Neural Machine Translation

Graham Neubig

Carnegie Mellon University

Language Technologies Institute

To Learn More:

"Neural Machine Translation and Sequence-to-sequence Models: A Tutorial"

Machine Translation

- Translation from one language to another

I'm giving a talk at University of Pennsylvania

↓

ペンシルベニア大学で講演をしています。

Review: Recurrent Neural Networks

Long-distance Dependencies in Language

- Agreement in number, gender, etc.

He does not have very much confidence in **himself**.
She does not have very much confidence in **herself**.

- Selectional preference

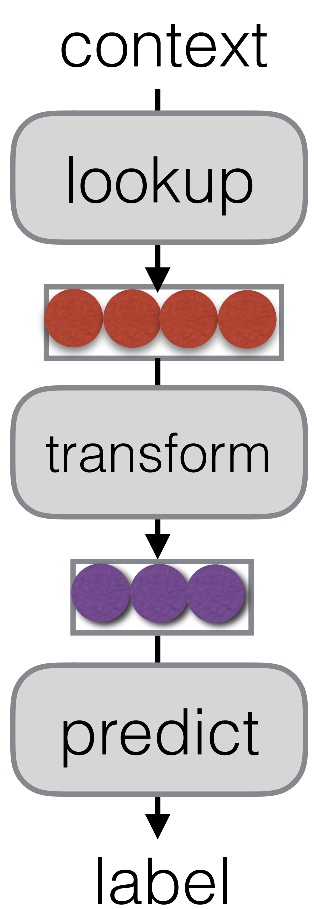
The **reign** has lasted as long as the life of the **queen**.
The **rain** has lasted as long as the life of the **clouds**.

Recurrent Neural Networks

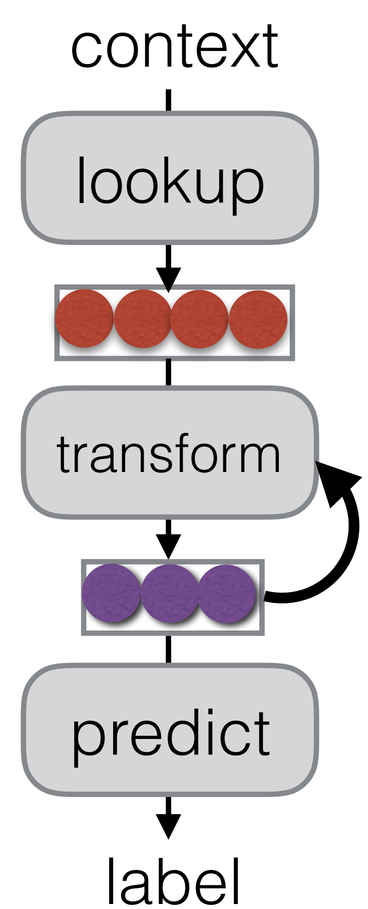
(Elman 1990)

- Tools to “remember” information

Feed-forward NN

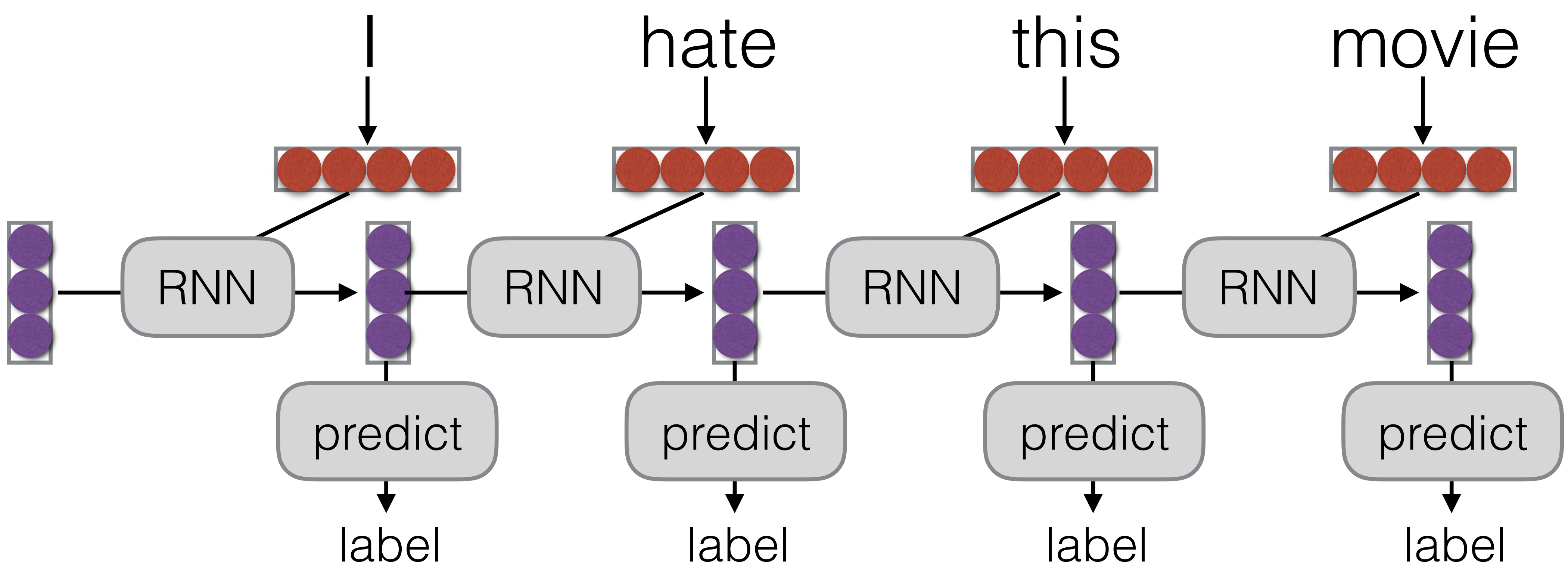


Recurrent NN

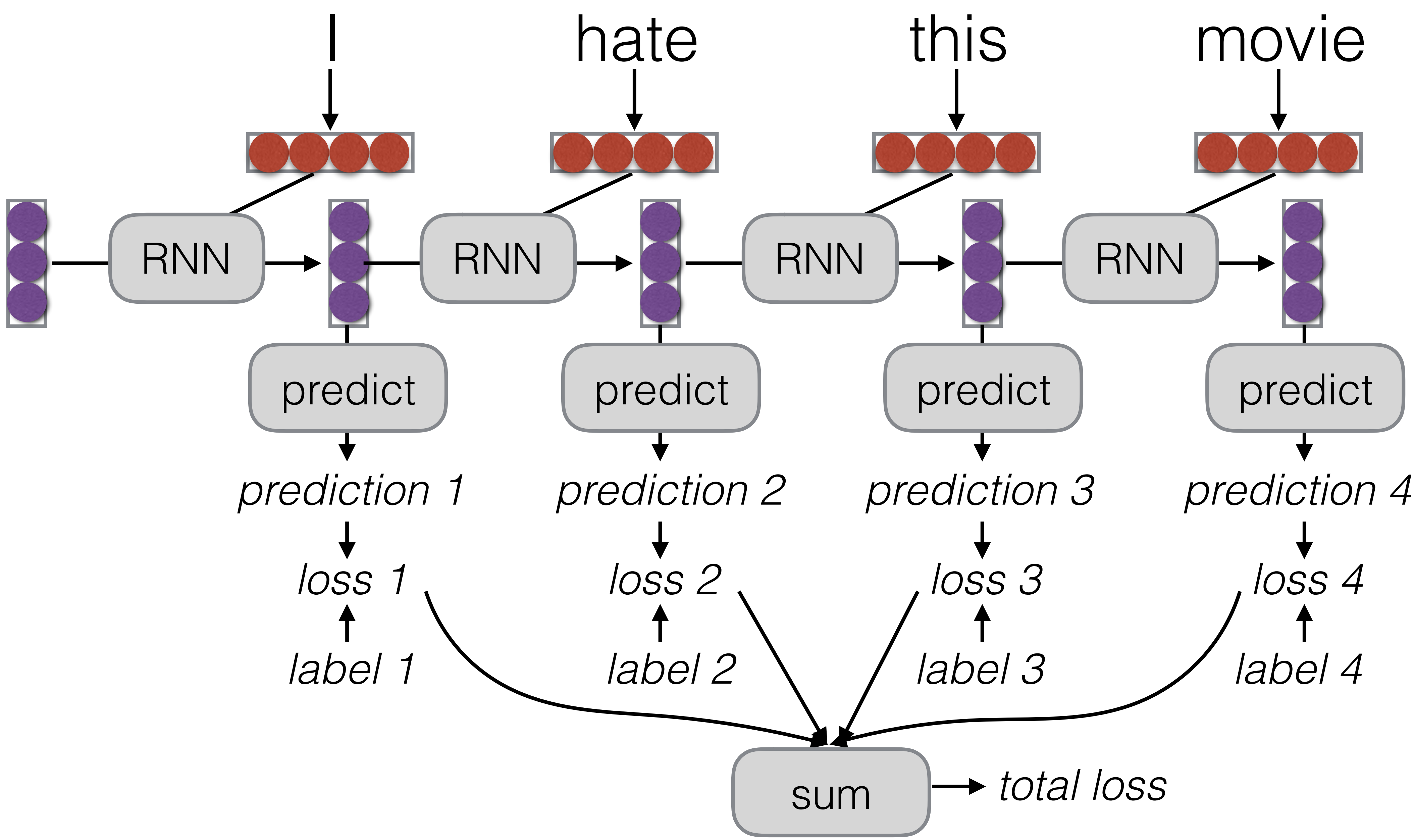


Unrolling in Time

- What does processing a sequence look like?

