CIFAR ML/RL Summer School

Natural Language Understanding

Graham Neubig

Carnegie Mellon University

Language Technologies Institute

What Does it Mean to "Understand" Language?

- **Language modeling: P(text)**
  - "Does this sound like good English/French/…?"
- **Text classification: P(label | text)**
  - "Is this review a positive review?"
  - "What topic does this article belong to?"
- **Sequence transduction: P(text | text)**
  - "How do you say this in Japanese?"
  - "How do you respond to this comment?"
- **Language Analysis: P(labels/tree/graph | text)**
  - "Is this word a person, place or thing"?
  - "What is the syntactic structure of this sentence?"
  - "What is the latent meaning of this sentence?"
- **etc. etc.**

Language Modeling:
Models of P(text)

Are These Sentences OK?

- Jane went to the store.
- store to Jane went the.
- Jane went store.
- Jane goed to the store.
- The store went to Jane.
- The food truck went to Jane.

Engineering Solutions

- Jane went to the store.
- store to Jane went the.
- Jane went store.

} Create a grammar of the language

- Jane goed to the store.

} Consider morphology and exceptions

- The store went to Jane.

} Semantic categories, preferences

- The food truck went to Jane. } And their exceptions

Are These Sentences OK?

- ジェインは店へ行った。
- は店行ったジェインは。
- ジェインは店へ行た。
- 店はジェインへ行った。
- 屋台はジェインのところへ行った。

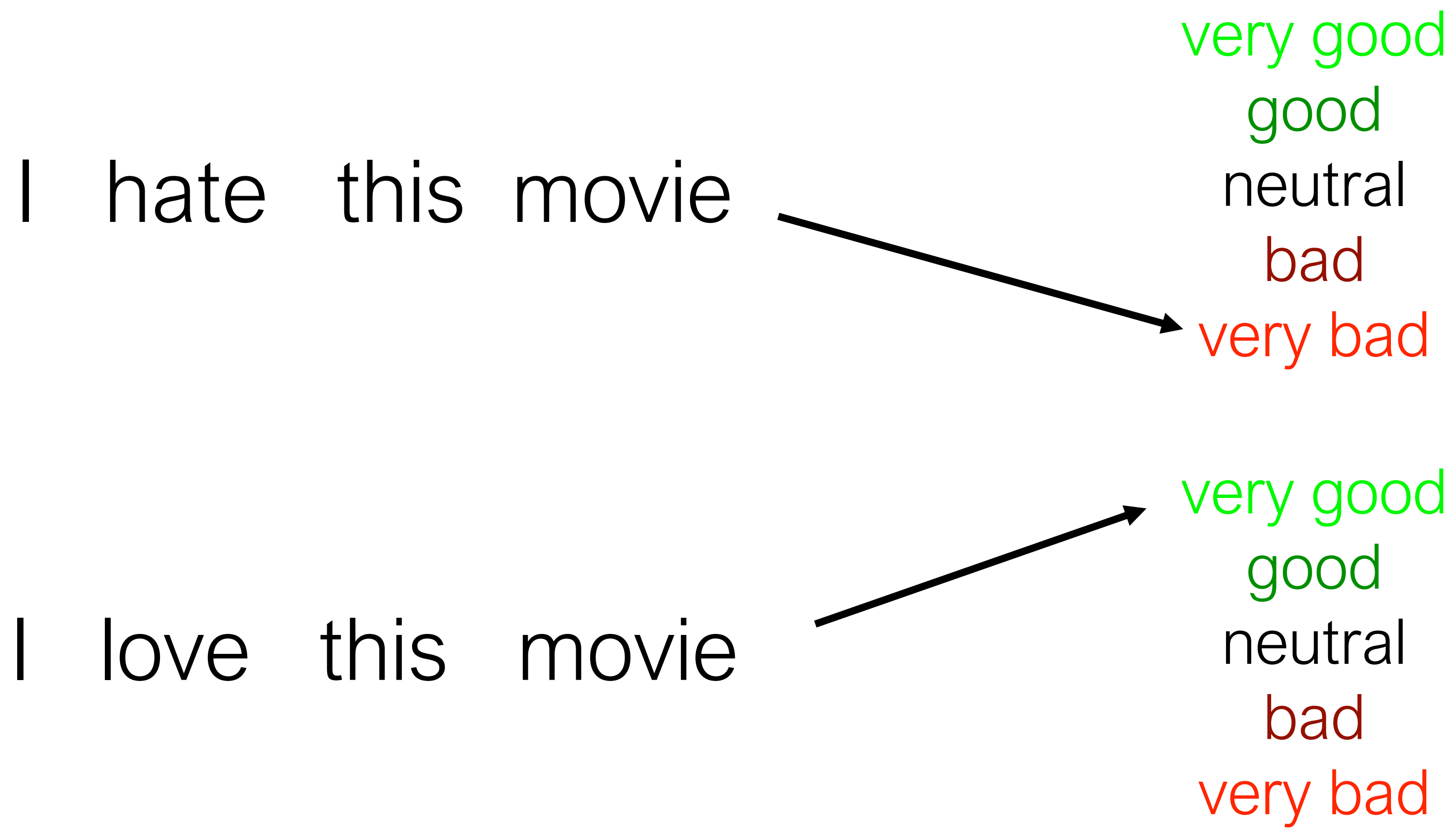
Phenomena to Handle

- Morphology
- Syntax
- Semantics/World Knowledge
- Discourse
- Pragmatics
- Multilinguality

Neural networks give us a flexible tool to handle these phenomena

A Slight Simplification:
Sentence Classification
Models of P(label | text)

An Example Prediction Problem: Sentence Classification



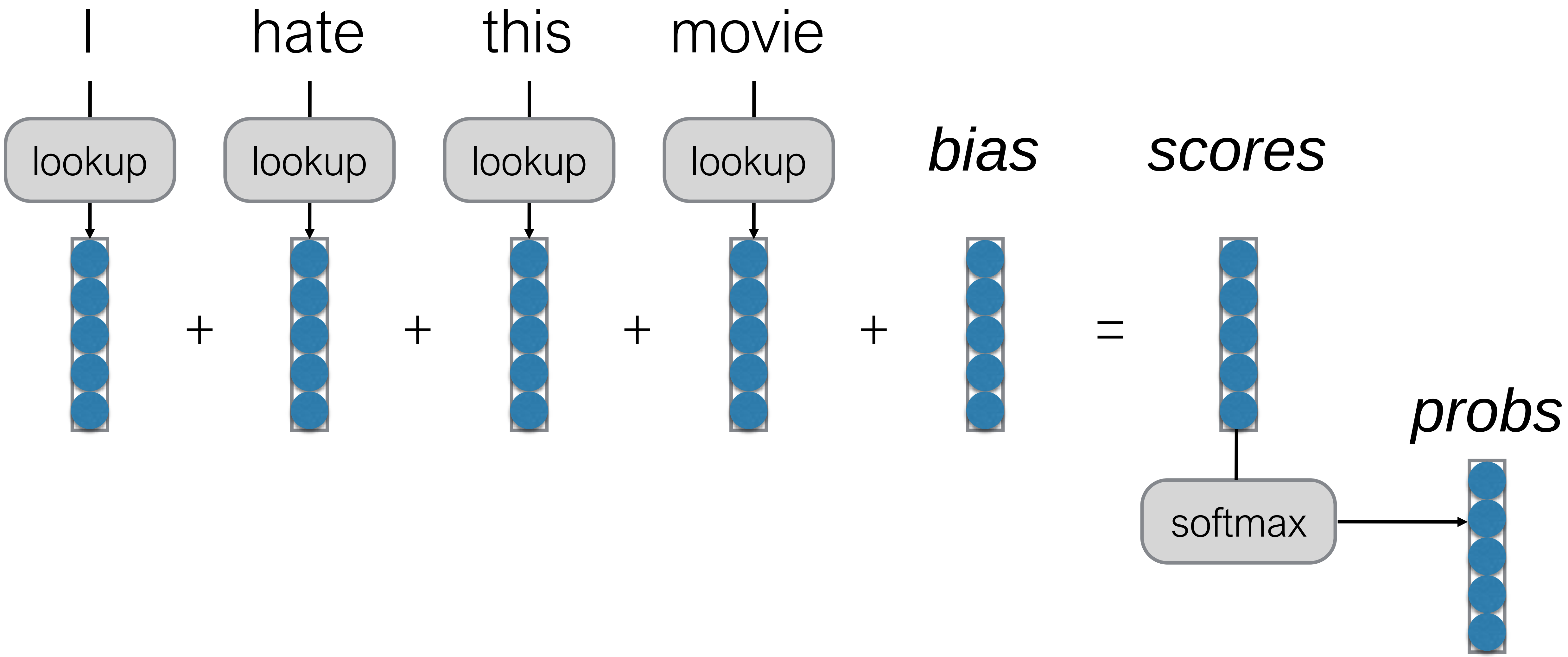
very good
good
neutral
bad
very bad

I hate this movie

very good
good
neutral
bad
very bad

I love this movie

A First Try:
Bag of Words (BOW)

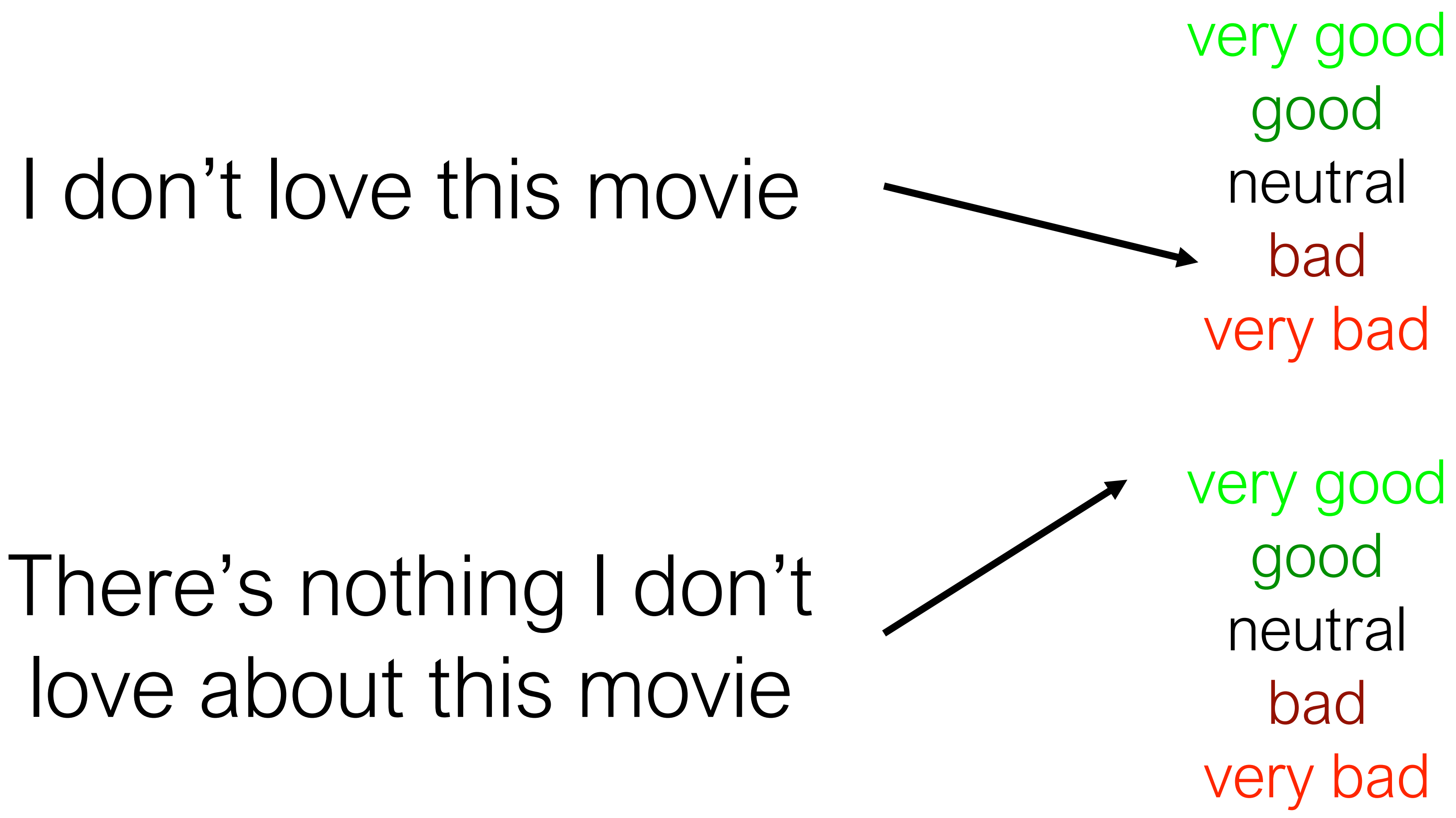


What do Our Vectors Represent?

- Each word has its own 5 elements corresponding to [very good, good, neutral, bad, very bad]

- “hate” will have a high value for “very bad”, etc.

