CS769 Advanced NLP

Language Modeling

Junjie Hu

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Graham

https://junjiehu.github.io/cs769-spring23/

Goals for Today

- Problem definition
- **N-gram Language Model**
- **Log-Linear Language Model**
- **Neural Language Model**
- Evaluation

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

Quick Review of Probability

- Event space (e.g., $\mathscr{X}, \mathscr{Y}$)—in this class, usually discrete
- Random variables (e.g.,X, Y)
- Typical statement: "random variable X takes value $x \in \mathscr{X}$ with probability $P(X = x)$, or in shorthand, $P(x)$"
- Joint probability: $P(X = x, Y = y)$
- Conditional probability: $P(X = x \mid Y = y) = \dfrac{P(X = x, Y = y)}{P(Y = y)}$
- Bayes rule: $P(X, Y) = P(X \mid Y)P(Y) = P(Y \mid X)P(X)$
- Independent variables X, Y: $P(X, Y) = P(X)P(Y)$
- The difference between *true* and *estimated* probability distributions

Notation and Definitions

- $\mathscr{V}$ is a finite set of (discrete) symbols (e.g., words or characters); $V = |\mathscr{V}|$

- $\mathscr{V}*$ is the (infinite) set of sequences of symbols from $\mathscr{V}$

- In language modeling, we imagine a sequence of random variables $X = \langle x_1, x_2, \ldots, x_n \rangle$ that continues until $x_n =$ "[EOS]"

- $\mathscr{V}^+$ is the (infinite) set of sequences of $\mathscr{V}$ symbols, with the last token $x_n =$ "[EOS]"

- LM problem: Estimate the probability of a sequence $P(X)$, $X \in \mathscr{V}^+$

Language Modeling Problem

- Input: training data a sequence $X = \langle x_1, x_2, \ldots, x_n \rangle \in \mathscr{V}^+$
  - Sometimes it's useful to consider a collection of training sentences, each in $\mathscr{V}^+$, but it complicates notation.
- Output: $P : \mathscr{V}^+ \rightarrow \mathbb{R}$

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

The big problem: How do we predict

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

?!?

What Can we Do w/ LMs?

- Score sentences, e.g., $P(X =$ "Jane went to the store"$)$:

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, i.e., [EOS]:
  calculate probability $P($Next Word | Context$)$
  sample a new word from the probability distribution

N-gram Language Models

Review: Count-based Unigram Model

- **Independence assumption:** $P(x_i|x_1, \ldots, x_{i-1}) \approx P(x_i)$

- **Maximum-likelihood estimation (MLE): counting how likely each word appearing in a corpus**

$$P_{\mathrm{MLE}}(x_i) = \frac{c_{\mathrm{train}}(x_i)}{\sum_{\tilde{x}} c_{\mathrm{train}}(\tilde{x})}$$

- **Interpolation w/ UNK model:**

$$P(x_i) = (1 - \lambda_{\mathrm{unk}}) * P_{\mathrm{MLE}}(x_i) + \lambda_{\mathrm{unk}} * P_{\mathrm{unk}}(x_i)$$

From Unigram to Bigram LM

- Next word prediction only depends on the previous word.

$$P(x_i|x_{i-n+1},\ldots,x_1) \approx P(x_i|x_{i-1})$$

- Given a training corpus of 3 sentences:

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

- Compute the bigram probability by counting (MLE):

$P(\texttt{I}|\texttt{<s>}) = \frac{2}{3} = .67$ $\quad P(\texttt{Sam}|\texttt{<s>}) = \frac{1}{3} = .33$ $\quad P(\texttt{am}|\texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>}|\texttt{Sam}) = \frac{1}{2} = 0.5$ $\quad P(\texttt{Sam}|\texttt{am}) = \frac{1}{2} = .5$ $\quad P(\texttt{do}|\texttt{I}) = \frac{1}{3} = .33$

- Probability of a sentence is just the product of all bigram probabilities in this sentence.