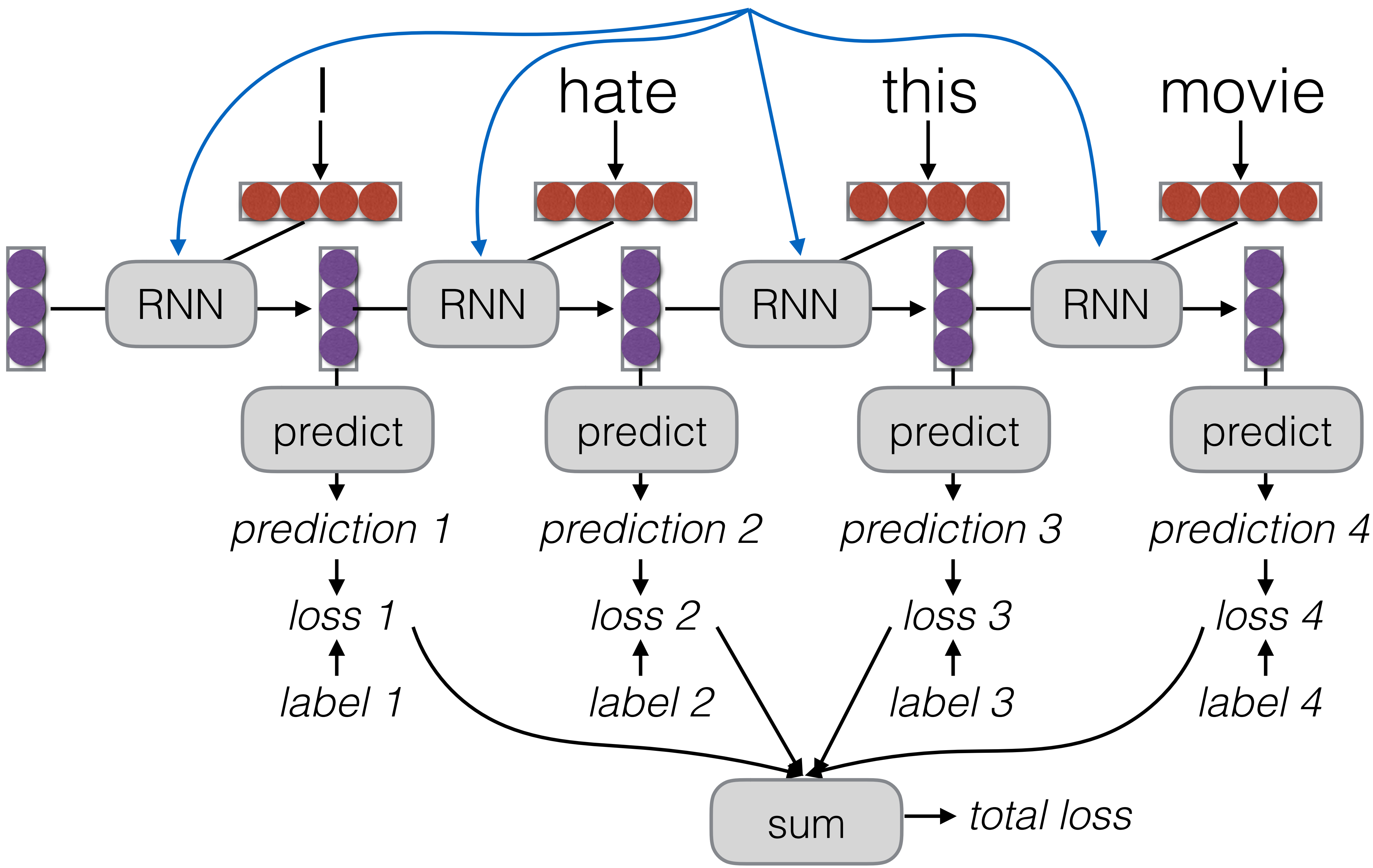


Training RNNs



Parameter Tying

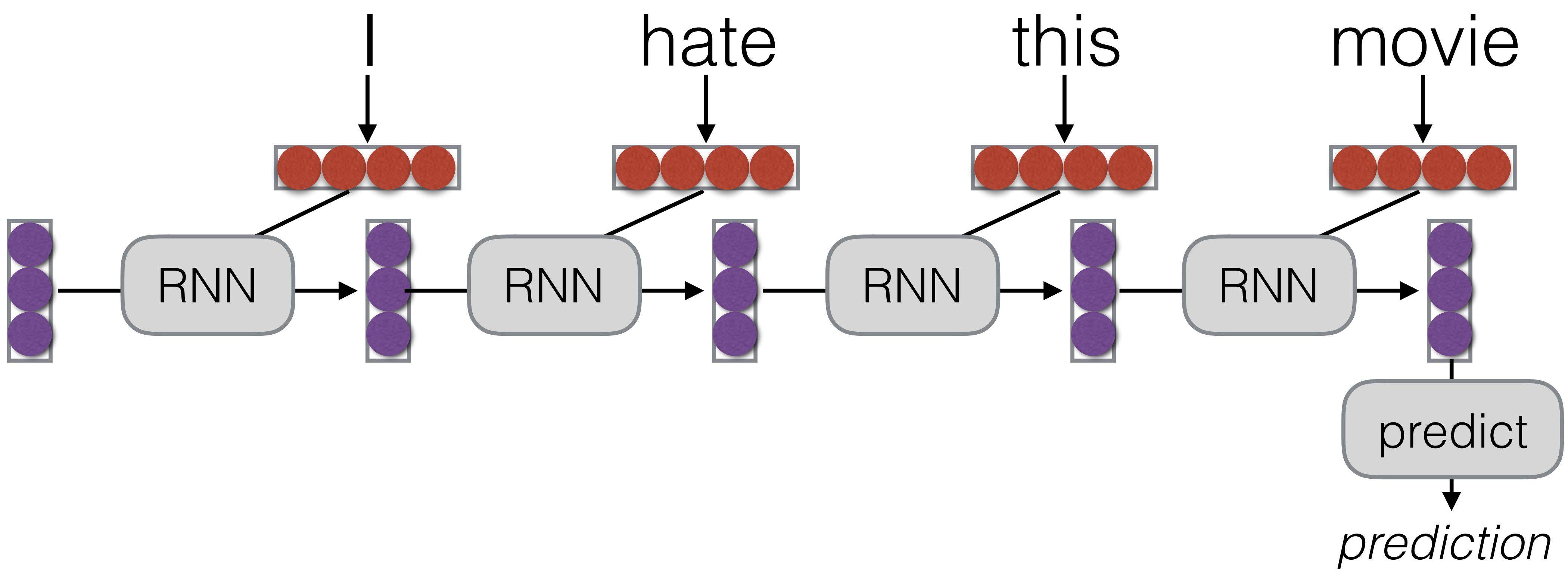
Parameters are shared! Derivatives are accumulated.



What Can RNNs Do?

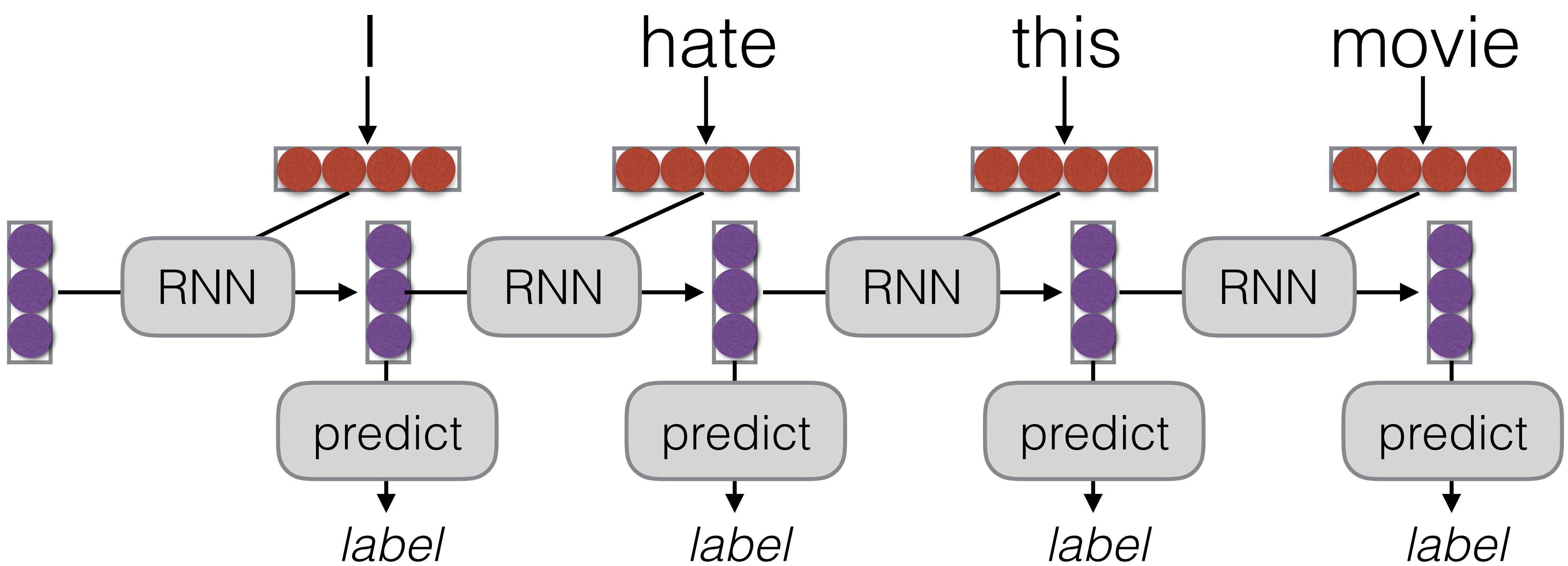
- Represent a sentence
  - Read whole sentence, make a prediction
- Represent a context within a sentence
  - Read context up until that point

Representing Sentences



- Sentence classification
- Conditioned generation
- Retrieval

Representing Contexts



- Tagging
- Language Modeling
- Calculating Representations for Parsing, etc.

e.g. Language Models

- Language models are generative models of text

$$s \sim P(x)$$

↓

"The Malfoys!" said Hermione.