Build It, Break It



I don’t love this movie

very good
good
neutral
bad
very bad

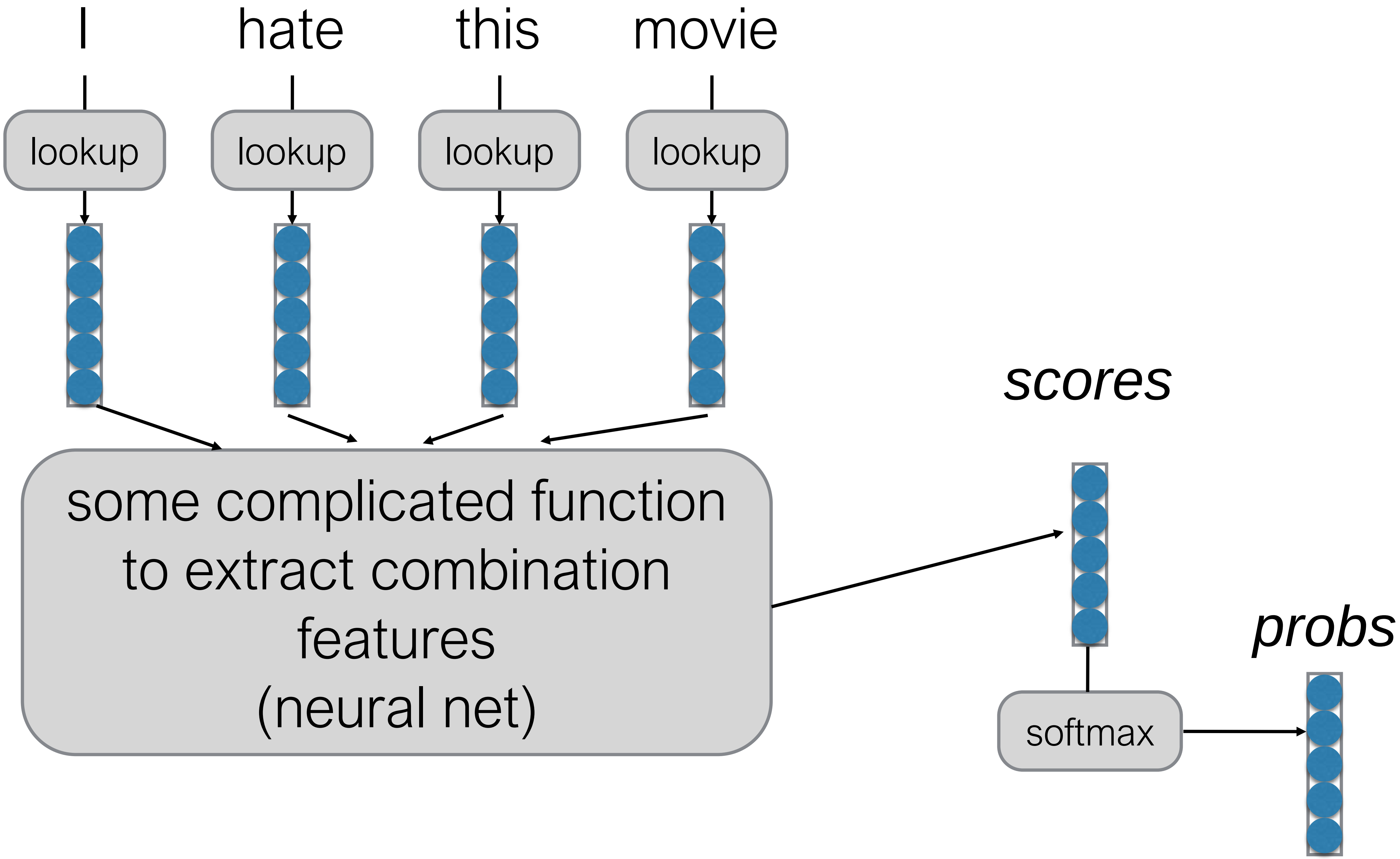
There’s nothing I don’t love about this movie

very good
good
neutral
bad
very bad

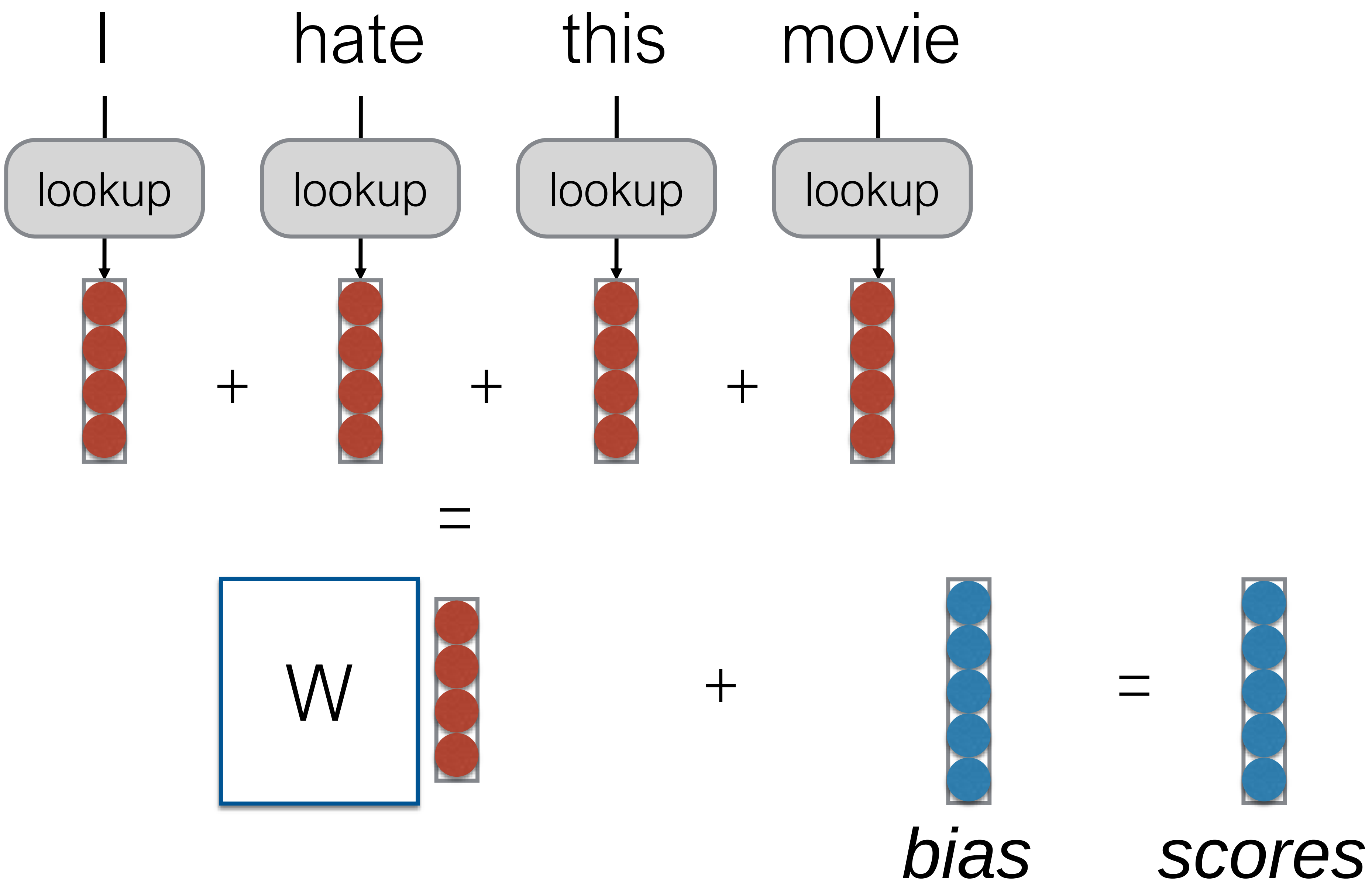
Combination Features

- Does it contain “don’t” and “love”?
- Does it contain “don’t”, “i”, “love”, and “nothing”?

Basic Idea of Neural Networks (for NLP Prediction Tasks)



Continuous Bag of Words (CBOW)



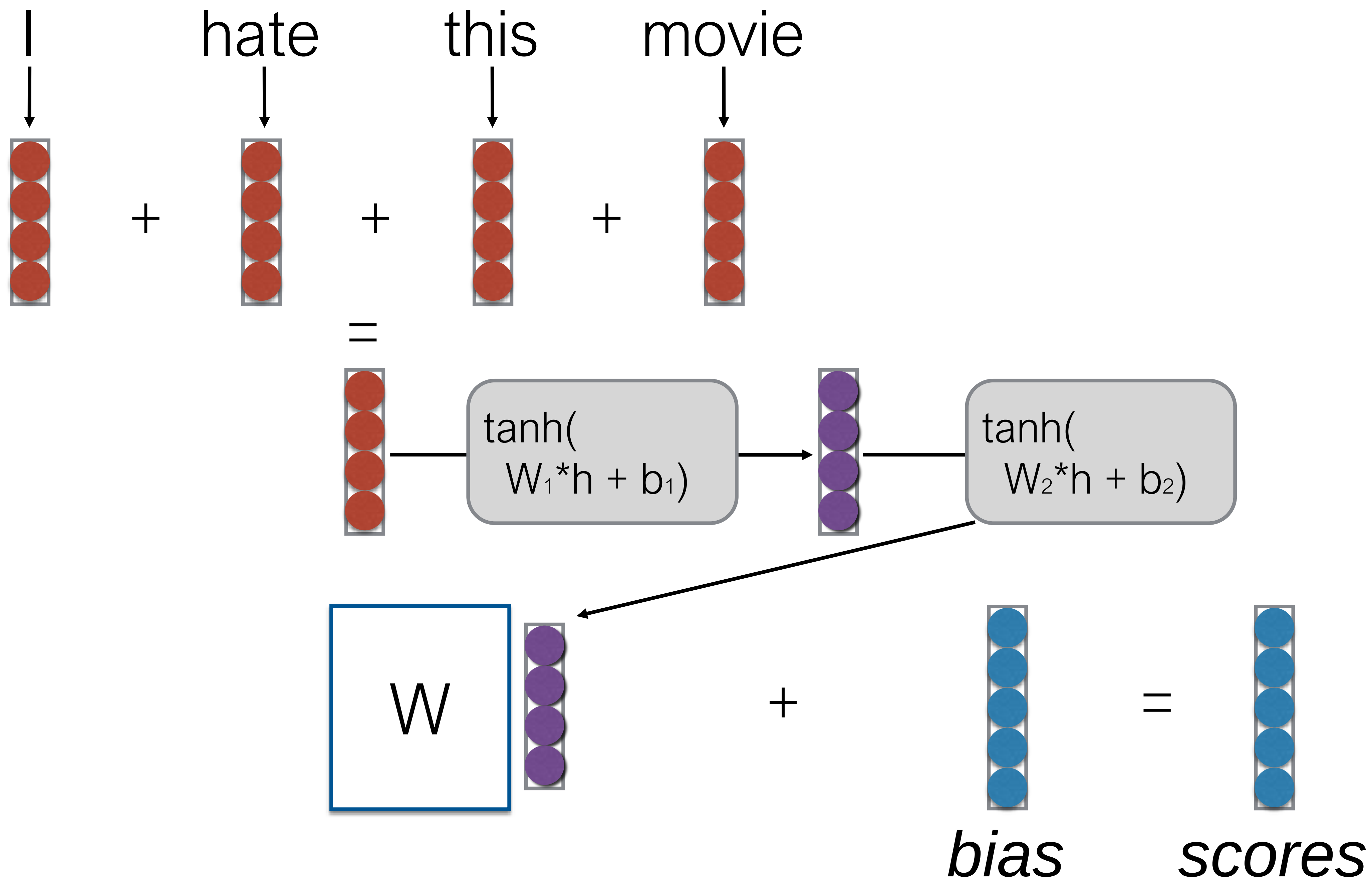
What do Our Vectors Represent?

- Each vector has “features” (e.g. is this an animate object? is this a positive word, etc.)

- We sum these features, then use these to make predictions

- Still no combination features: only the expressive power of a linear model, but dimension reduced

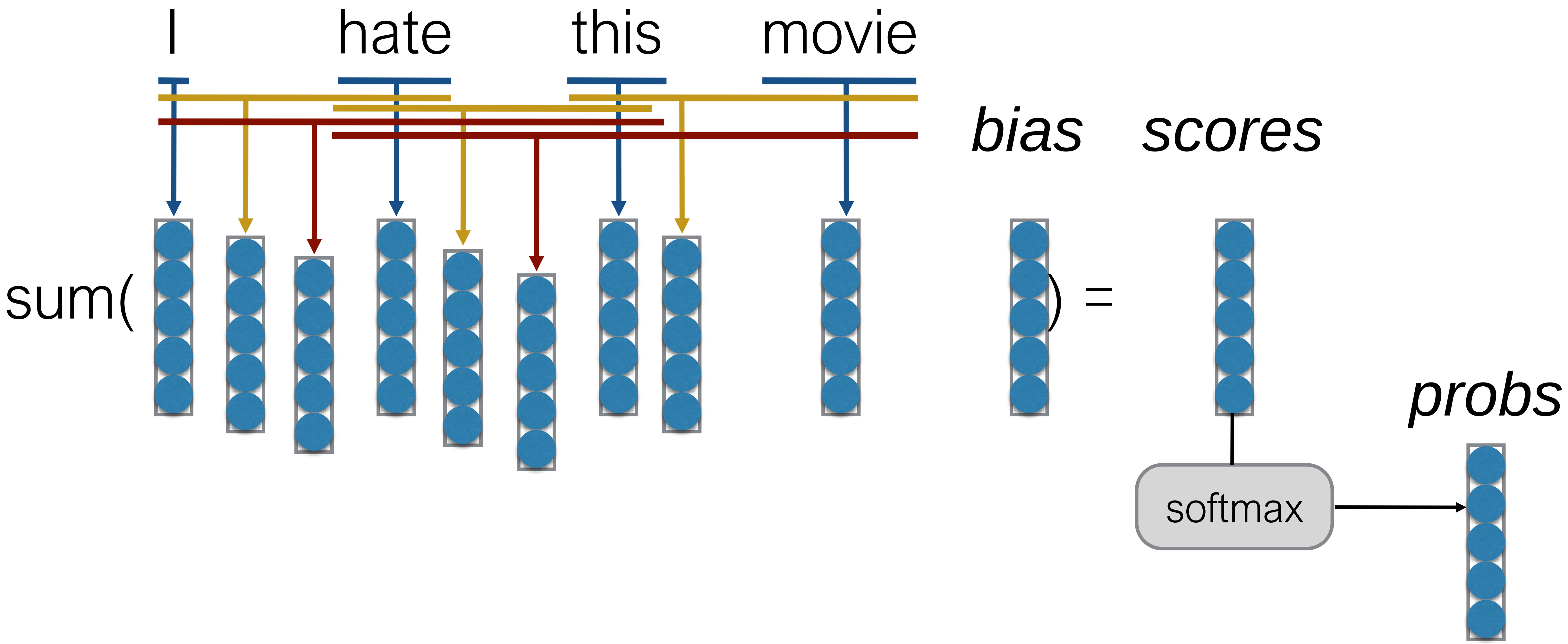
Deep CBOW



What do Our Vectors Represent?