Higher-order *n*-gram Models

- Limit context length to *n*

$$P(x_i|x_{i-n+1},\ldots,x_1) \approx P(x_i|x_{i-n+1},...,x_{i-1})$$

- Maximum likelihood estimation: count, and divide

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

$$\text{P(example | this is an)} = \frac{\text{c(this is an example)}}{\text{c(this is an)}}$$

- Add smoothing by linear interpolation with (n-1)-gram LM, to deal with zero counts:

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

Smoothing for words in unknown context

- Add-one smoothing, i.e., every word adds 1 count.

$$P(x_i) = \frac{\text{count}(x_i)}{N}$$

$$P(x_i) = \frac{\text{count}(x_i) + 1}{N + V}$$

- Add smoothing by linear interpolation with (n-1)-gram LM, to deal with zero counts:

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

More Smoothing Methods
(e.g. Goodman 1998)

- **Additive/Dirichlet:**

fallback distribution

$$P(x_i \mid x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i) + \boxed{\alpha} \boxed{P(x_i \mid x_{i-n+2}, \ldots, x_{i-1})}}{c(x_{i-n+1}, \ldots, x_{i-1}) + \boxed{\alpha}}$$

interpolation hyperparameter

- **Discounting:**

discount hyperparameter

$$P(x_i|x_{i-n+1}, \ldots, x_{i-1}) := \frac{c(x_{i-n+1}, \ldots, x_i) - \boxed{d} + \boxed{\alpha} P(x_i|x_{i-n+2}, \ldots, x_{i-1})}{c(x_{i-n+1}, \ldots, x_{i-1})}$$

interpolation calculated by sum of discounts $\boxed{\alpha} = \sum_{\{\tilde{x}; c(x_{i-n+1}, \ldots, \tilde{x}) > 0\}} d$

- **Kneser-Ney:** discounting w/ modification of the lower-order distribution

Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. 1998.

Problems and Solutions?

- Cannot share strength among **similar words**

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

→ solution: class based language models

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

Dr. Jane Smith	Dr. Gertrude Smith

→ solution: skip-gram language models

- 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

→ solution: cache, trigger, topic, syntactic models, etc.

When to Use n-gram Models?

- Neural language models (next) achieve better performance, but
- n-gram models are extremely fast to estimate/apply
- n-gram models can be better at modeling low-frequency phenomena
- **Toolkit:** kenlm

https://github.com/kpu/kenlm

An Alternative:
Featurized Log-Linear Models

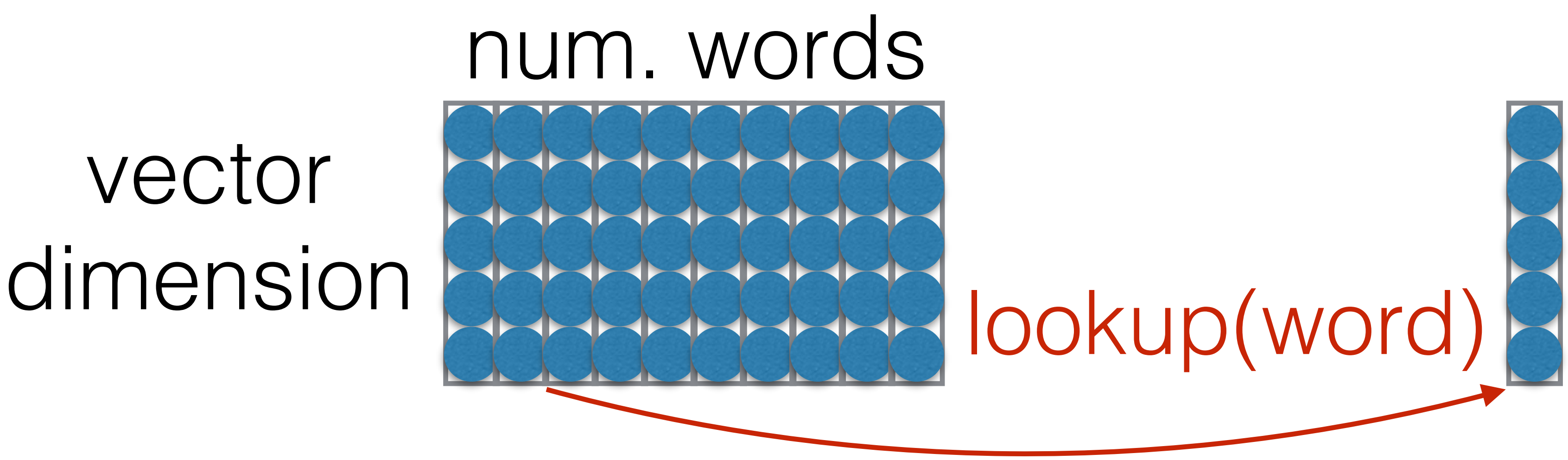
(Rosenfeld 1996)