Harry was watching him. He looked like Madame Maxime. When she strode up the wrong staircase to visit himself.

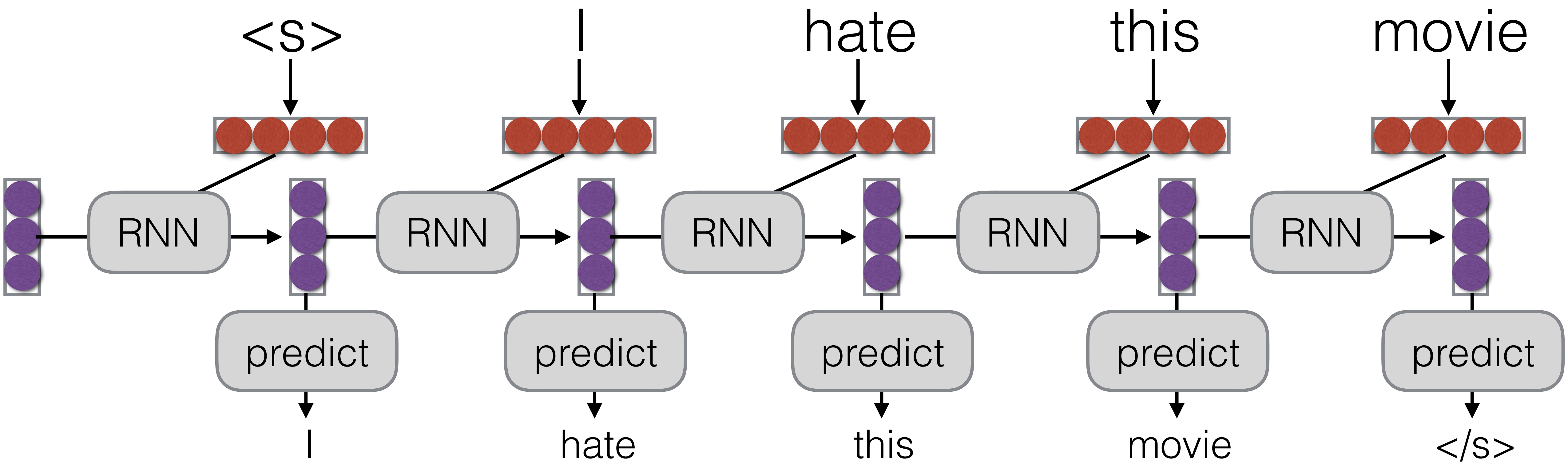
"I'm afraid I've definitely been suspended from power, no chance—indeed?" said Snape. He put his head back behind them and read groups as they crossed a corner and fluttered down onto their ink lamp, and picked up his spoon. The doorbell rang. It was a lot cleaner down in London.

Text Credit: Max Deutsch (https://medium.com/deep-writing/)

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

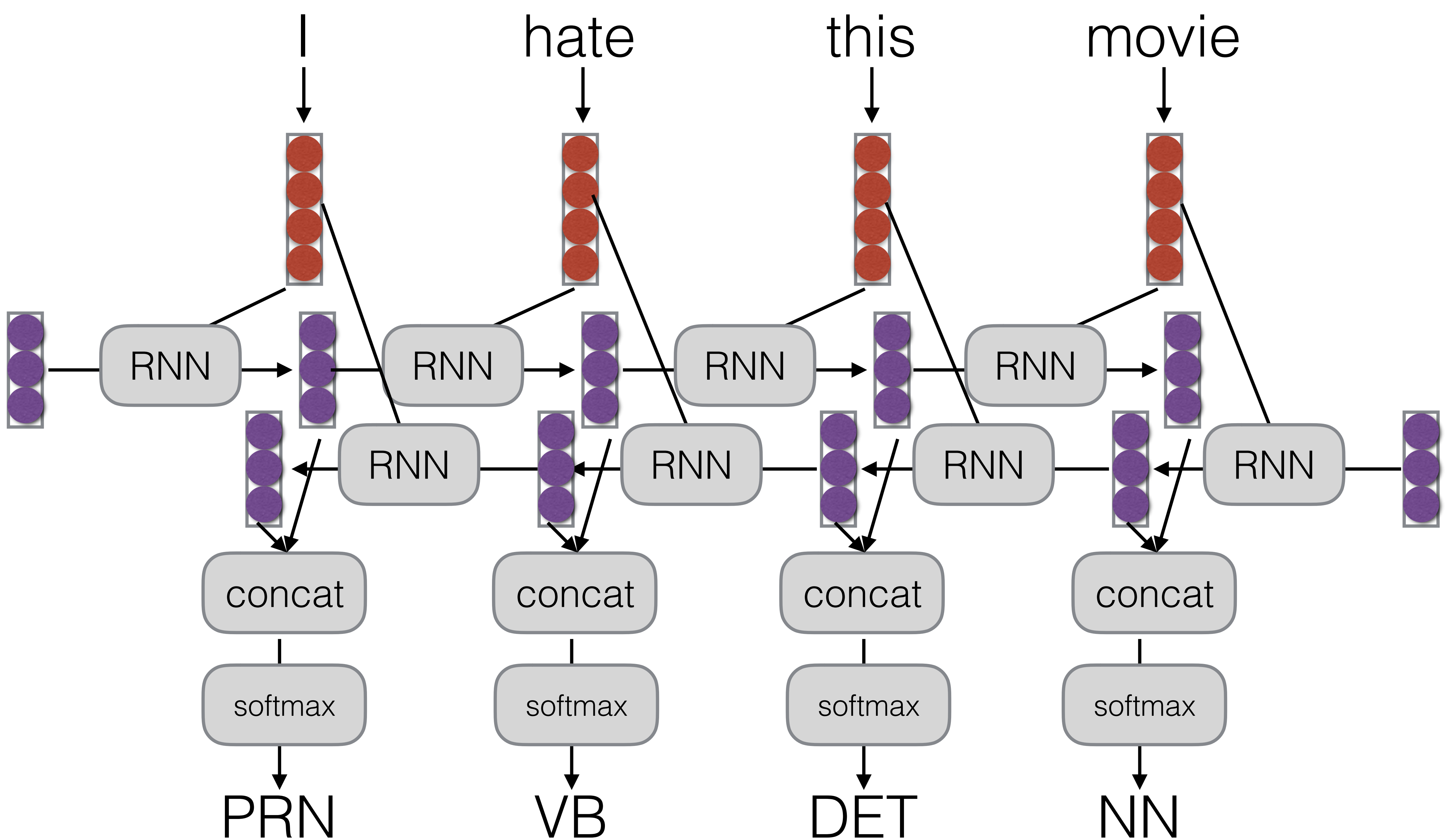
Language Modeling w/ RNNs



- At each step, calculate probability of next word

Bi-RNNs

- A simple extension, run the RNN in both directions



Conditional Language Modeling / Machine Translation

Conditioned Language Models

- Not just generate text, generate text according to some specification

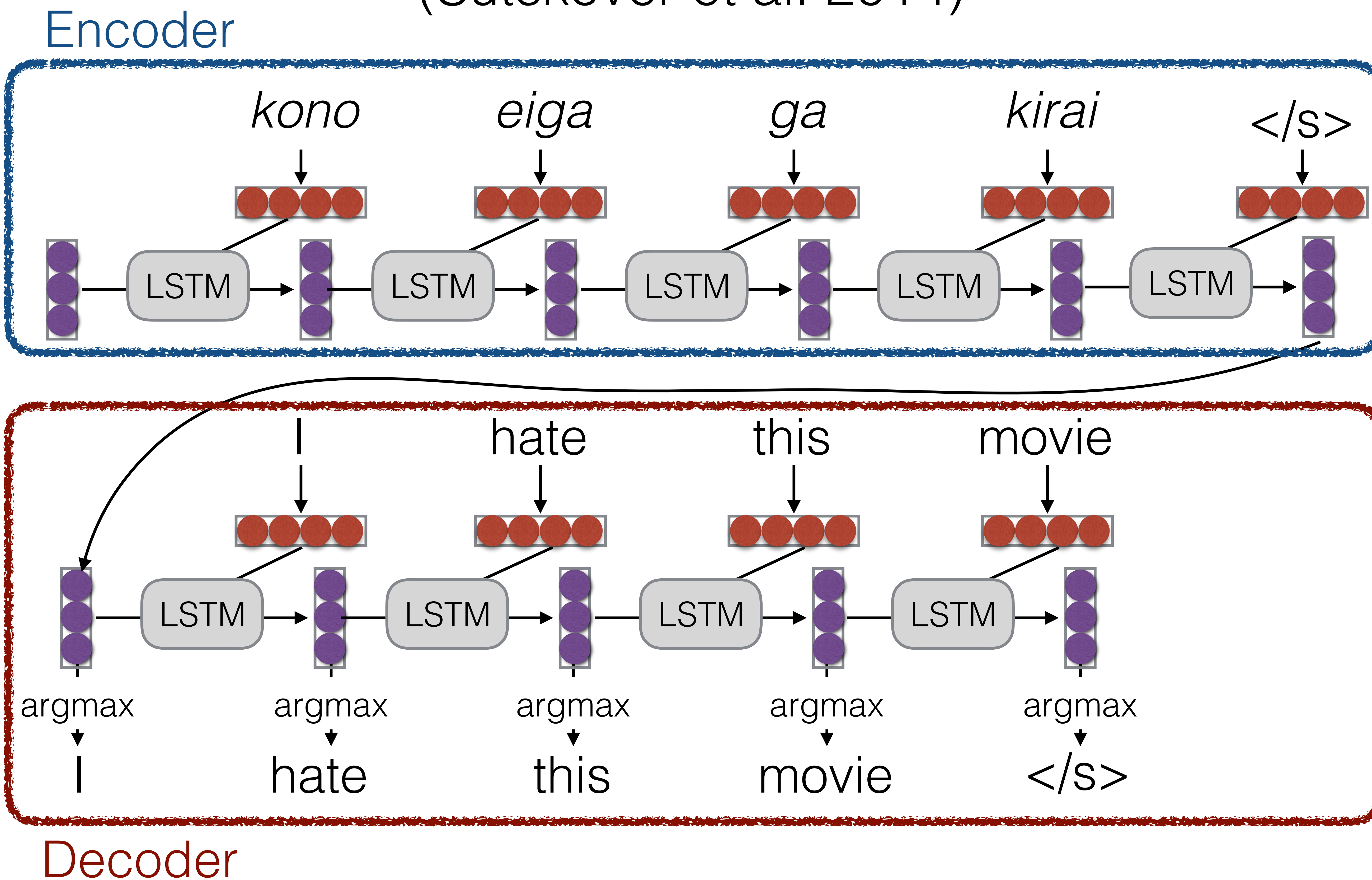
Input *X*	Output *Y* (**Text**)	Task
Structured Data	NL Description	NL Generation
English	Japanese	Translation
Document	Short Description	Summarization
Utterance	Response	Response Generation
Image	Text	Image Captioning
Speech	Transcript	Speech Recognition