- Now things are more interesting!
- We can learn feature combinations (a node in the second layer might be “feature 1 AND feature 5 are active”)
- e.g. capture things such as “not” AND “hate”
- BUT! Cannot handle “not hate”

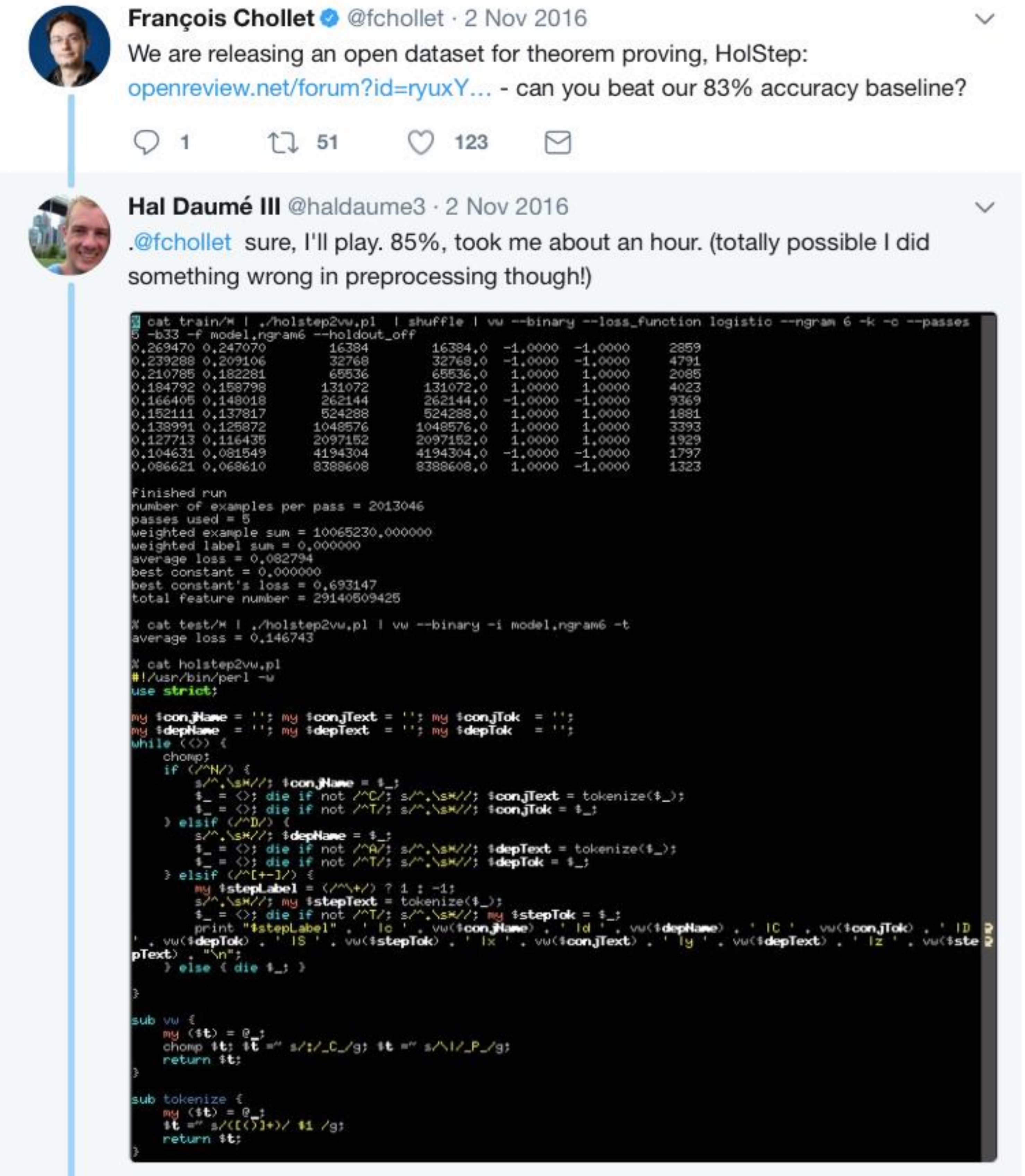
Modeling Sentences w/ n-grams



Why Bag of n-grams?

- Allow us to capture combination features in a simple way “don’t love”, “not the best”

- Works pretty well



François Chollet @fchollet · 2 Nov 2016

We are releasing an open dataset for theorem proving, HolStep: openreview.net/forum?id=ryuxY... - can you beat our 83% accuracy baseline?

1 51 123

Hal Daumé III @haldaume3 · 2 Nov 2016

.@fchollet sure, I'll play. 85%, took me about an hour. (totally possible I did something wrong in preprocessing though!)

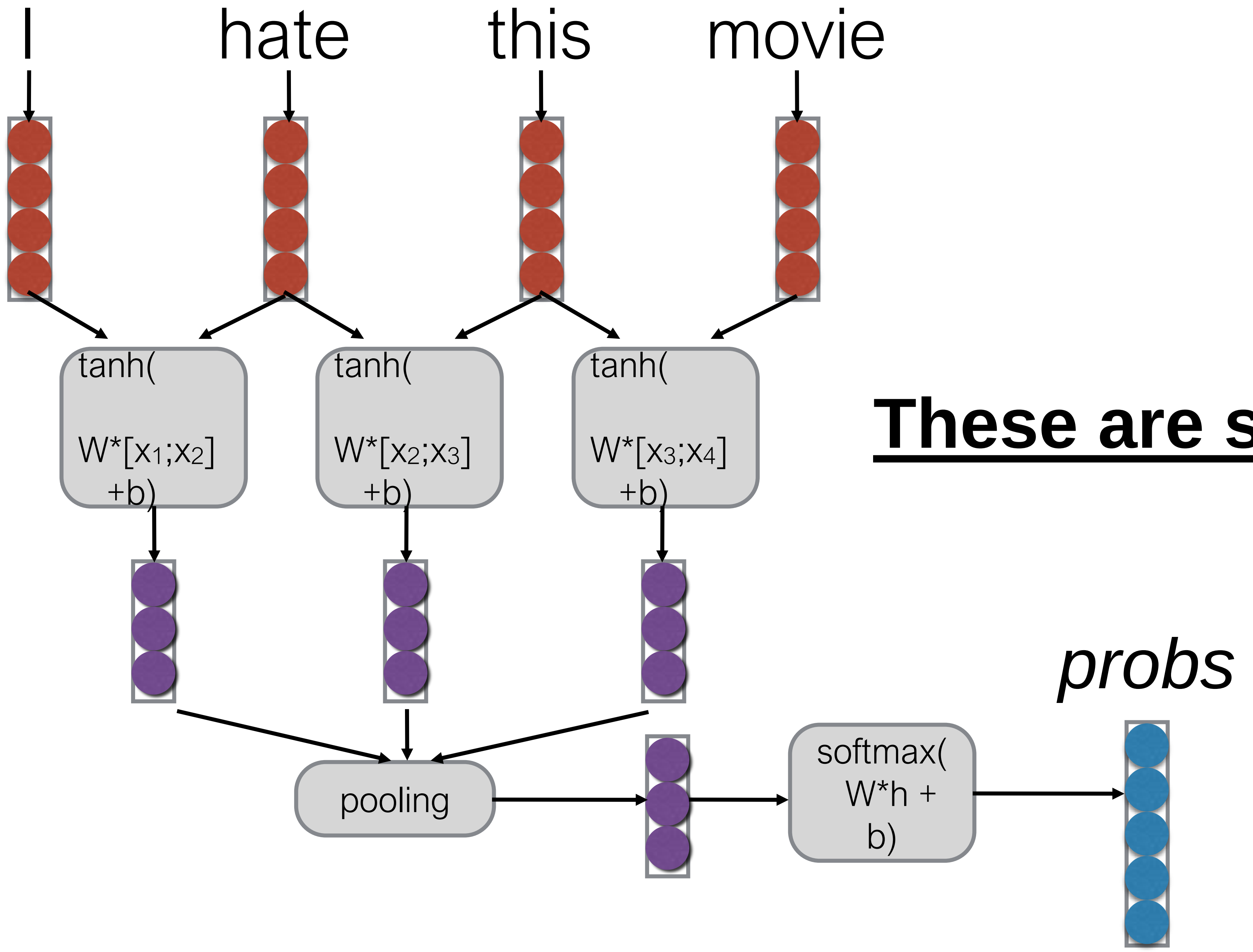
```
% cat train/* | ./holstep2vw.pl  | shuffle | vw --binary --loss_function logistic --ngram 6 -k -c --passes
5 -b33 -f model.ngram6 --holdout_off
0.269470 0.247070        16384       16384.0  -1.0000  -1.0000      2859
0.239288 0.209106        32768       32768.0  -1.0000  -1.0000      4791
0.210785 0.182281        65536       65536.0   1.0000   1.0000      2085
0.184792 0.158798       131072      131072.0   1.0000   1.0000      4023
0.166405 0.148018       262144      262144.0  -1.0000  -1.0000      9369
0.152111 0.137817       524288      524288.0   1.0000   1.0000      1881
0.138991 0.125872      1048576     1048576.0   1.0000   1.0000      3393
0.127713 0.116435      2097152     2097152.0   1.0000   1.0000      1929
0.104631 0.081549      4194304     4194304.0  -1.0000  -1.0000      1797
0.086621 0.068610      8388608     8388608.0   1.0000  -1.0000      1323

finished run
number of examples per pass = 2013046
passes used = 5
weighted example sum = 10065230.000000
weighted label sum = 0.000000
average loss = 0.082794
best constant = 0.000000
best constant's loss = 0.693147
total feature number = 29140509425

% cat test/* | ./holstep2vw.pl | vw --binary -i model.ngram6 -t
average loss = 0.146743

% cat holstep2vw.pl
#!/usr/bin/perl -w
use strict;

my $conjName = ''; my $conjText = ''; my $conjTok  = '';
my $depName  = ''; my $depText  = ''; my $depTok   = '';
while (<>) {
    chomp;
    if (/^N/) {
        s/^.\s*//; $conjName = $_;
        $_ = <>; die if not /^C/; s/^.\s*//; $conjText = tokenize($_);
        $_ = <>; die if not /^T/; s/^.\s*//; $conjTok = $_;
    } elsif (/^D/) {
        s/^.\s*//; $depName = $_;
        $_ = <>; die if not /^A/; s/^.\s*//; $depText = tokenize($_);
        $_ = <>; die if not /^T/; s/^.\s*//; $depTok = $_;
    } elsif (/^[+-]/) {
        my $stepLabel = (/^\+/) ? 1 : -1;
        s/^.\s*//; my $stepText = tokenize($_);
        $_ = <>; die if not /^T/; s/^.\s*//; my $stepTok = $_;
        print "$stepLabel" . ' |c ' . vw($conjName) . ' |d ' . vw($depName) . ' |C ' . vw($conjTok) . ' |D 
' . vw($depTok) . ' |S ' . vw($stepTok) . ' |x ' . vw($conjText) . ' |y ' . vw($depText) . ' |z ' . vw($ste
pText) . "\n";
    } else { die $_; }

}

sub vw {
    my ($t) = @_;
    chomp $t; $t =~ s/:/_C_/g; $t =~ s/\|/_P_/g;
    return $t;
}

sub tokenize {
    my ($t) = @_;
    $t =~ s/([()]+)/ $1 /g;
    return $t;
}
```

What Problems
w/ Bag of n-grams?

- Leads to sparsity: many of the n-grams will never be seen in a training corpus
- No sharing between similar words/n-grams

Time Delay Neural Networks

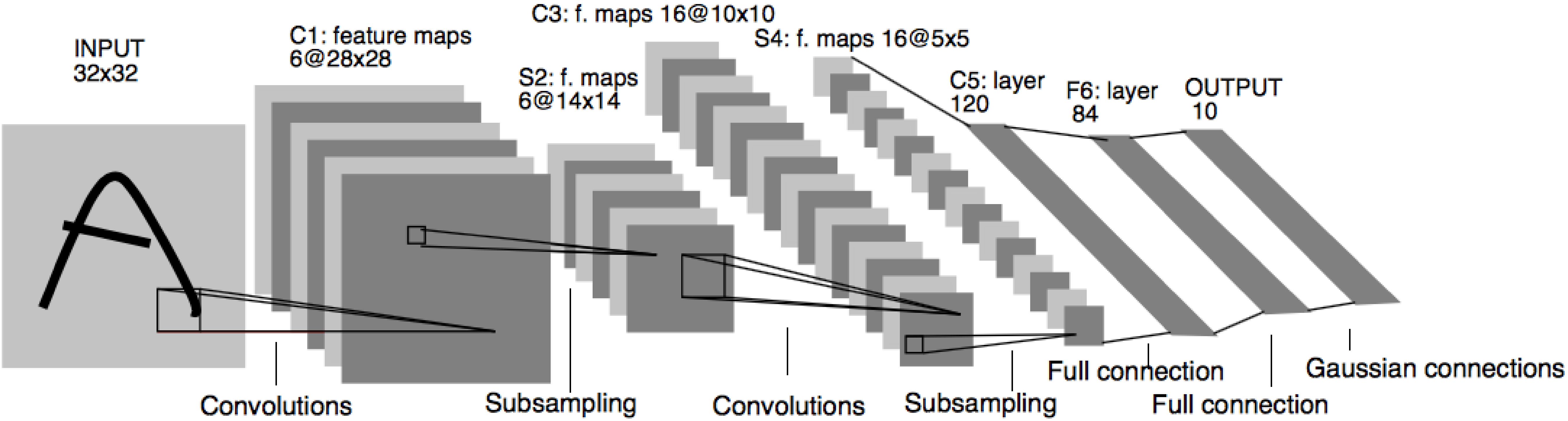
(Waibel et al. 1989)



These are soft 2-grams!