An Alternative: Featurized Models

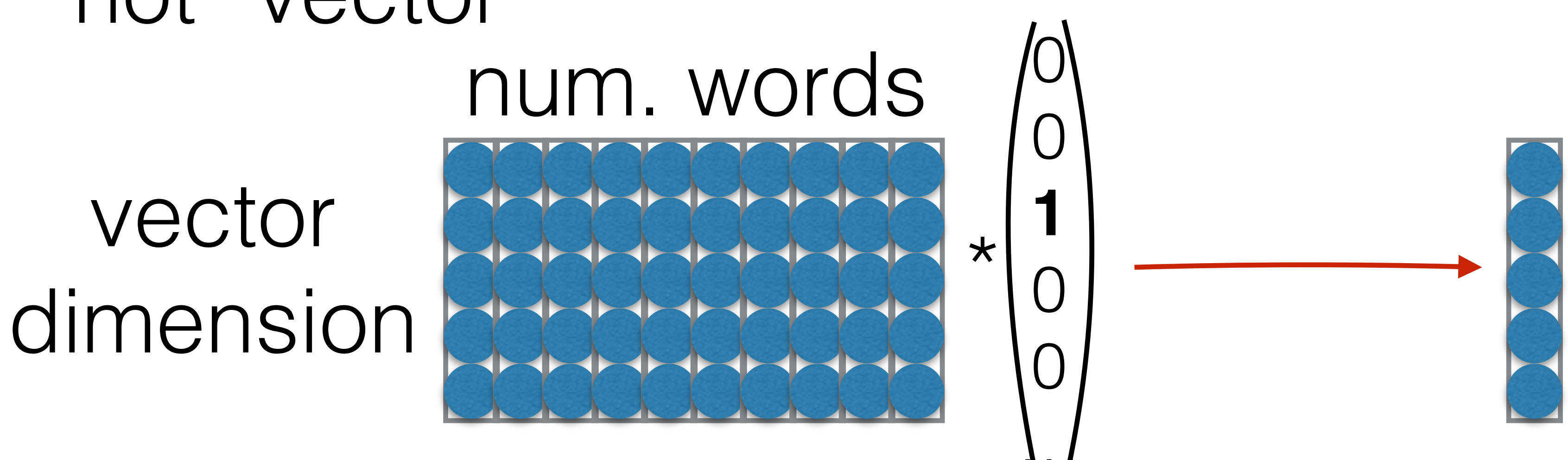
- Calculate features of the context
- Based on the features, calculate probabilities
- Optimize feature weights using gradient descent, etc.

A Note: “Lookup”

- Lookup can be viewed as “grabbing” a single vector from a big matrix of word embeddings