Conditional Language Models

$$P(Y|X) = \prod_{j=1}^{J} P(y_j \mid \underline{X}, y_1, \ldots, y_{j-1})$$

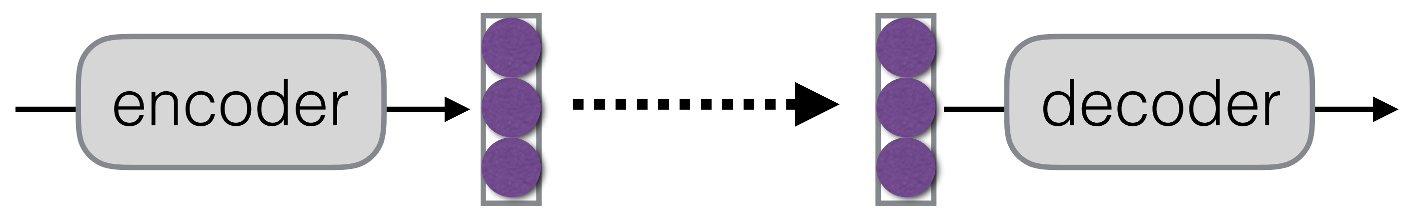
Added Context!

(One Type of) Conditional Language Model
(Sutskever et al. 2014)



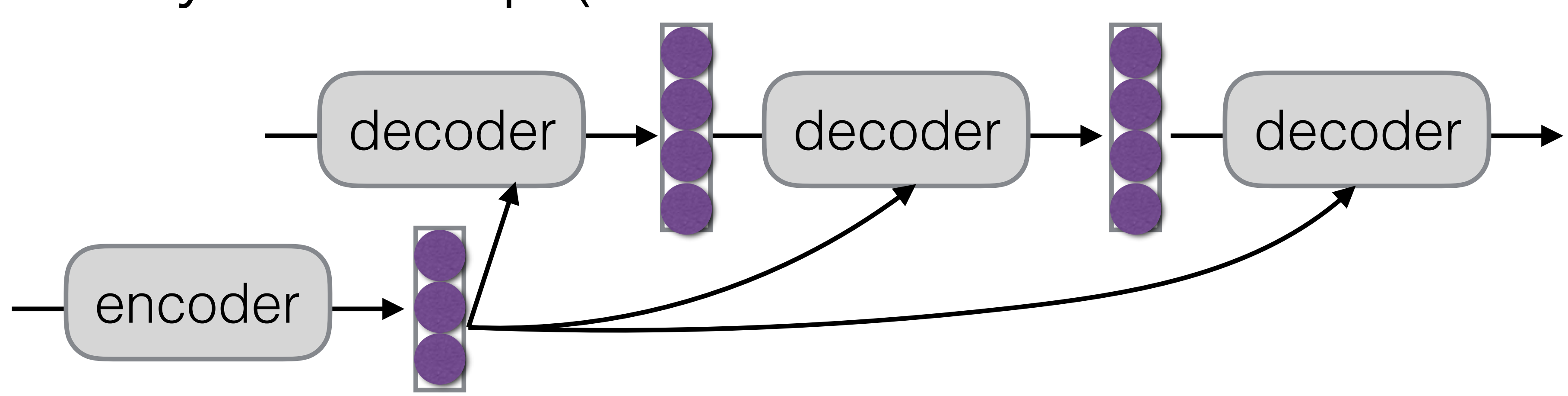
How to Pass Hidden State?

- Initialize decoder w/ encoder (Sutskever et al. 2014)



- Transform (can be different dimensions)

- Input at every time step (Kalchbrenner & Blunsom 2013)



Training Conditional LMs

- Get parallel corpus of inputs and outputs
- Maximize likelihood
- Standard corpora for MT:
  - WMT Conference on Machine Translation runs an evaluation every year with large-scale (e.g.10M sentence) datasets
  - Smaller datasets, e.g. 200k sentence TED talks from IWSLT, can be more conducive to experimentation

The Generation Problem

- We have a model of P(Y|X), how do we use it to generate a sentence?
- Two methods:
  - **Sampling:** Try to generate a *random* sentence according to the probability distribution.
  - **Argmax:** Try to generate the sentence with the *highest* probability.

Ancestral Sampling

- **Randomly generate** words one-by-one.

while y_{j-1} != “</s>”:
  $y_j \sim P(y_j \mid X, y_1, \ldots, y_{j-1})$

- An **exact method** for sampling from P(X), no further work needed.

Greedy Search

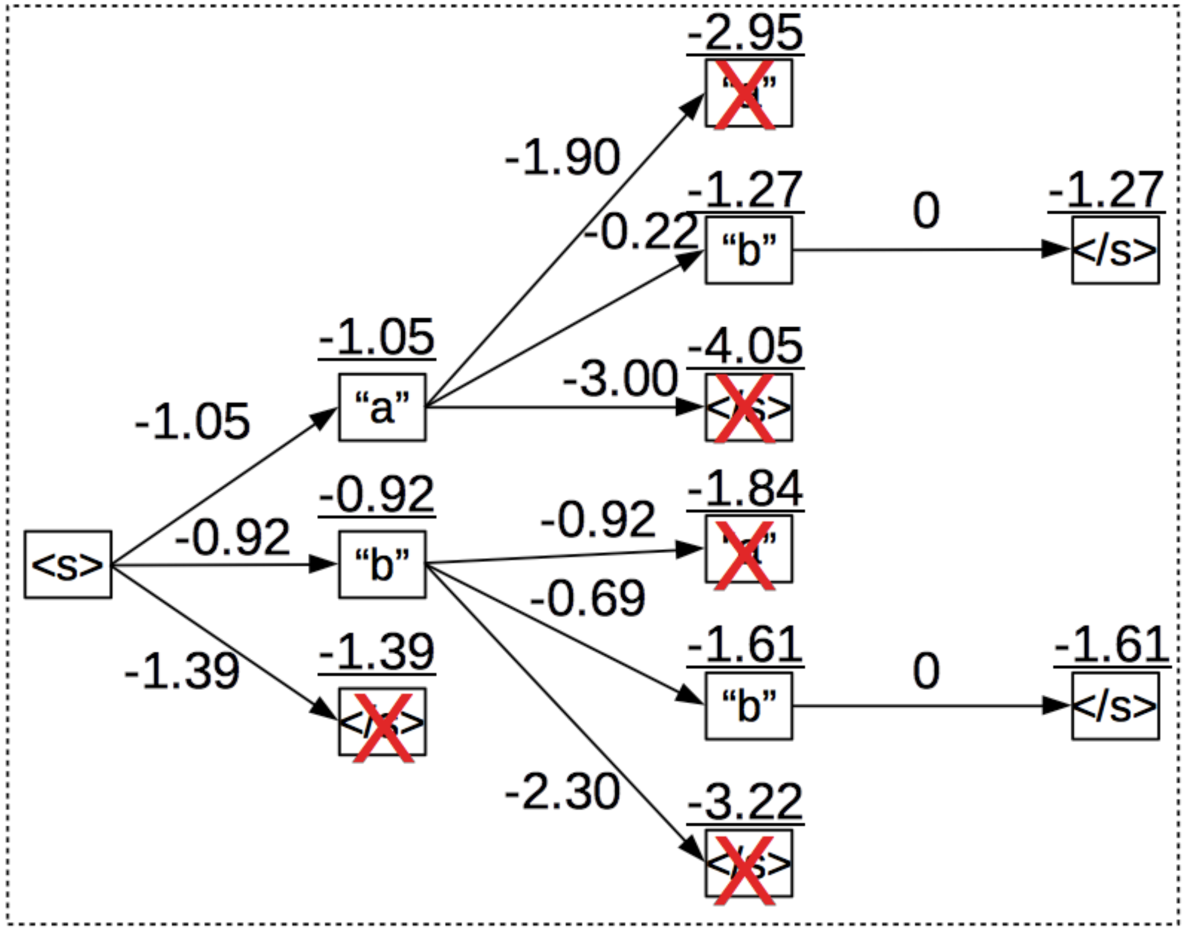
- One by one, pick the single highest-probability word

```
while y_{j-1} != "</s>":
  y_j = argmax P(y_j | X, y_1, …, y_{j-1})
```

- **Not exact, real problems:**
  - Will often generate the "easy" words first
  - Will prefer multiple common words to one rare word

Beam Search

- Instead of picking one high-probability word, maintain several paths



- Some in reading materials, more in a later class

How do we Evaluate?