Convolutional Networks

(LeCun et al. 1997)



Parameter extraction performs a 2D sweep, not 1D

CNNs/TDNNs for Text

(Collobert and Weston 2011)

- Generally 1D convolution ≈ Time Delay Neural Network
  - But often uses terminology/functions borrowed from image processing
- Two main paradigms:
  - **Context window modeling:** For tagging, etc. get the surrounding context before tagging
  - **Sentence modeling:** Do convolution to extract n-grams, pooling to combine over whole sentence

Pooling

- Calculate some reduction function feature-wise
- **Max pooling:** “Did you see this feature anywhere in the range?” (most common)
- **Average pooling:** “How prevalent is this feature over the entire range”
- **k-Max pooling:** “Did you see this feature up to k times?”
- **Dynamic pooling:** “Did you see this feature in the beginning? In the middle? In the end?”

Weaknesses of CNNs

- CNNs are great for short-distance feature extractors
- But don't have holistic view of the sentence to capture long-distance dependencies

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

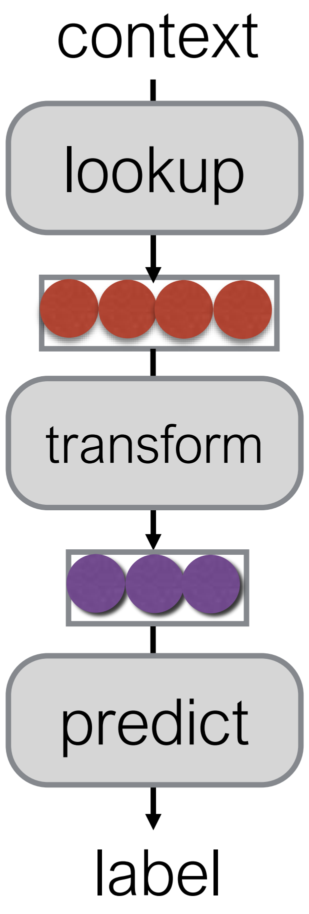
The **reign** has lasted as long as the life of the **queen**.
The **rain** has lasted as long as the life of the **clouds**.

Recurrent Neural Networks

(Elman 1990)

- Tools to “remember” information

Feed-forward NN



context

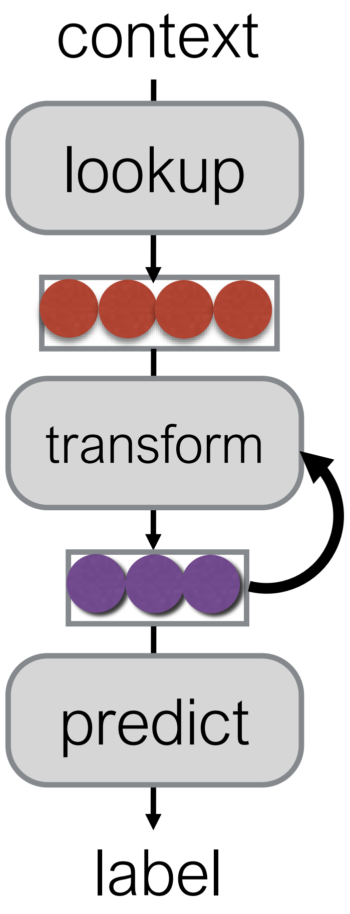
lookup

transform

predict

label

Recurrent NN



context

lookup

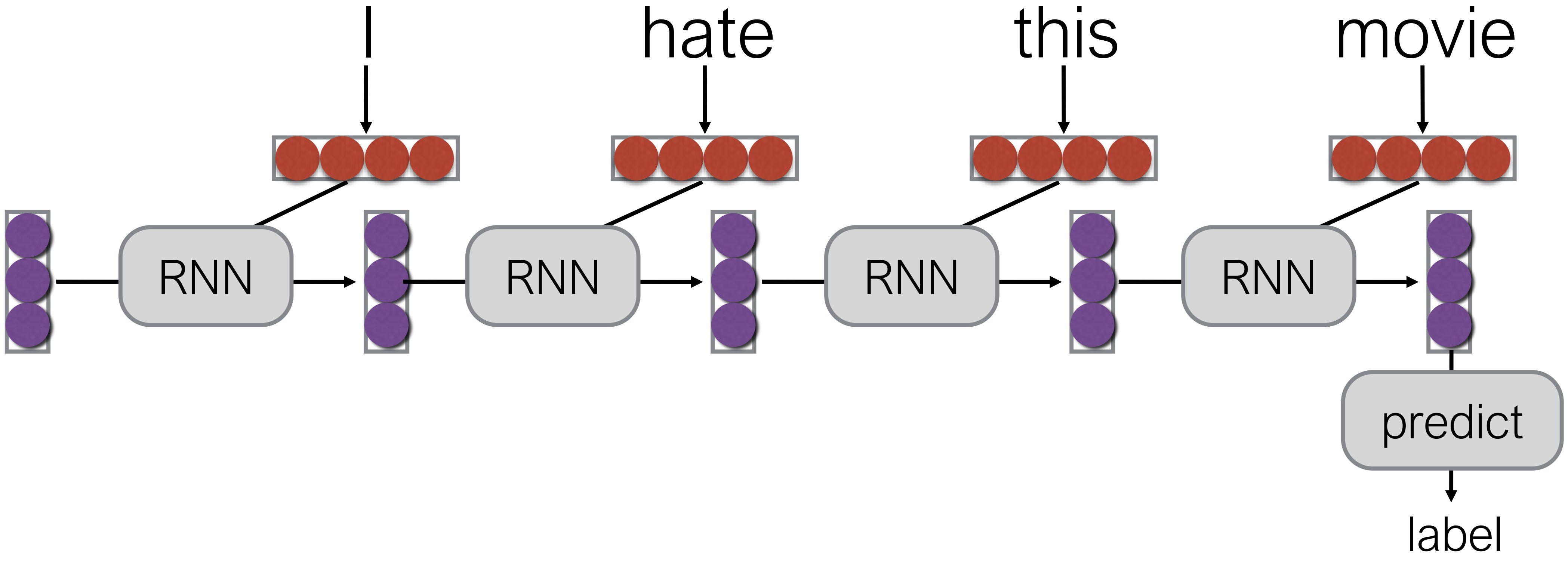
transform

predict

label

Making Predictions w/ RNNs

- What does processing a sequence look like?



Weaknesses of RNNs

- Indirect passing of information, credit assignment more difficult
    - Made better by LSTMs/GRUs/etc. but not perfect
- Can be slow, due to incremental processing

Back to
Language Modeling:
Models of P(text)

Are These Sentences OK?

- Jane went to the store.
- store to Jane went the.
- Jane went store.
- Jane goed to the store.
- The store went to Jane.
- The food truck went to Jane.

What Can we Do w/ LMs?

- Score sentences:

Jane went to the store . → high
store to Jane went the . → low

(same as calculating loss for training)

- Generate sentences:

```
while didn't choose end-of-sentence symbol:
  calculate probability
  sample a new word from the probability distribution
```

Calculating the Probability of a Sentence

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

This is a classification problem over the next word!

$$P(x_i \mid x_1, \ldots, x_{i-1})$$

How do we do this?!?!

Count-based Language Models

- Count up the frequency and divide:

$$P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i)}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

- Add smoothing, to deal with zero counts:

$$\begin{aligned} P(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) =& \lambda P_{ML}(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) \\ &+ (1-\lambda) P(x_i \mid x_{1-n+2}, \ldots, x_{i-1}) \end{aligned}$$

Problems w/ Count-based Models

- Cannot share strength among **similar words**

she bought a car she bought a bicycle
she purchased a car she purchased a bicycle

- Cannot condition on context with **intervening words**

Dr. Jane Smith Dr. Gertrude Smith

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet
for programming class he wanted to buy his own computer

An Alternative: Featurized Models

- Calculate features of the context
- Based on the features, calculate probabilities
- Optimize feature weights using gradient descent, etc.

Example:

Previous words: “giving a"

$$\begin{array}{c} \text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \dots \end{array} \quad b=\begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \dots \end{pmatrix} \quad w_{1,a}=\begin{pmatrix} -6.0 \\ -5.1 \\ 0.2 \\ 0.1 \\ 0.5 \\ \dots \end{pmatrix} \quad w_{2,giving}=\begin{pmatrix} -0.2 \\ -0.3 \\ 1.0 \\ 2.0 \\ -1.2 \\ \dots \end{pmatrix} \quad s=\begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \dots \end{pmatrix}$$

Words we’re predicting

How likely are they?

How likely are they given prev. word is “a”?

How likely are they given 2nd prev. word is “giving”?

Total score

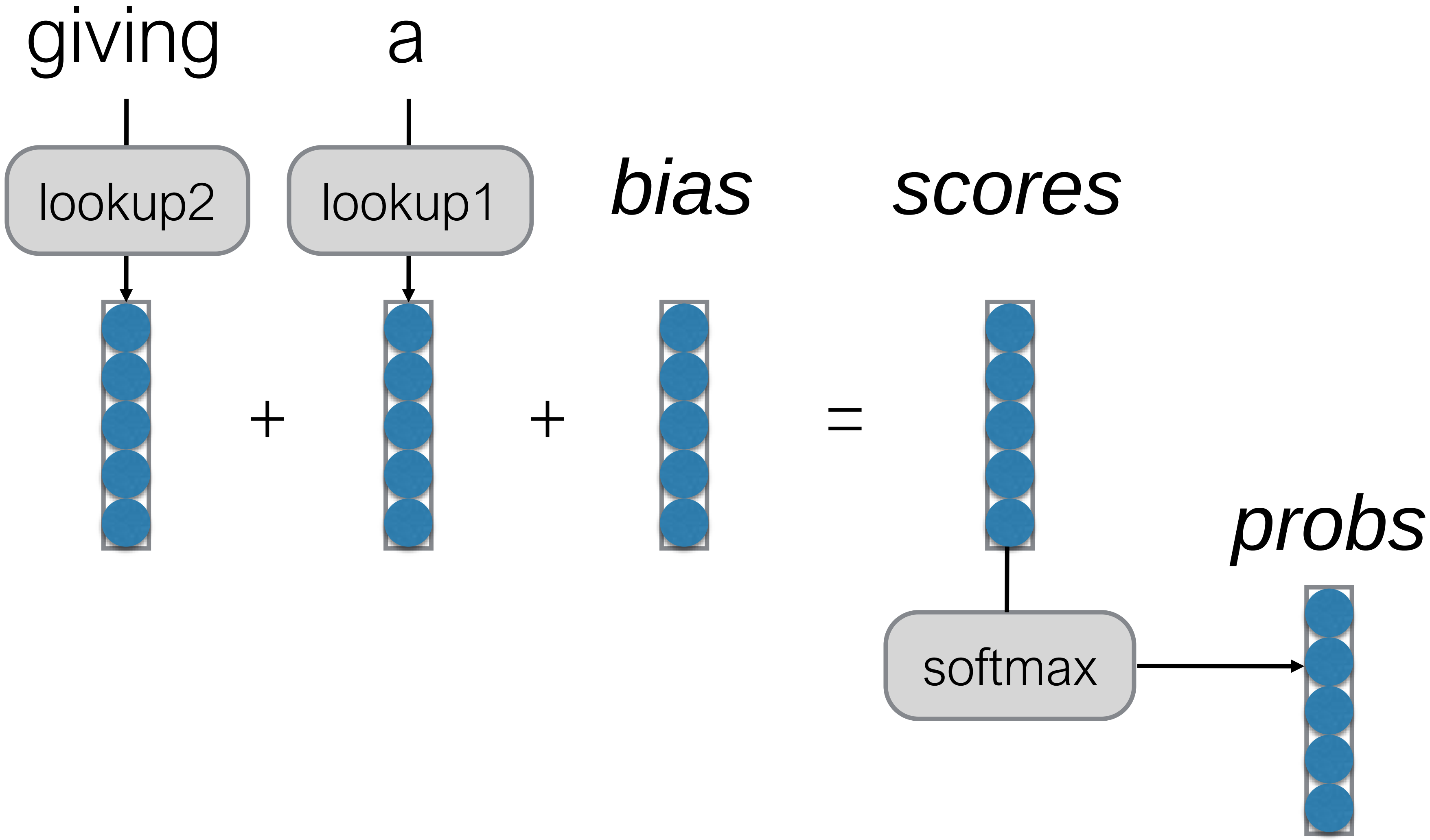
Softmax

- Convert scores into probabilities by taking the exponent and normalizing (softmax)

$$P(x_i \mid x_{i-n+1}^{i-1}) = \frac{e^{s(x_i \mid x_{i-n+1}^{i-1})}}{\sum_{\tilde{x}_i} e^{s(\tilde{x}_i \mid x_{i-n+1}^{i-1})}}$$

$$s = \begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \ldots \end{pmatrix} \longrightarrow p = \begin{pmatrix} 0.002 \\ 0.003 \\ 0.329 \\ 0.444 \\ 0.090 \\ \ldots \end{pmatrix}$$

