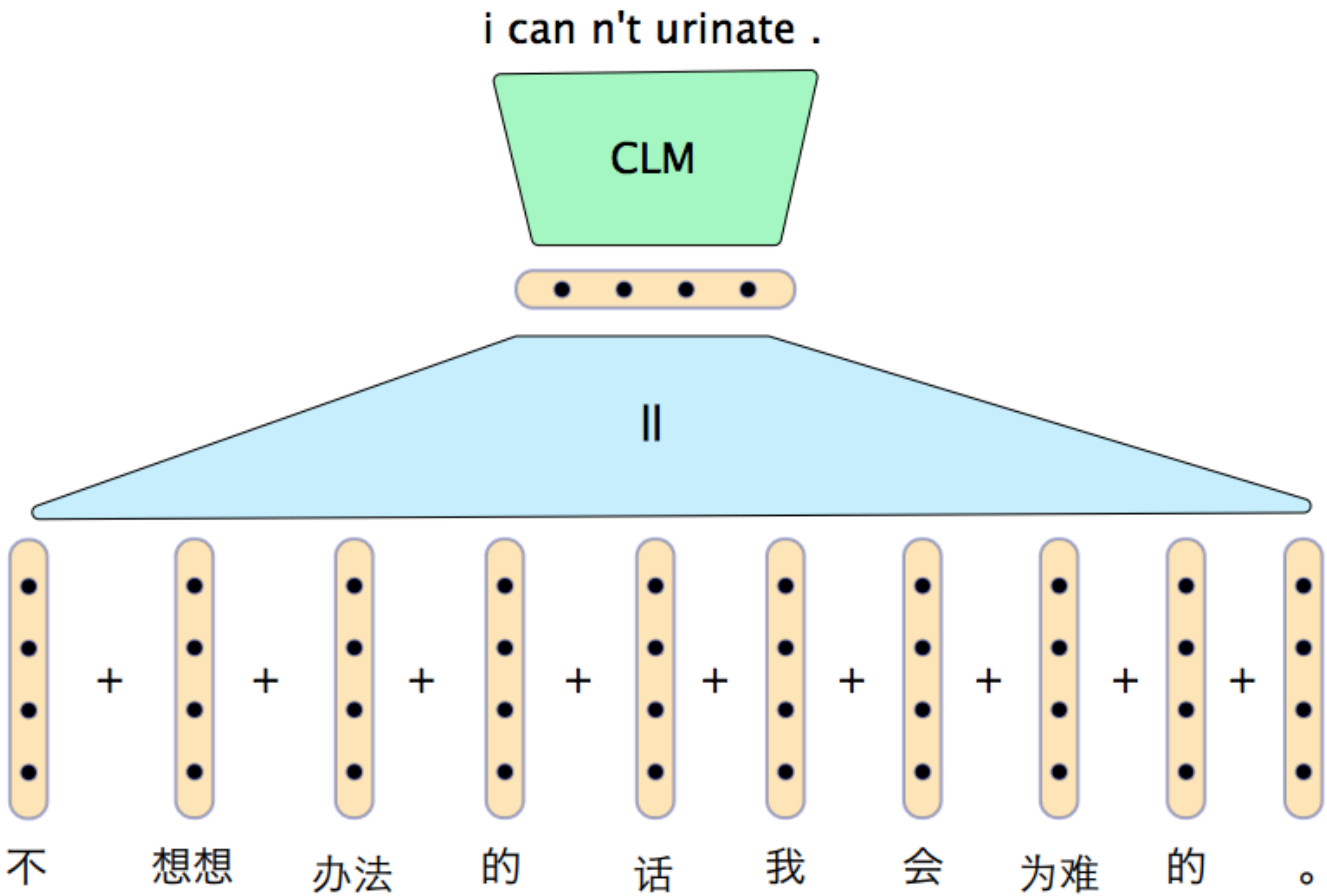


How well does this model do?



(Literal: I will feel bad if you do not find a solution.)

How well does this model do?



(Literal: I will feel bad if you do not find a solution.)

Summary

- Conditional language modeling provides a convenient formulation for a lot of practical applications
- Two big problems:
  - Model expressivity
  - Decoding difficulties
- Next time
  - A better encoder for vector to sequence models
  - “Attention” for better learning
  - Lots of results on machine translation

Questions?