Basic Evaluation Paradigm

- Use parallel test set
- Use system to generate translations
- Compare target translations w/ reference

Human Evaluation

- Ask a human to do evaluation

- Final goal, but slow, expensive, and sometimes inconsistent

BLEU

- Works by comparing n-gram overlap w/ reference

Reference: Taro visited Hanako

System: the Taro visited the Hanako

1-gram: 3/5
2-gram: 1/4

Brevity: min(1, |System|/|Reference|) = min(1, 5/3)

brevity penalty = 1.0

BLEU-2 $= (3/5 * 1/4)^{1/2} * 1.0$
$= 0.387$

- **Pros:** Easy to use, good for measuring system improvement
- **Cons:** Often doesn't match human eval, bad for comparing very different systems

Other Options

- **METEOR:** Considers synonyms
- **Translation Edit Rate (TER):** Considers number of edits to be turned into a good translation
- **chrF:** Considers score on the character level
- **RIBES:** Considers reordering
- etc. etc.

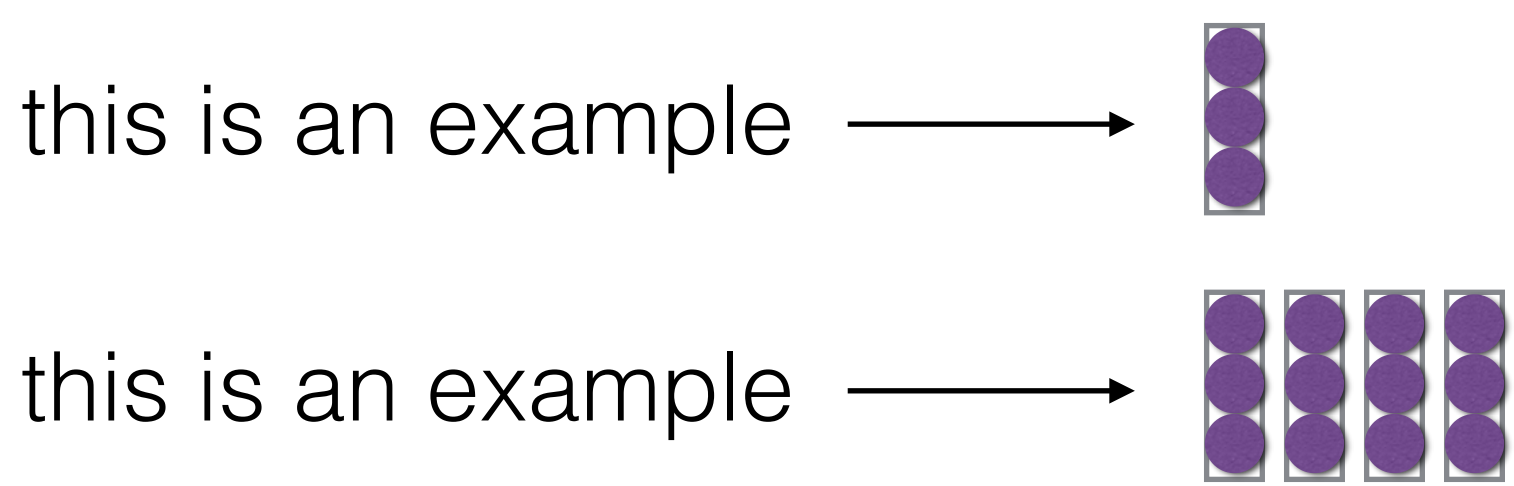
Attention

Sentence Representations

Problem!

> "You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!"
> — Ray Mooney

- But what if we could use multiple vectors, based on the length of the sentence.



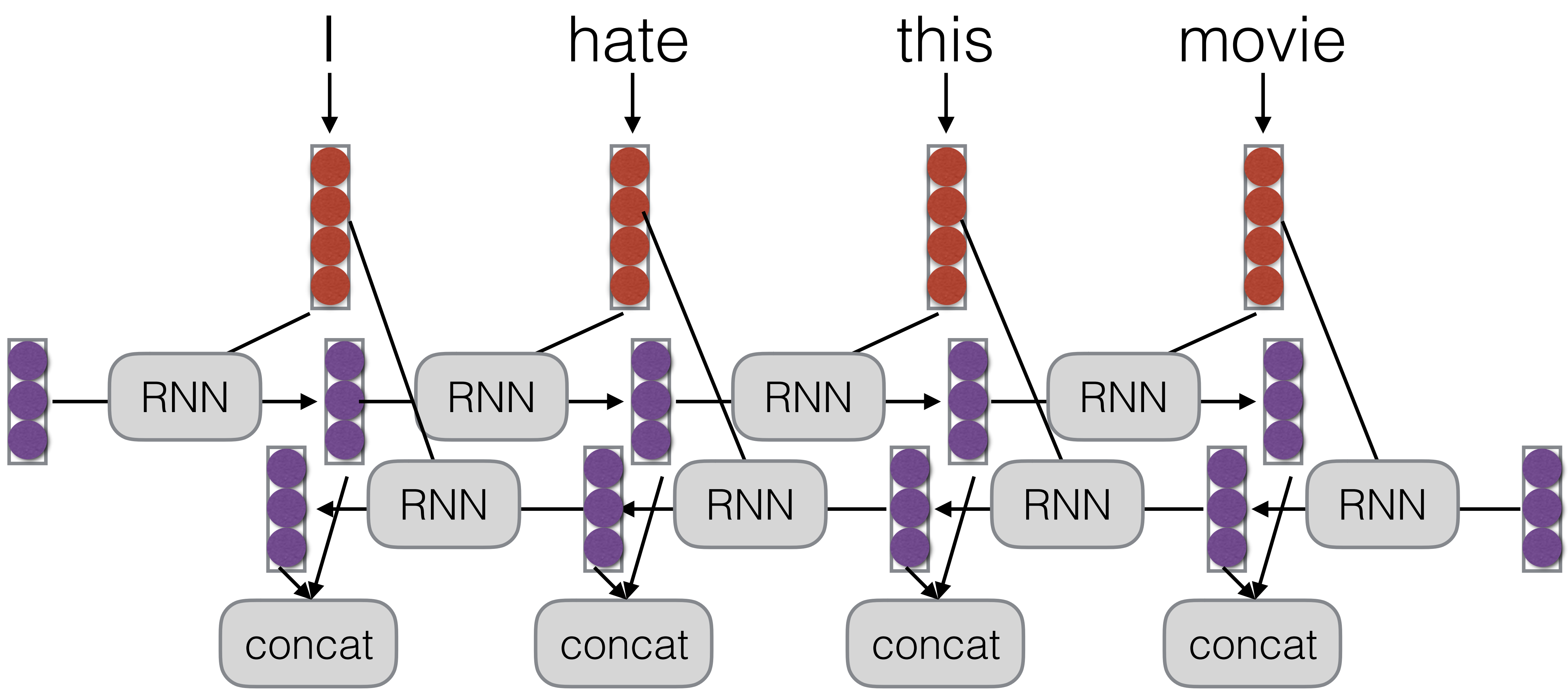
Basic Idea

(Bahdanau et al. 2015)

- Encode each word in the sentence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
- Use this combination in picking the next word