A Computation Graph View



Each vector is size of output vocabulary

What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car
she purchased a car

she bought a bicycle
she purchased a bicycle

→ not solved yet 😔

- Cannot condition on context with **intervening words**

Dr. Jane Smith Dr. Gertrude Smith

→ solved! 😀

- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet
for programming class he wanted to buy his own computer

→ not solved yet 😔

Linear Models can't Learn Feature Combinations

farmers eat steak → **high**

farmers eat hay → **low**

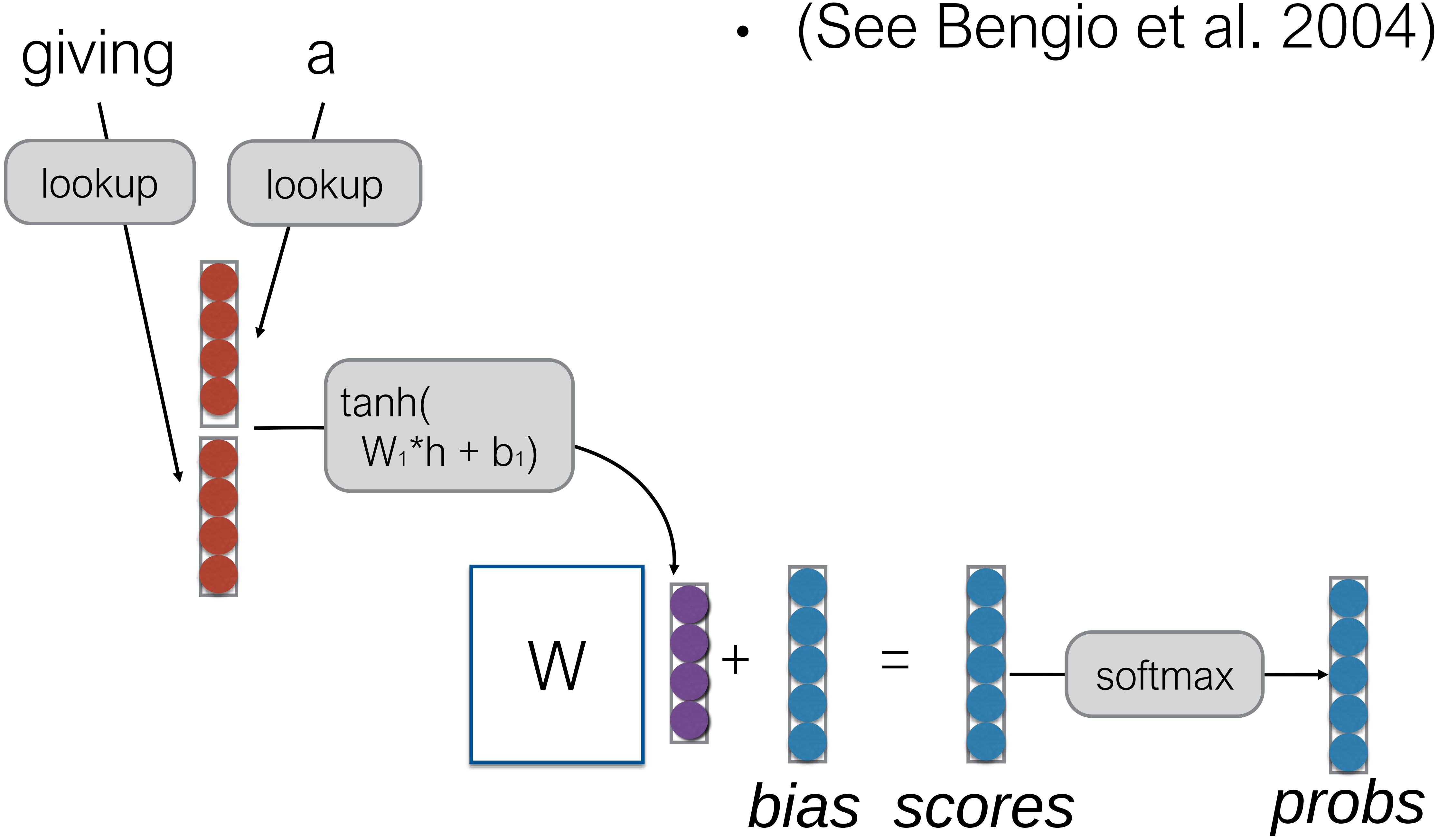
cows eat steak → **low**

cows eat hay → **high**

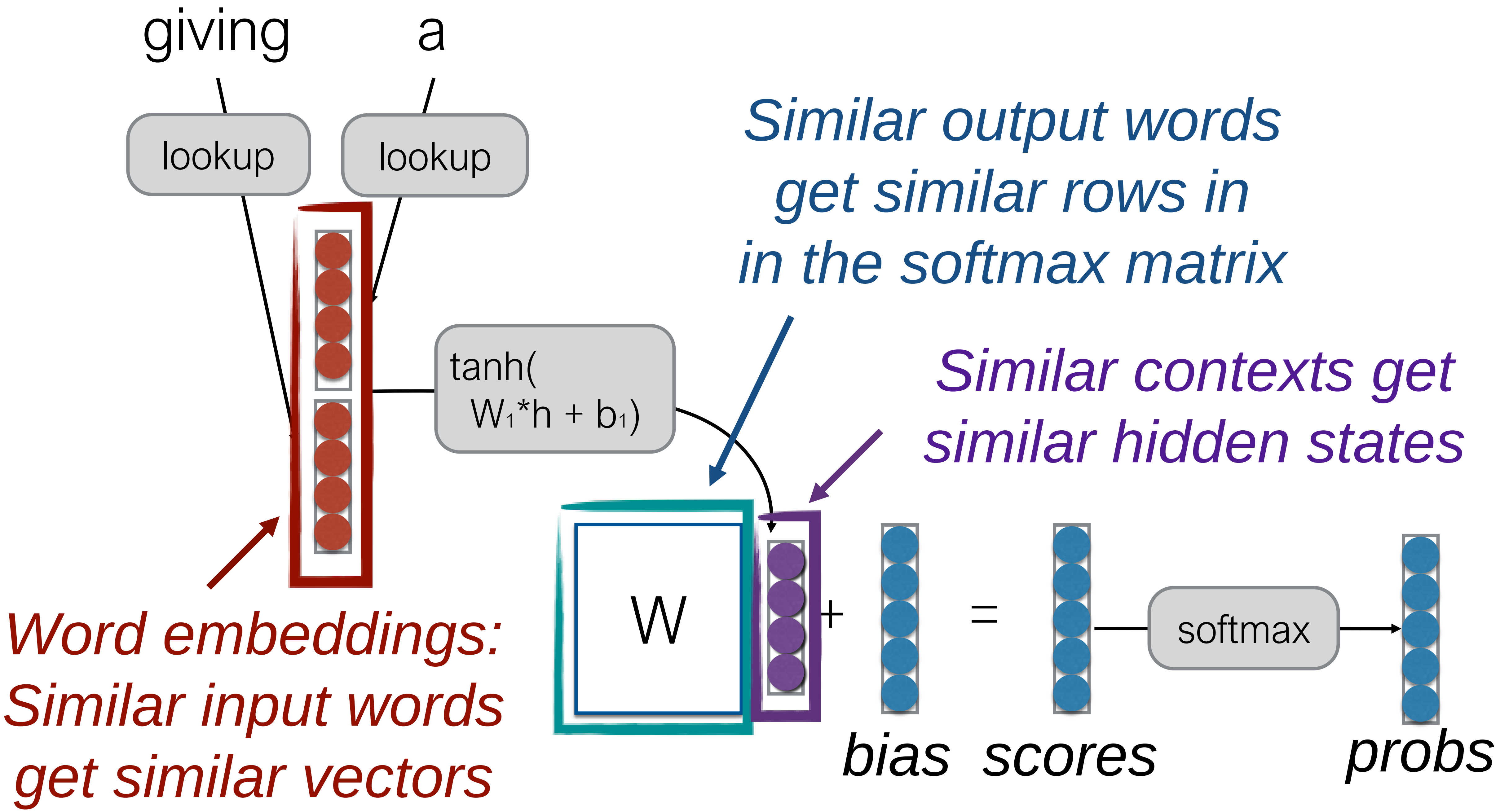
- These can't be expressed by linear features
- What can we do?
  - Remember combinations as features (individual scores for "farmers eat", "cows eat")
    → Feature space explosion!
  - Neural nets

Neural Language Models

- (See Bengio et al. 2004)



Where is Strength Shared?



Similar output words get similar rows in in the softmax matrix

Similar contexts get similar hidden states

Word embeddings: Similar input words get similar vectors

What Problems are Handled?

- Cannot share strength among **similar words**

she bought a car
she purchased a car

she bought a bicycle
she purchased a bicycle

→ solved, and similar contexts as well! 😁

- Cannot condition on context with **intervening words**

Dr. Jane Smith   Dr. Gertrude Smith

→ solved! 😁

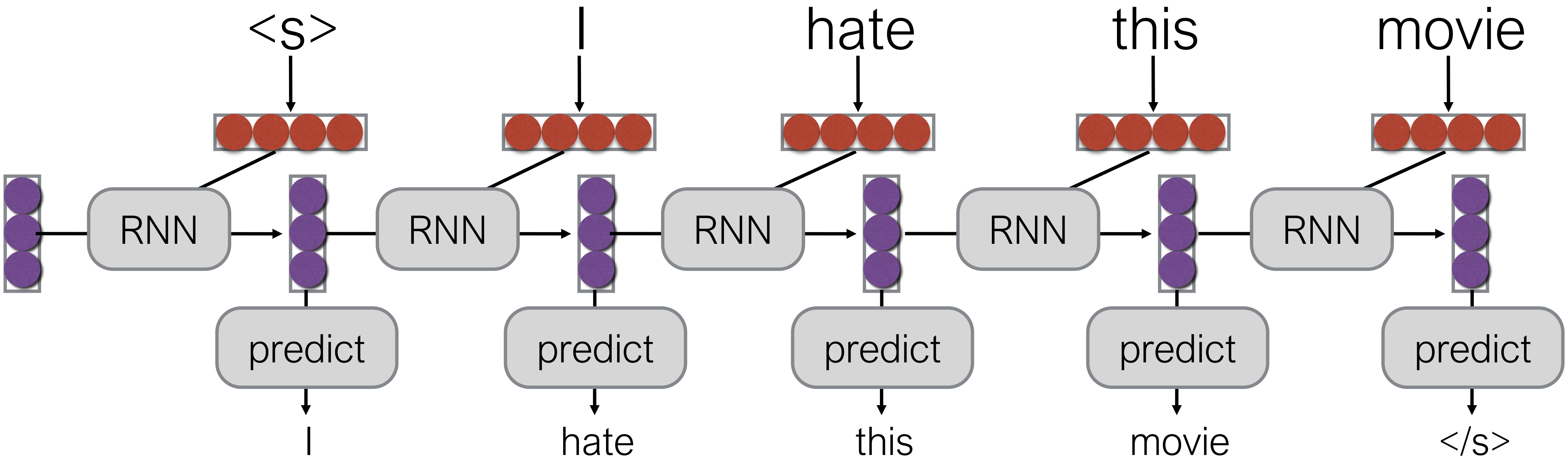
- Cannot handle **long-distance dependencies**

for tennis class he wanted to buy his own racquet

for programming class he wanted to buy his own computer

→ not solved yet 😣

RNN Language Models



- Can handle long-range dependencies, as long as we can learn!