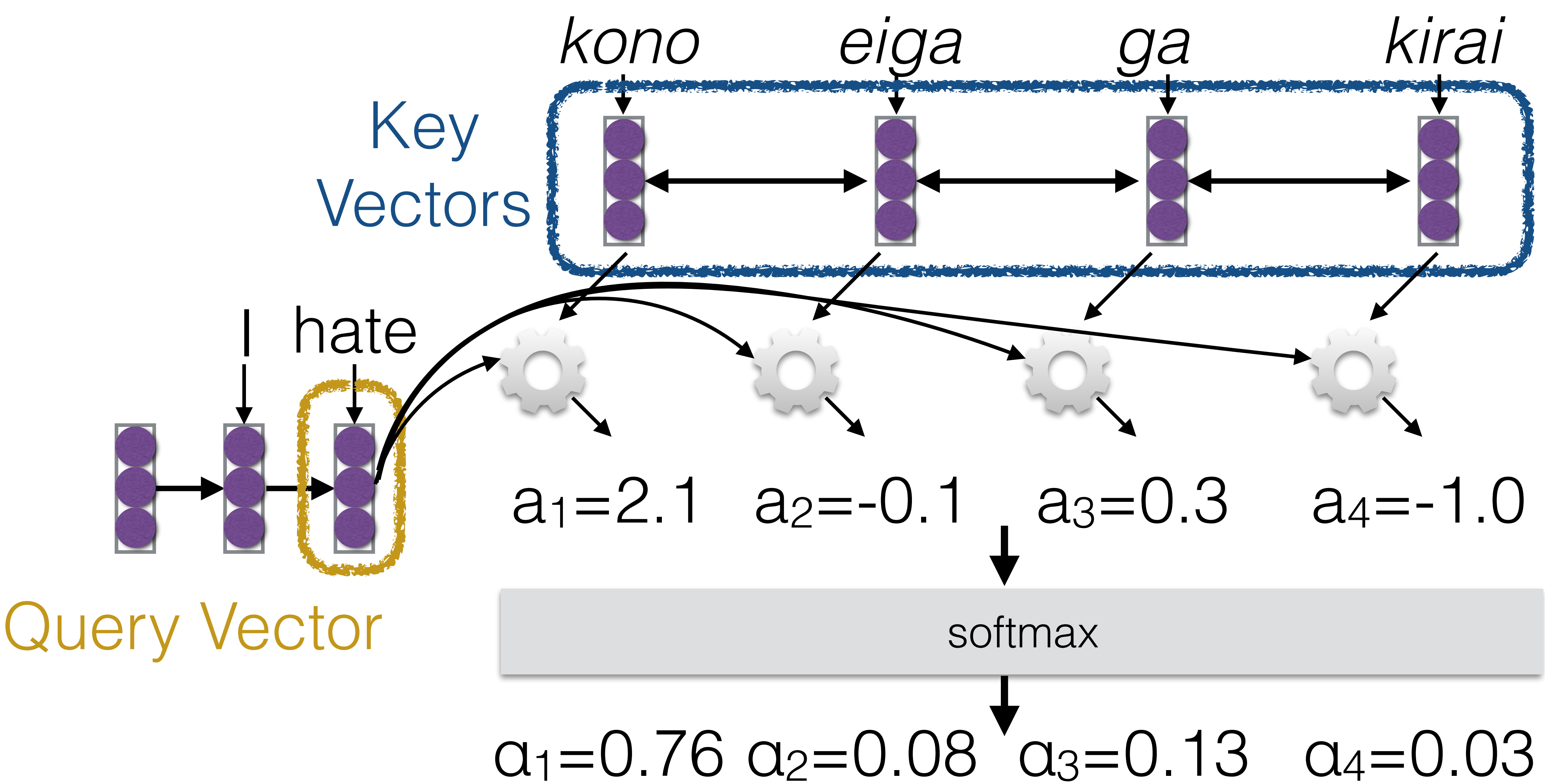
Encoder: Bi-RNNs

- A simple extension, run the RNN in both directions



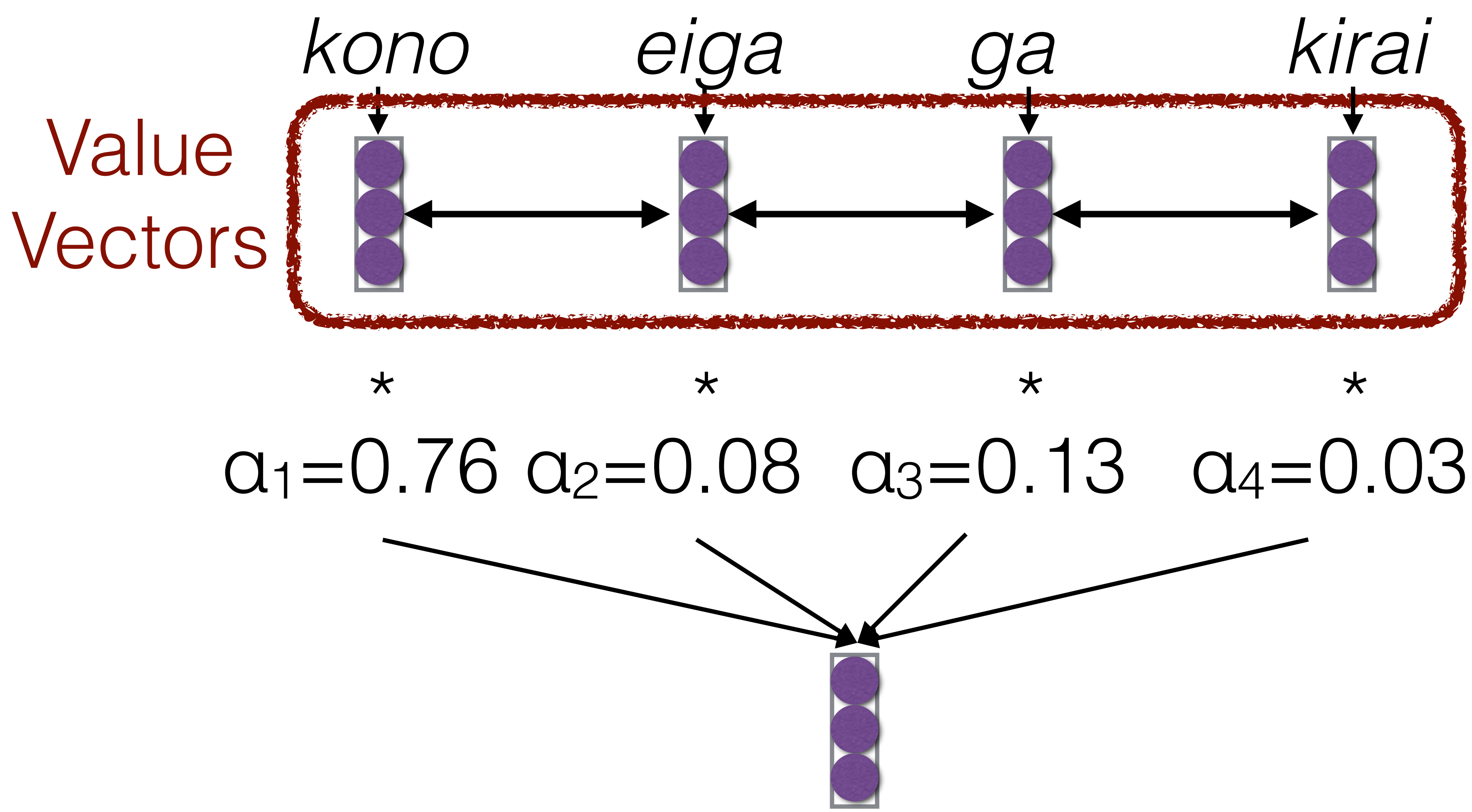
Calculating Attention (1)

- Use "query" vector (decoder state) and "key" vectors (all encoder states)
- For each query-key pair, calculate weight
- Normalize to add to one using softmax



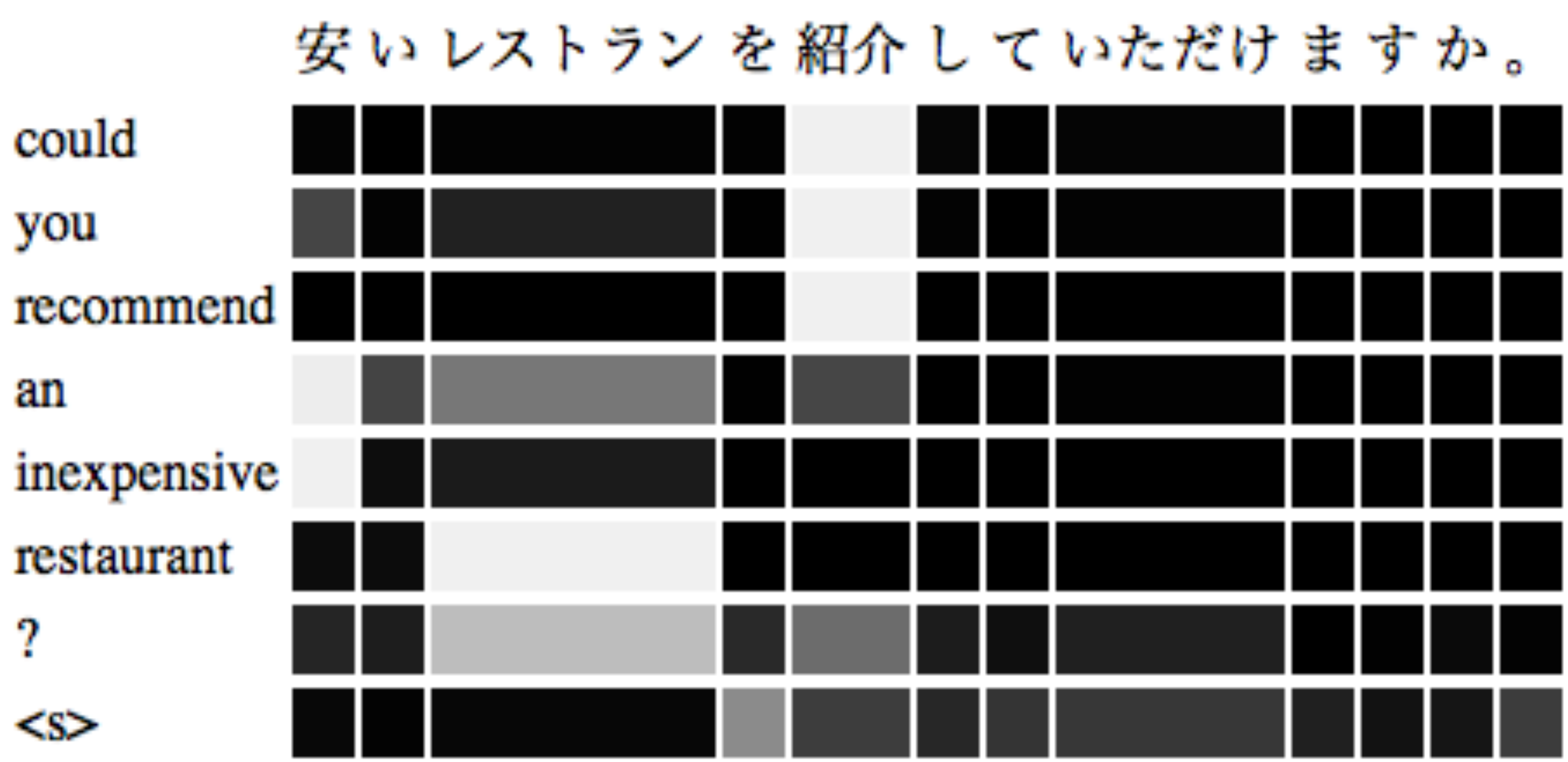
Calculating Attention (2)

- Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



- Use this in any part of the model you like

A Graphical Example



Attention Score Functions (1)

- ***q*** is the query and ***k*** is the key
- **Multi-layer Perceptron** (Bahdanau et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_2^{\intercal} \tanh(W_1[\boldsymbol{q}; \boldsymbol{k}])$$

  - Flexible, often very good with large data
- **Bilinear** (Luong et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^{\intercal} W \boldsymbol{k}$$

Attention Score Functions (2)

- **Dot Product** (Luong et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal \boldsymbol{k}$$

  - No parameters! But requires sizes to be the same.

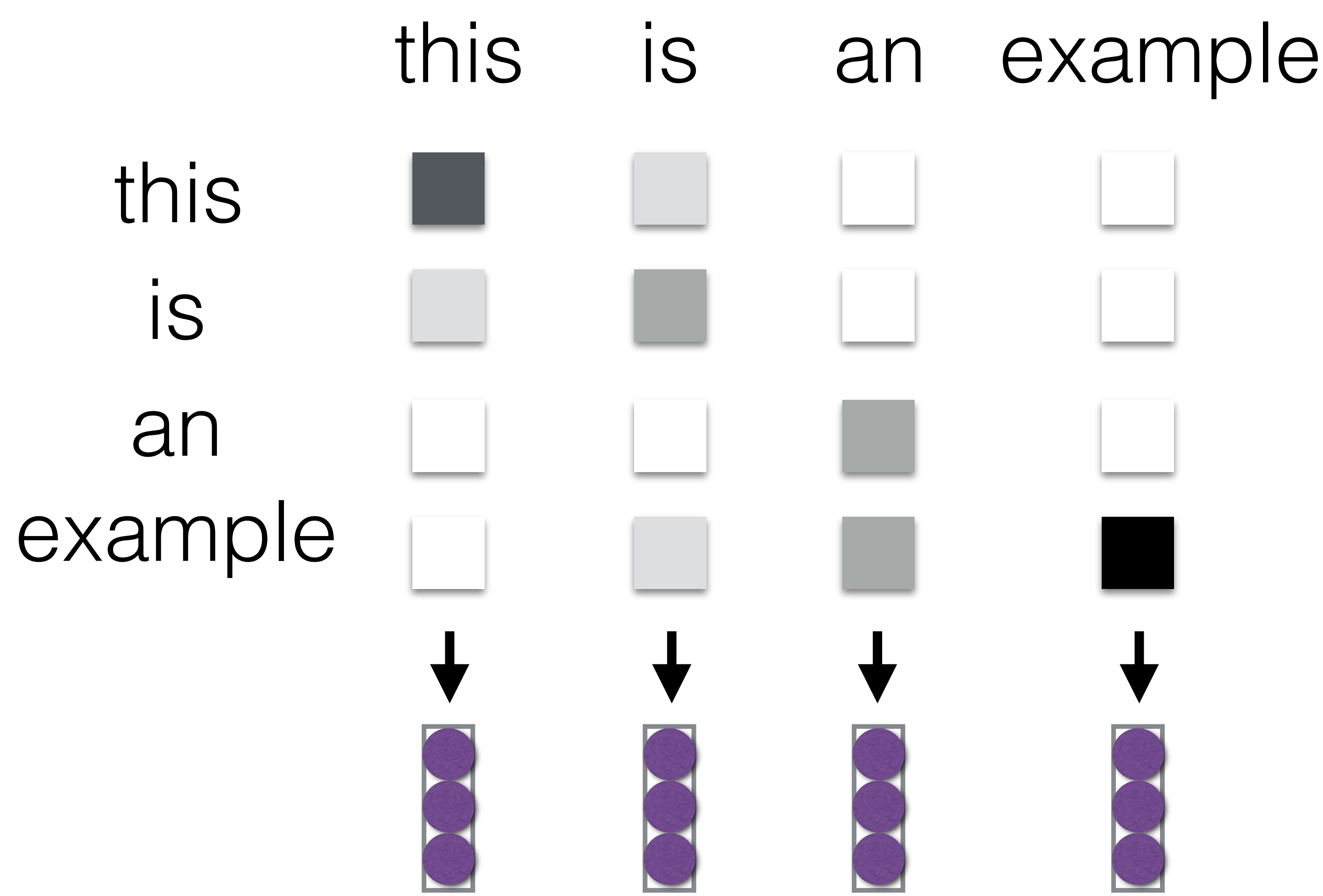
- **Scaled Dot Product** (Vaswani et al. 2017)
  - Problem: scale of dot product increases as dimensions get larger
  - Fix: scale by size of the vector

$$a(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\intercal \boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}}$$

Extensions to Attention

Intra-Attention / Self Attention
(Cheng et al. 2016)

- Each element in the sentence attends to other elements → context sensitive encodings!

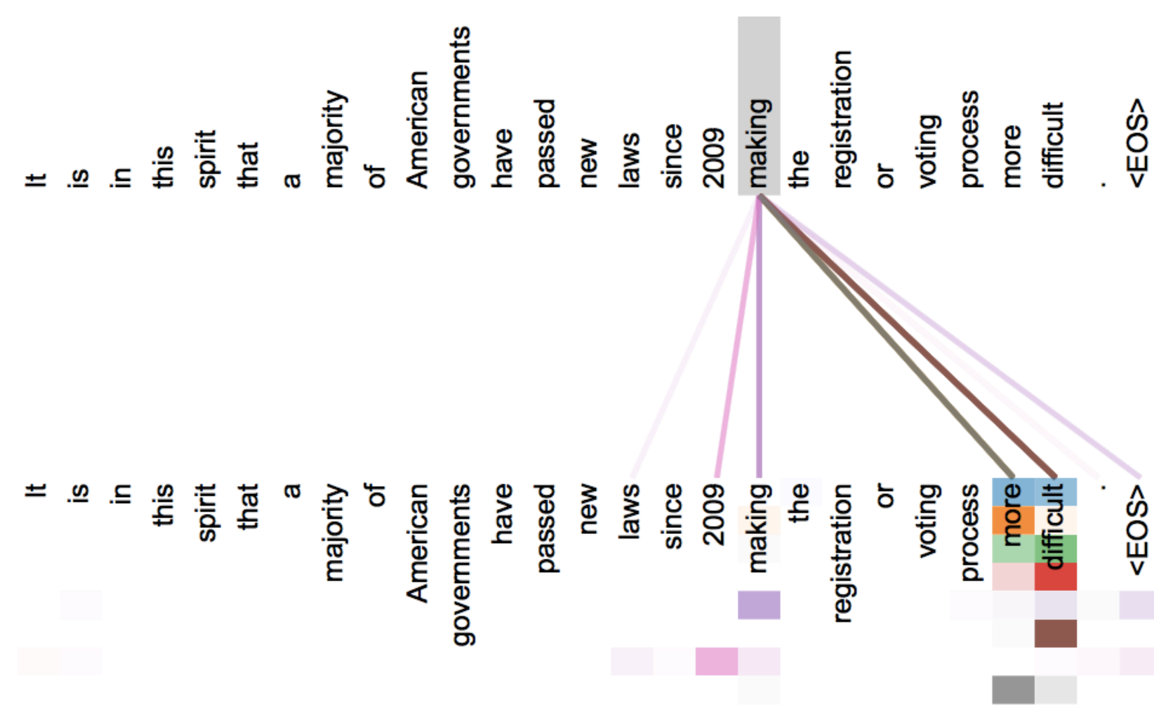


Multi-headed Attention

- **Idea:** multiple attention “heads” focus on different parts of the sentence
- e.g. Different heads for “copy” vs regular (Allamanis et al. 2016)

	Target		Attention Vectors	λ
m_1	set	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.012
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_2	use	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.974
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_3	browser	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.969
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_4	cache	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.583
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_5	END	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.066
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	

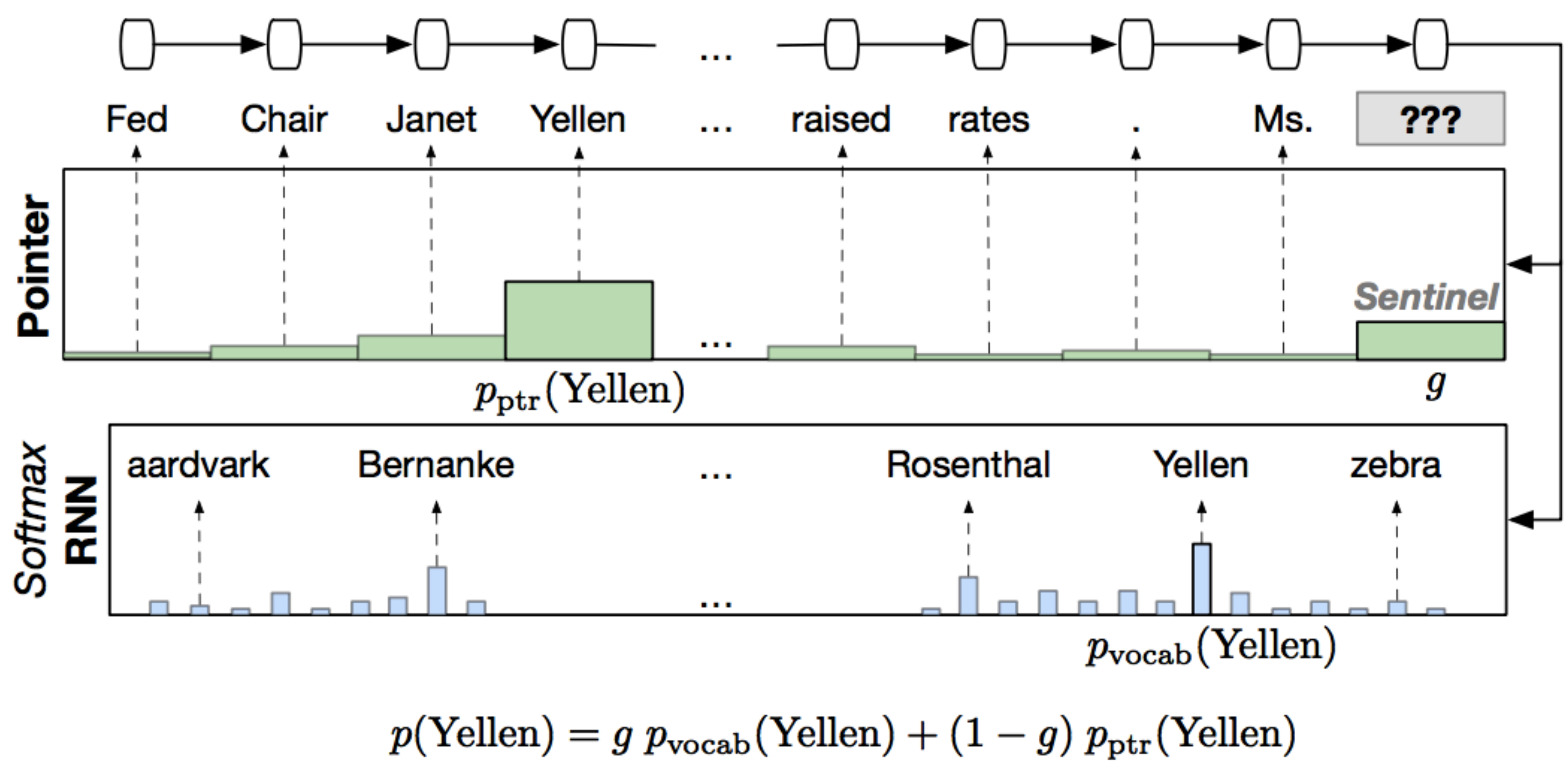
- Or multiple independently learned heads (Vaswani et al. 2017)