Evaluation of LMs

- **Log-likelihood:**

$$LL(\mathcal{E}_{test}) = \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word Log Likelihood:**

$$WLL(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} \log P(E)$$

- **Per-word (Cross) Entropy:**

$$H(\mathcal{E}_{test}) = \frac{1}{\sum_{E \in \mathcal{E}_{test}} |E|} \sum_{E \in \mathcal{E}_{test}} -\log_2 P(E)$$

- **Perplexity:**

$$ppl(\mathcal{E}_{test}) = 2^{H(\mathcal{E}_{test})} = e^{-WLL(\mathcal{E}_{test})}$$

Sequence Transduction:
Models of P(text | text)

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

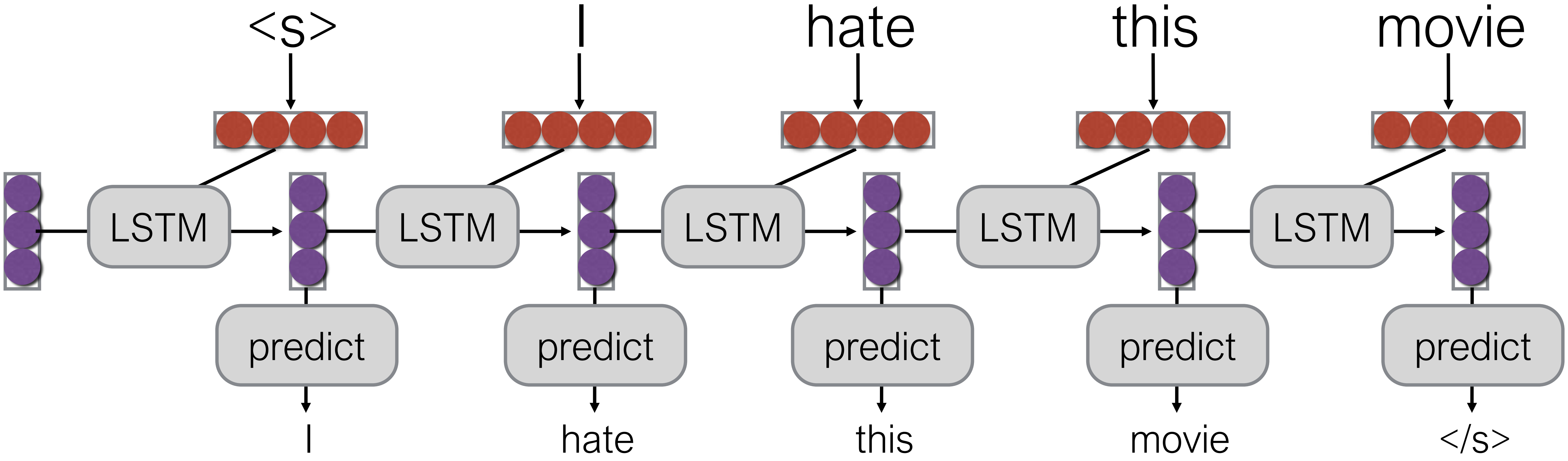
Calculating the Probability of a Sentence

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

Conditional Language Models

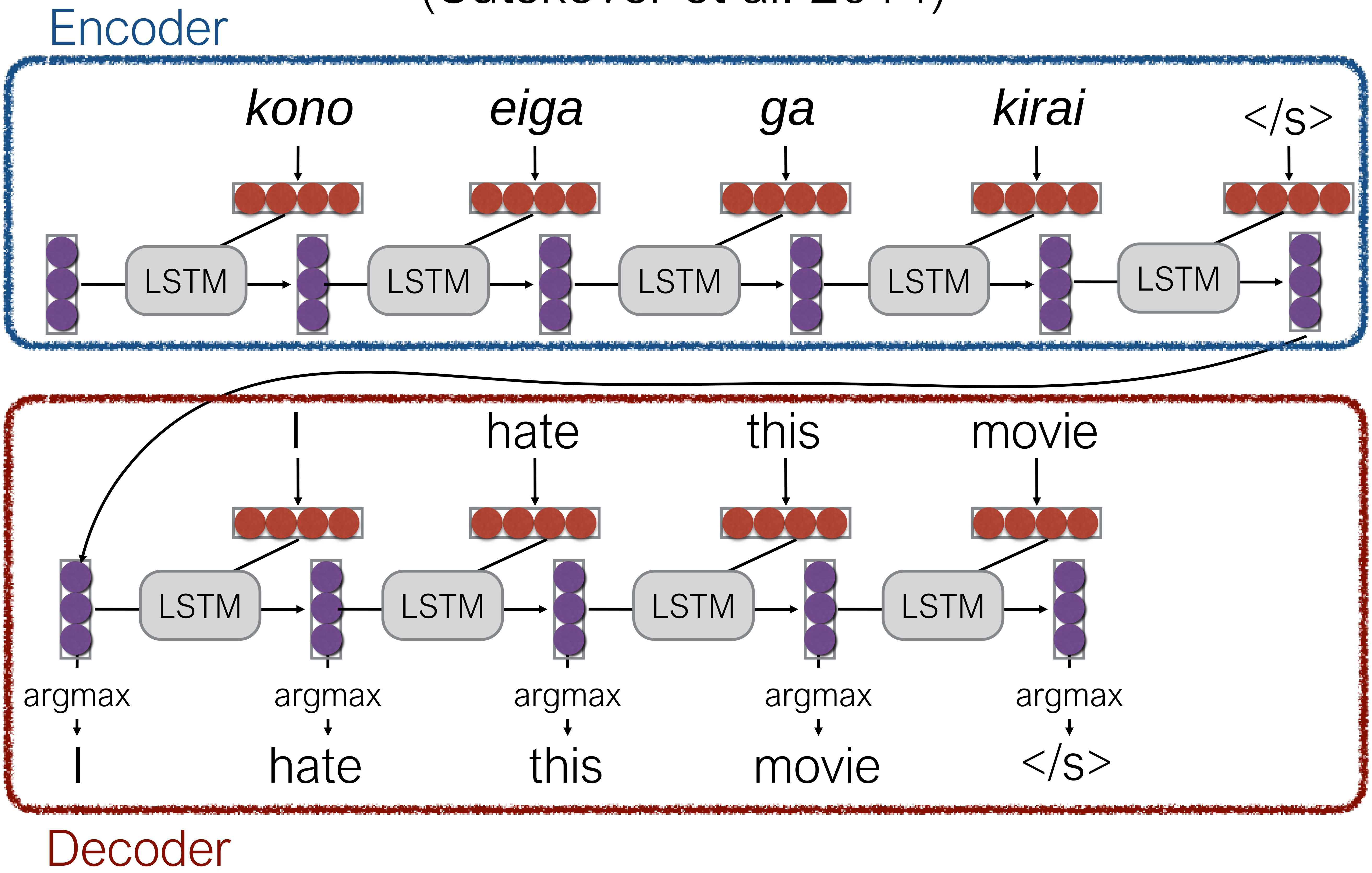
$$P(Y|X) = \prod_{j=1}^{J} P(y_j \mid \underset{\substack{\uparrow \\ \text{Added Context!}}}{\underline{X}}, y_1, \ldots, y_{j-1})$$

(One Type of) Language Model
(Mikolov et al. 2011)



(One Type of) Conditional Language Model

(Sutskever et al. 2014)



Encoder

kono *eiga* *ga* *kirai* </s>

LSTM LSTM LSTM LSTM LSTM

I hate this movie

LSTM LSTM LSTM LSTM

argmax argmax argmax argmax argmax

I hate this movie </s>

Decoder

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.

Argmax Search

- **Greedy search:** One by one, pick the single highest-probability word

```
while $y_{j-1}$ != "</s>":
  $y_j = \text{argmax } P(y_j \mid X, y_1, \ldots, y_{j-1})$
```

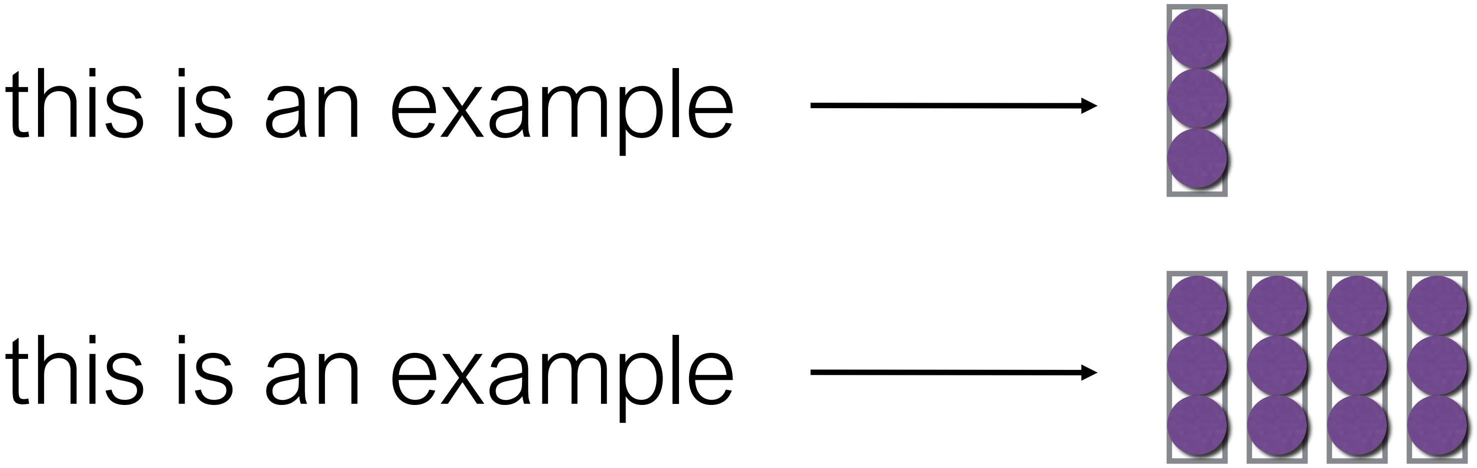
- **Beam search:** keep multiple hypotheses

Representing Sentences as Vectors

<u>Problem!</u>

> "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.



this is an example →

this is an example →

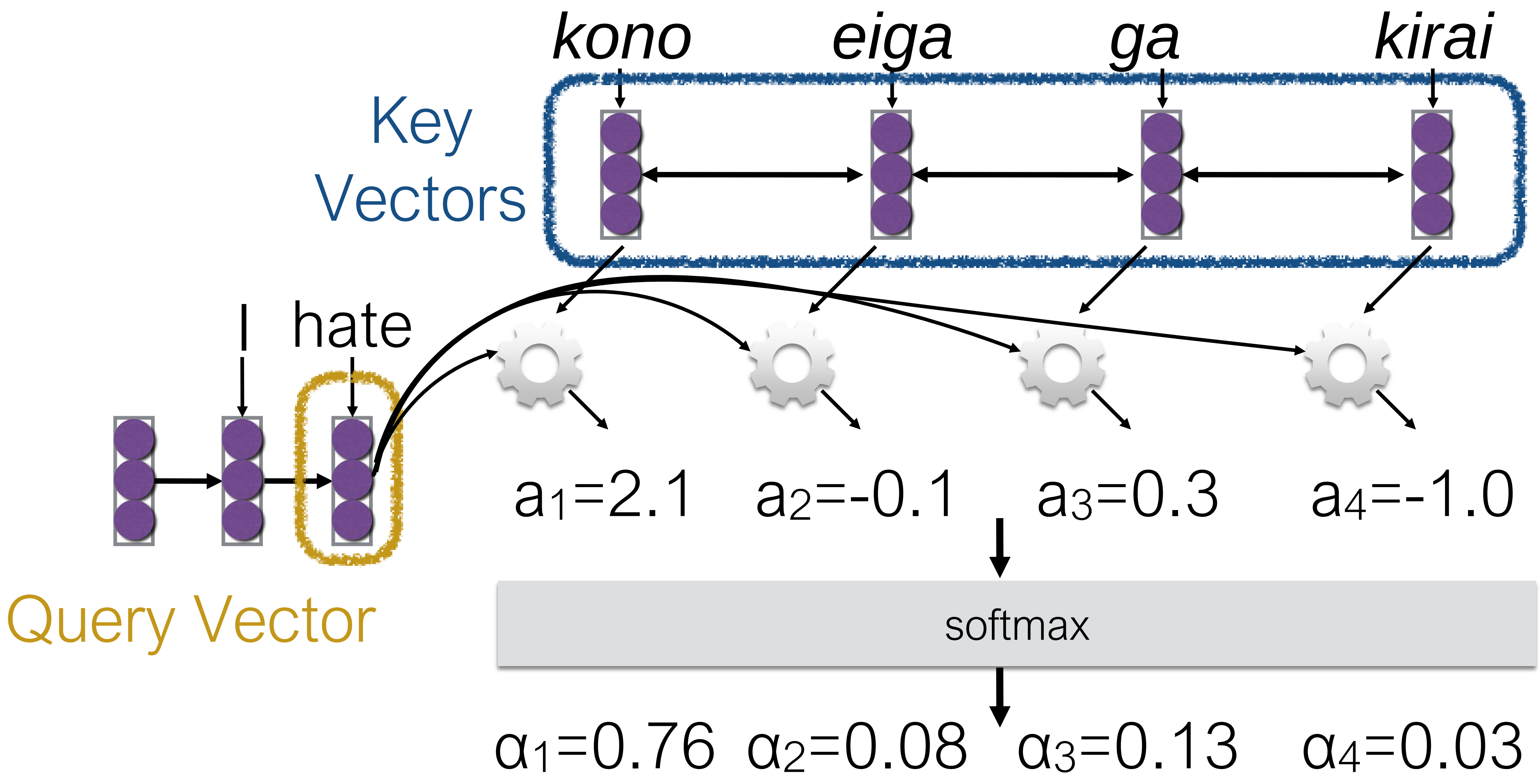
"Attention"!

(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

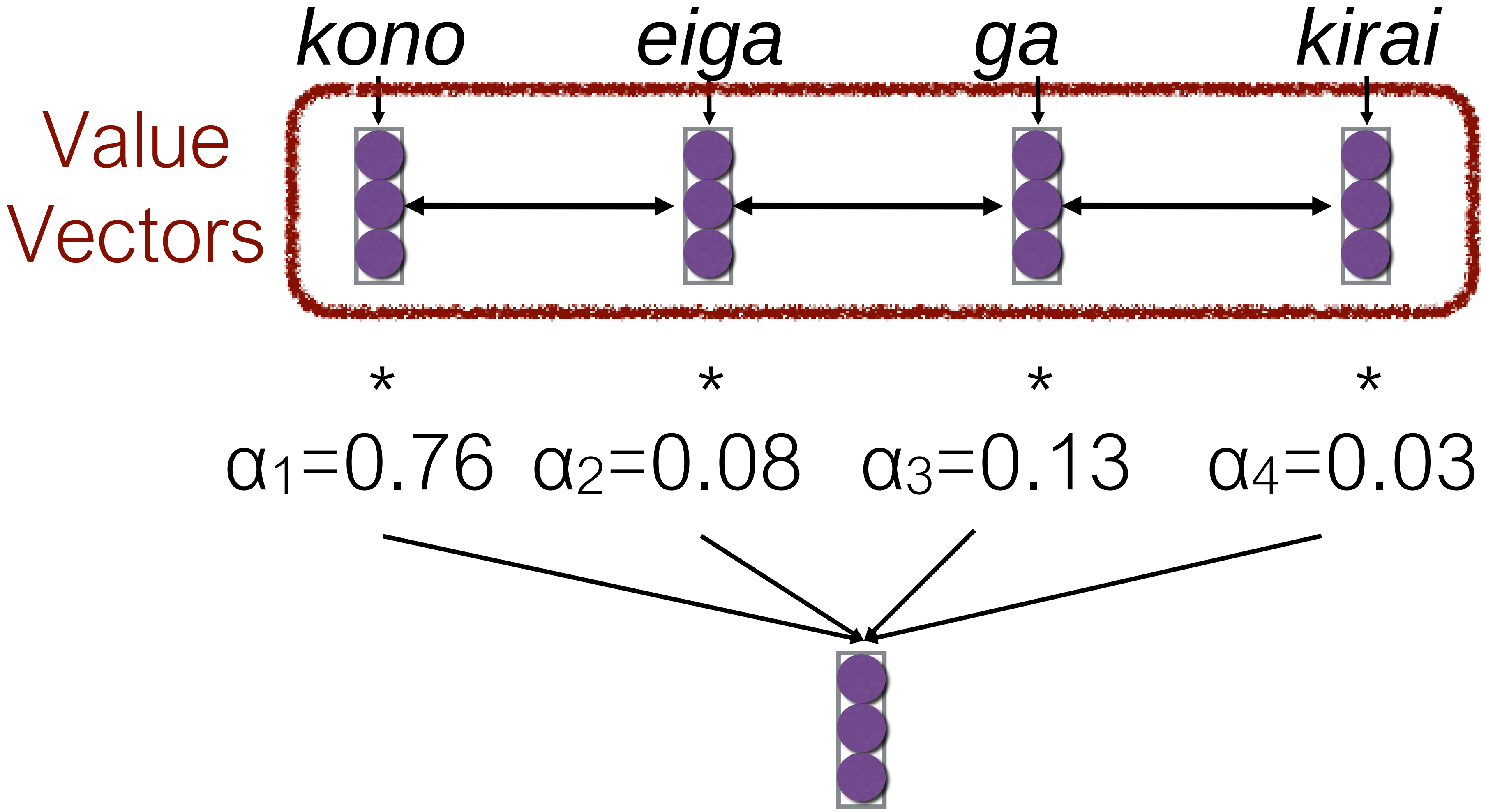
Calculating Attention (1)

- 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



kono *eiga* *ga* *kirai*

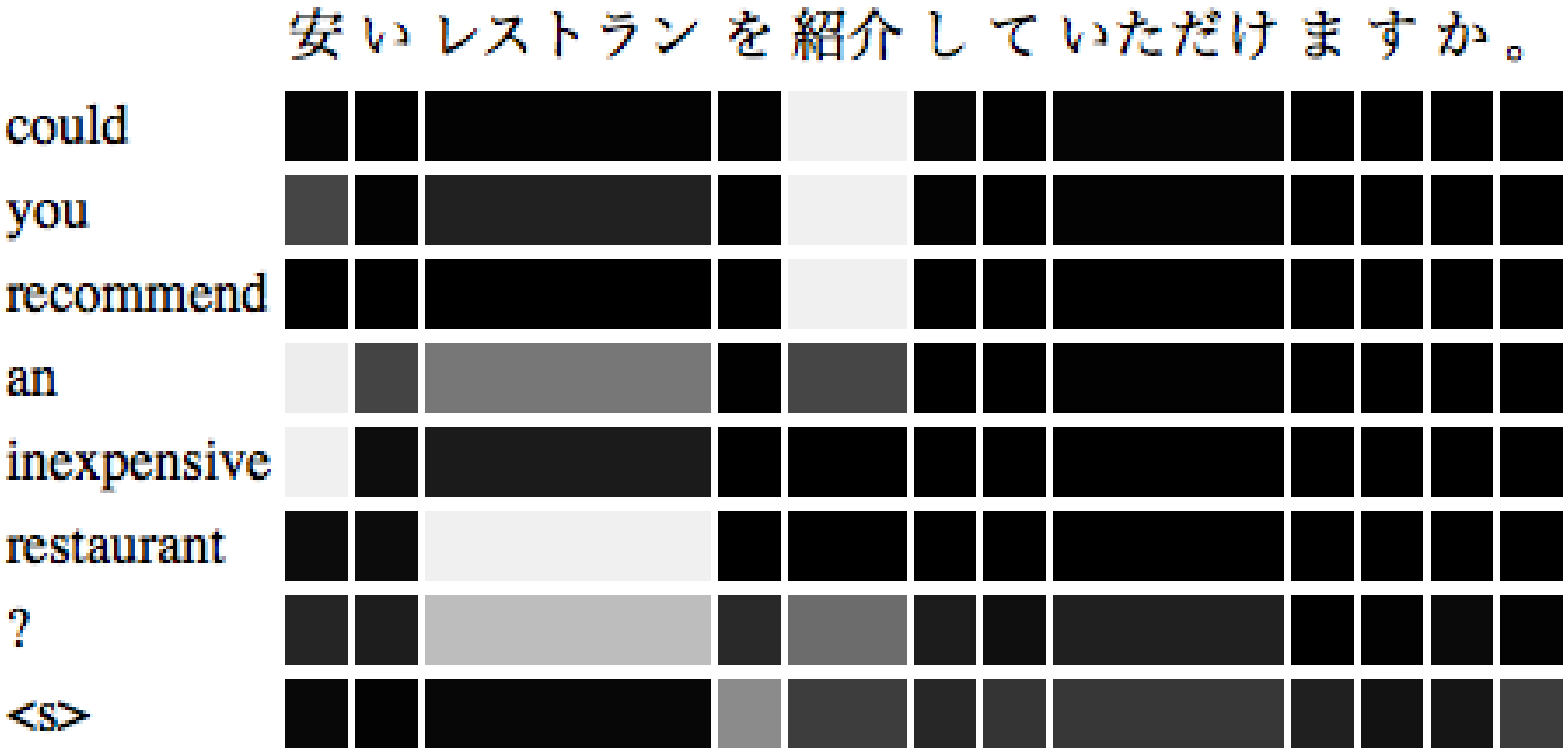
Value Vectors

* * * *

$\alpha_1=0.76$ $\alpha_2=0.08$ $\alpha_3=0.13$ $\alpha_4=0.03$

- Use this in any part of the model you like

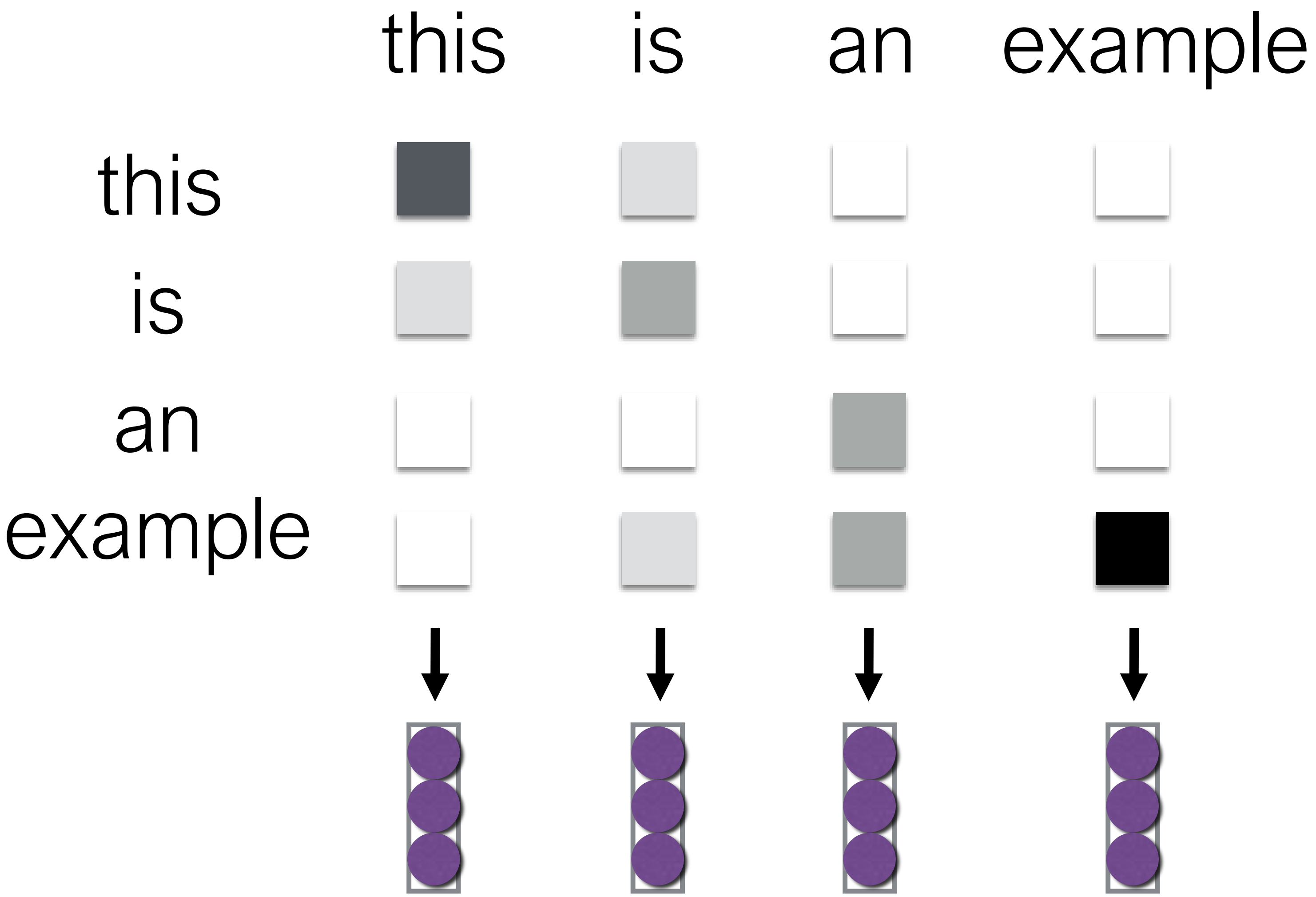
A Graphical Example



Intra-Attention / Self Attention

(Cheng et al. 2016)

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

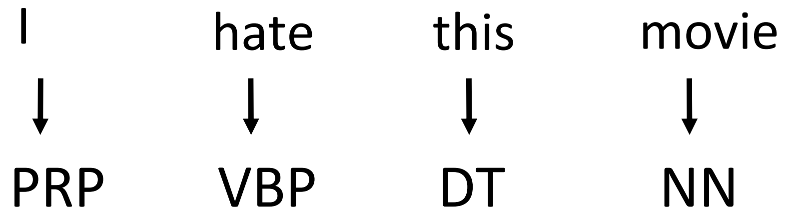


- Excellent results for translation in Vaswani et al. (2017)

Language Analysis: Models of P(structure | text)