- Or one head for every hidden node! (Choi et al. 2018)

Attending to Previously Generated Things

- In language modeling, attend to the previous words (Merity et al. 2016)



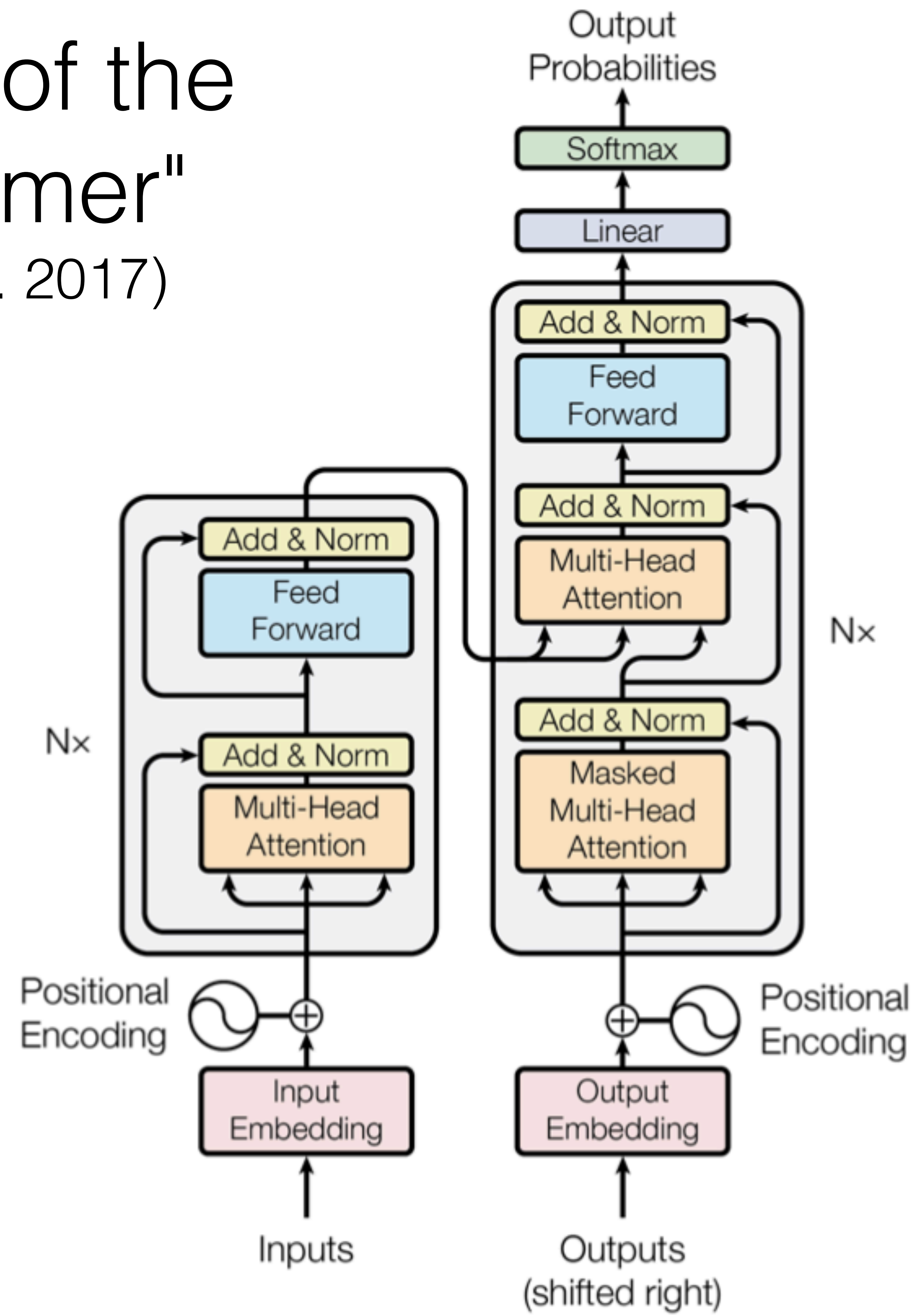
- In translation, attend to either input or previous output (Vaswani et al. 2017)

An Interesting Case Study: “Attention is All You Need”

(Vaswani et al. 2017)

Summary of the "Transformer" (Vaswani et al. 2017)

- A sequence-to-sequence model based entirely on attention
- Also have attention on the output side! Calculate probability of next word by attention over previous words.
- Fast: only matrix multiplications



Attention Tricks

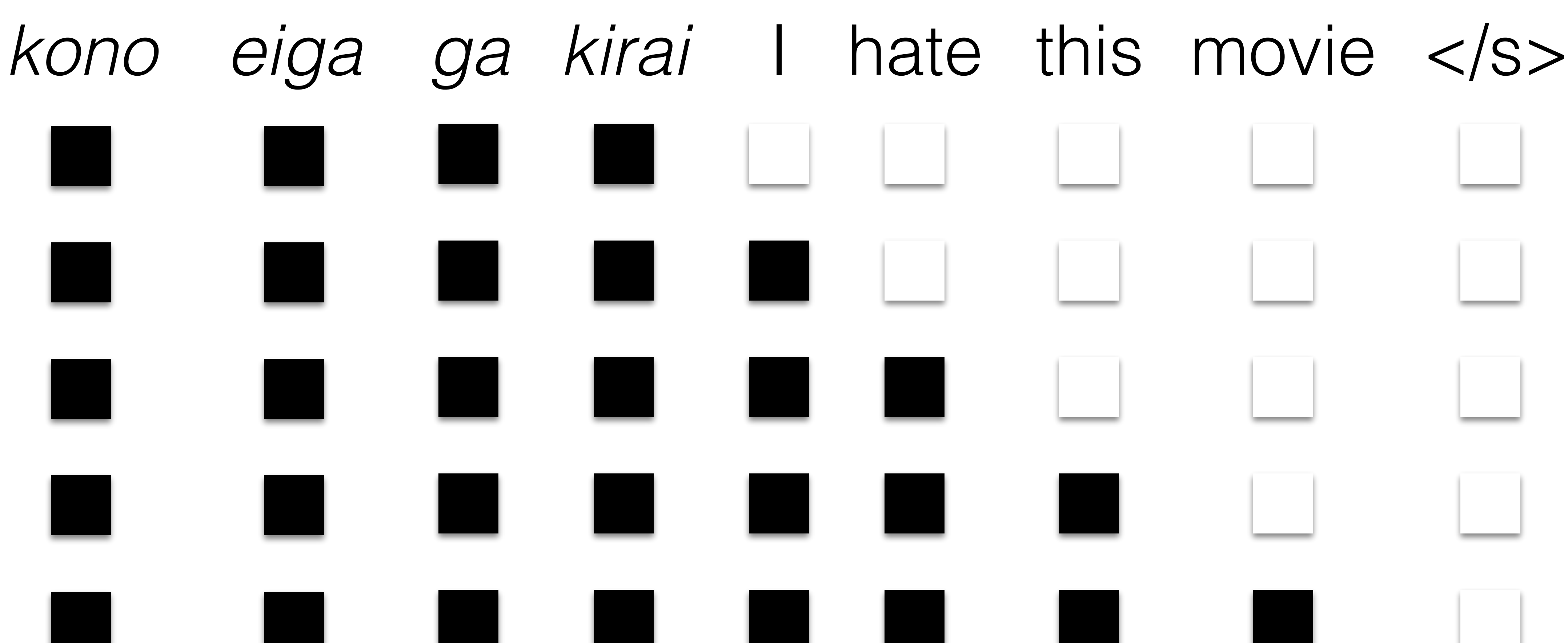
- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Normalized Dot-product Attention:** Remove bias in dot product when using large networks
- **Positional Encodings:** Make sure that even if we don’t have RNN, can still distinguish positions

Training Tricks

- **Layer Normalization:** Help ensure that layers remain in reasonable range
- **Specialized Training Schedule:** Adjust default learning rate of the Adam optimizer
- **Label Smoothing:** Insert some uncertainty in the training process
- **Masking for Efficient Training**

Masking for Training

- We want to perform training in as few operations as possible using big matrix multiplies
- We can do so by “masking” the results for the output



How to Get Started?

Getting Started

- Find training data, (e.g. TED talks from IWSLT), in your favorite language
- Download a toolkit (e.g. OpenNMT, fairseq, Sockeye, xnmt) and run it on the data
- Calculate the BLEU score and look at the results
- Think of what's going right, what's going wrong!

Questions?

To Learn More:
"Neural Machine Translation and Sequence-to-sequence Models: A Tutorial"