Analysis Tasks

- **Tagging:**



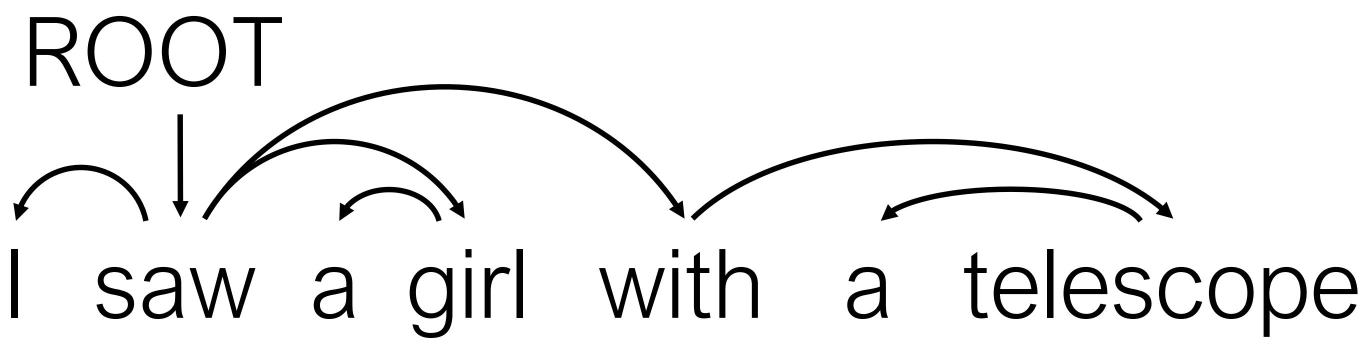
I → PRP

hate → VBP

this → DT

movie → NN

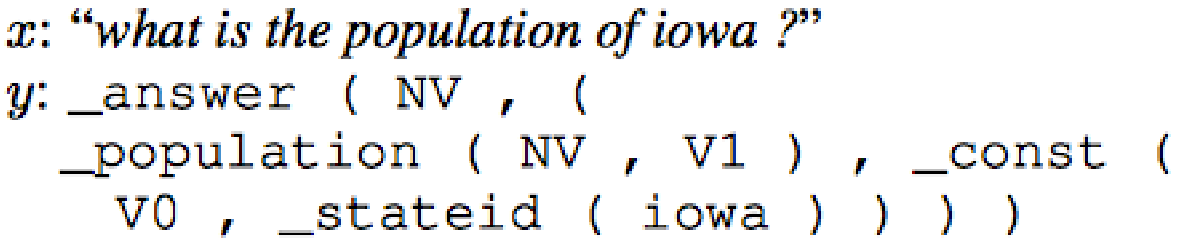
- **Syntactic Parsing:**



ROOT

I saw a girl with a telescope

- **Semantic Parsing:**



x: *“what is the population of iowa ?”*

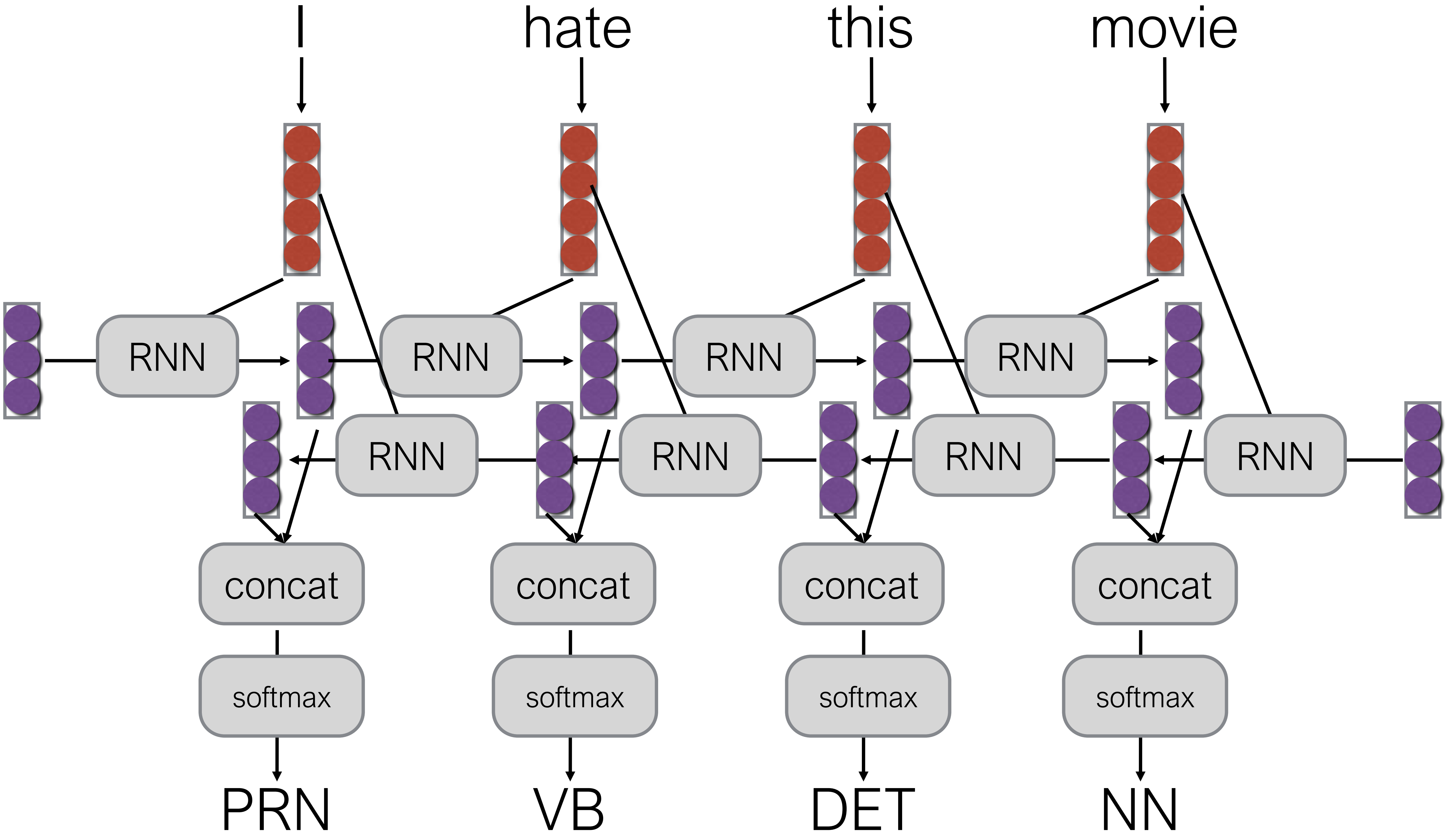
```
y: _answer ( NV , (
  _population ( NV , V1 ) , _const (
    V0 , _stateid ( iowa ) ) ) )
```

copy the content of
file 'file.txt' to file 'file2.txt'

↓

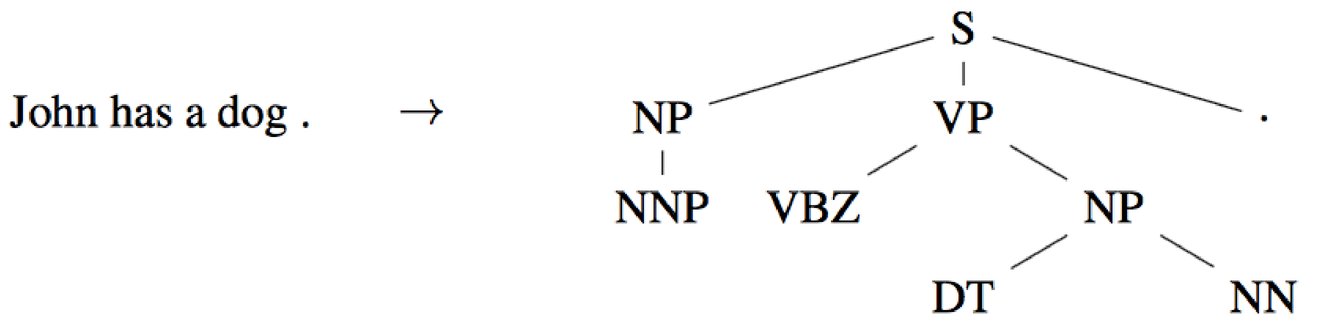
```
shutil.copy(
    'file.txt', 'file2.txt')
```

Simple Tagging: BiLSTMs



Simple Parsing: Linearized Tree + seq2seq

- Convert a tree structure into a sequence of symbols



John has a dog . → $\text{(S (NP NNP)}_{\text{NP}}\text{ (VP VBZ (NP DT NN)}_{\text{NP}}\text{)}_{\text{VP}}\text{ .)}_{\text{S}}$

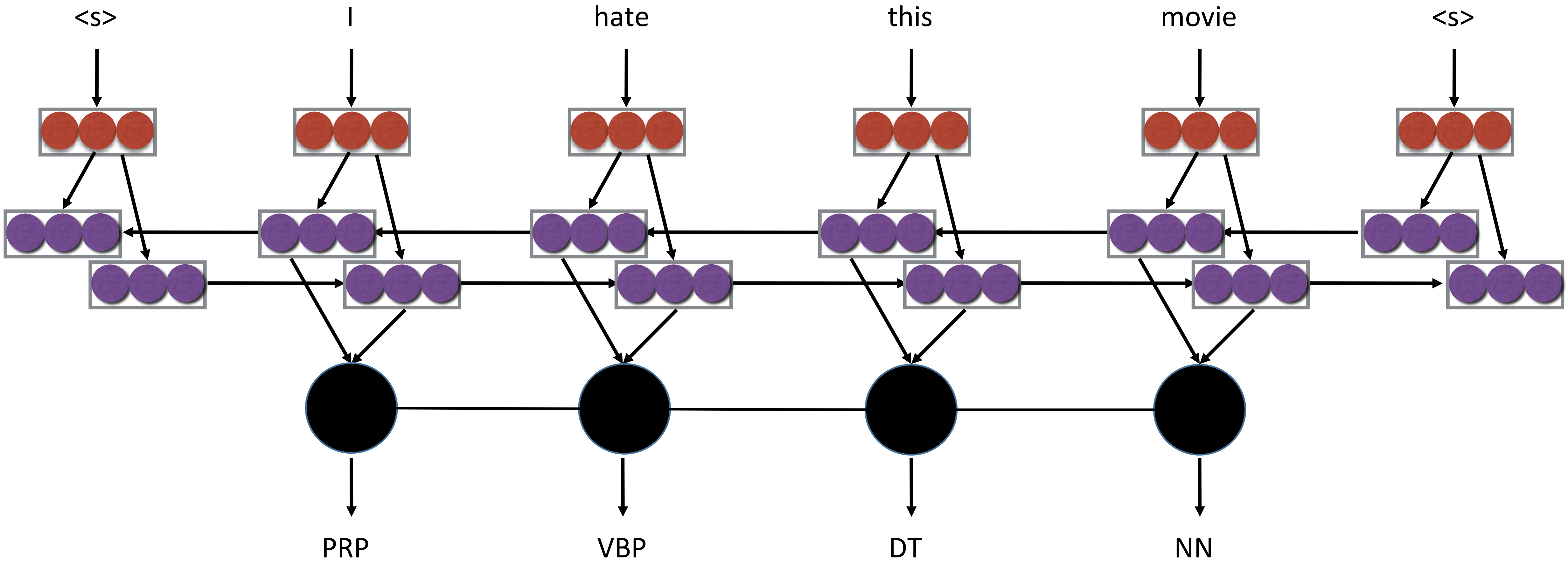
- Solve using standard seq2seq model

Exploiting Problem Structure

- Simple methods can get you pretty far!
  - esp. if you have lots of training data
- But exploiting problem structure can get you farther

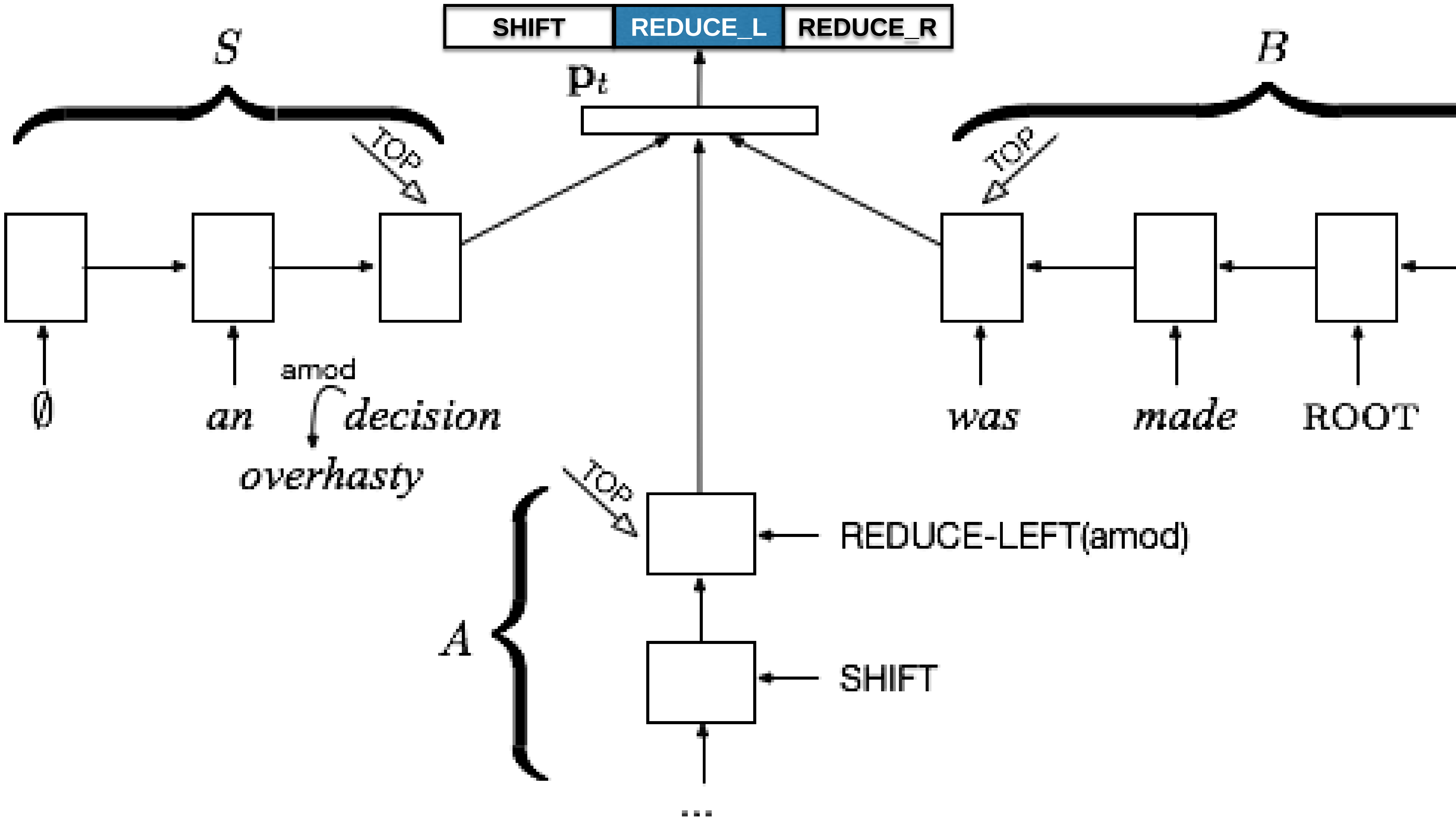
Better Tagging: BiLSTM Conditional Random Fields (e.g. Ma and Hovy 2015)

- Add an additional layer that ensures consistency between tags



- Training and prediction use dynamic programming

Better Parsing: Stack LSTMs (Dyer et al. 2015)



Overcoming Data Sparsity

- Unlike MT or language modeling, analysis tasks there is usually not enough data

- Strong allies:

    - Pre-trained word embeddings (e.g. FastText)

    - Pre-trained language models (e.g. Elmo)

Learn More!

Learn More!

- NLP is a big field, too big for 1.5 hours

- Lots of nice resources online, e.g.:

    - Stanford CS224n: Natural Language Processing with Deep Learning
      https://web.stanford.edu/class/cs224n/

    - CMU CS11-747: Neural Networks for Natural Language Processing
      http://phontron.com/class/nn4nlp2018/

Any Questions?