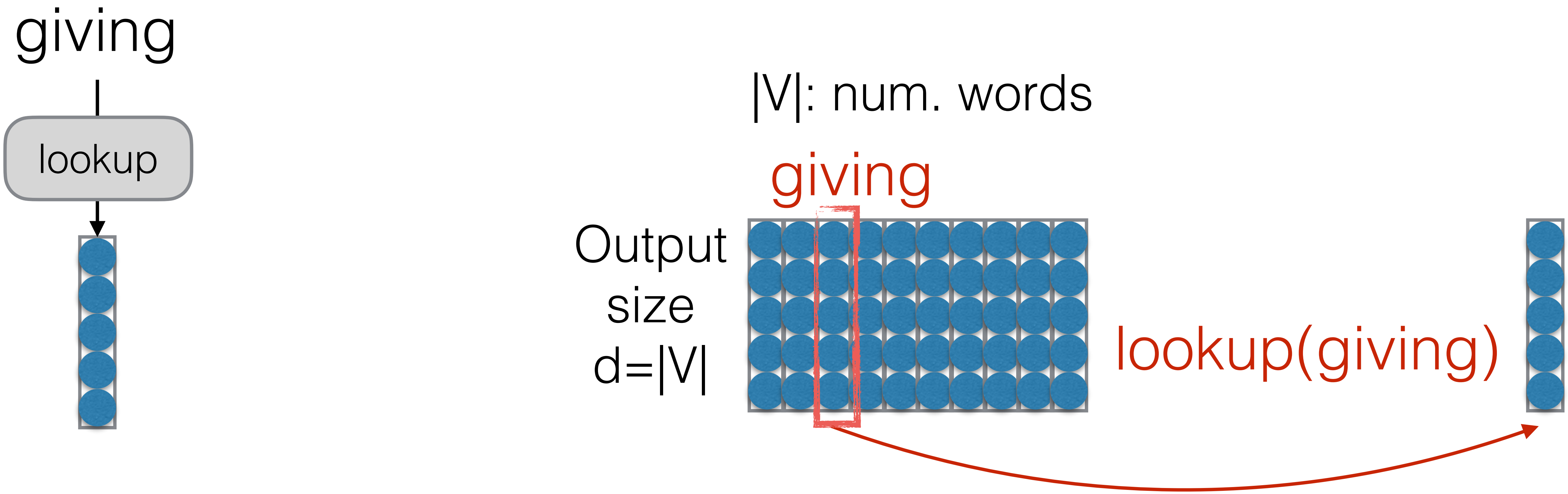
- Similarly, can be viewed as multiplying by a “one-hot” vector



- Former tends to be faster

An Alternative: Featurized Models

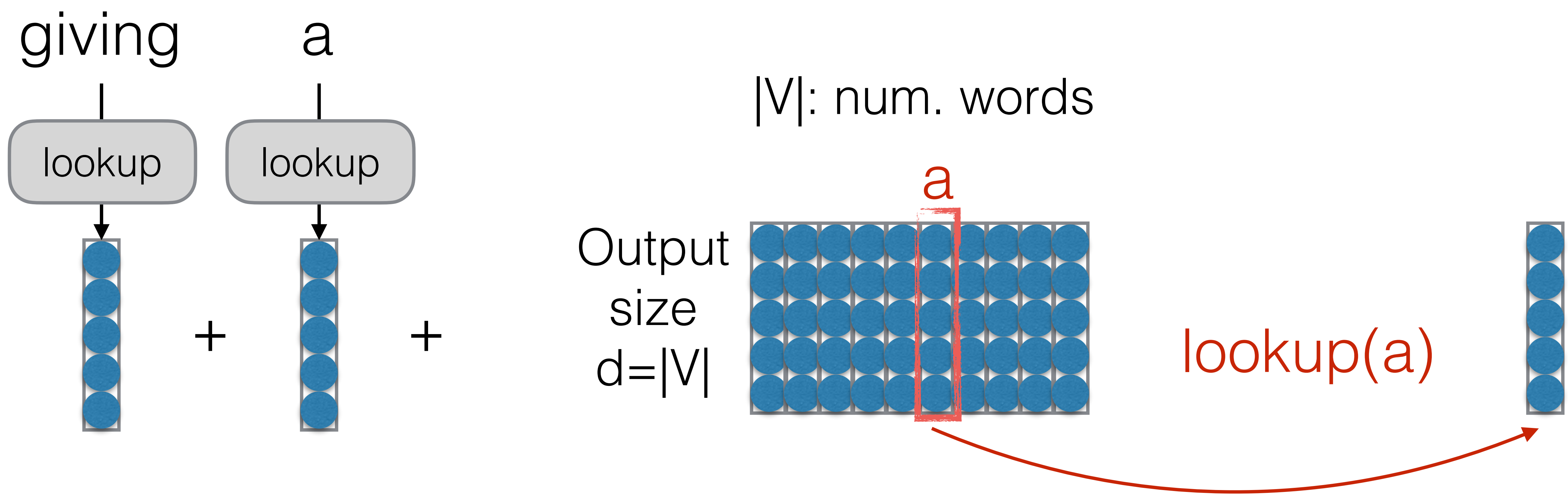
- Assume that we aim to learn a feature matrix W_0 where each column corresponds to a feature vector for each word.



- The word vector learns the similarity (coexistence) between the selected word (i.e., “giving”) and the other words, i.e., the likelihood of the next word coexisting with “giving” in the context

An Alternative: Featurized Models

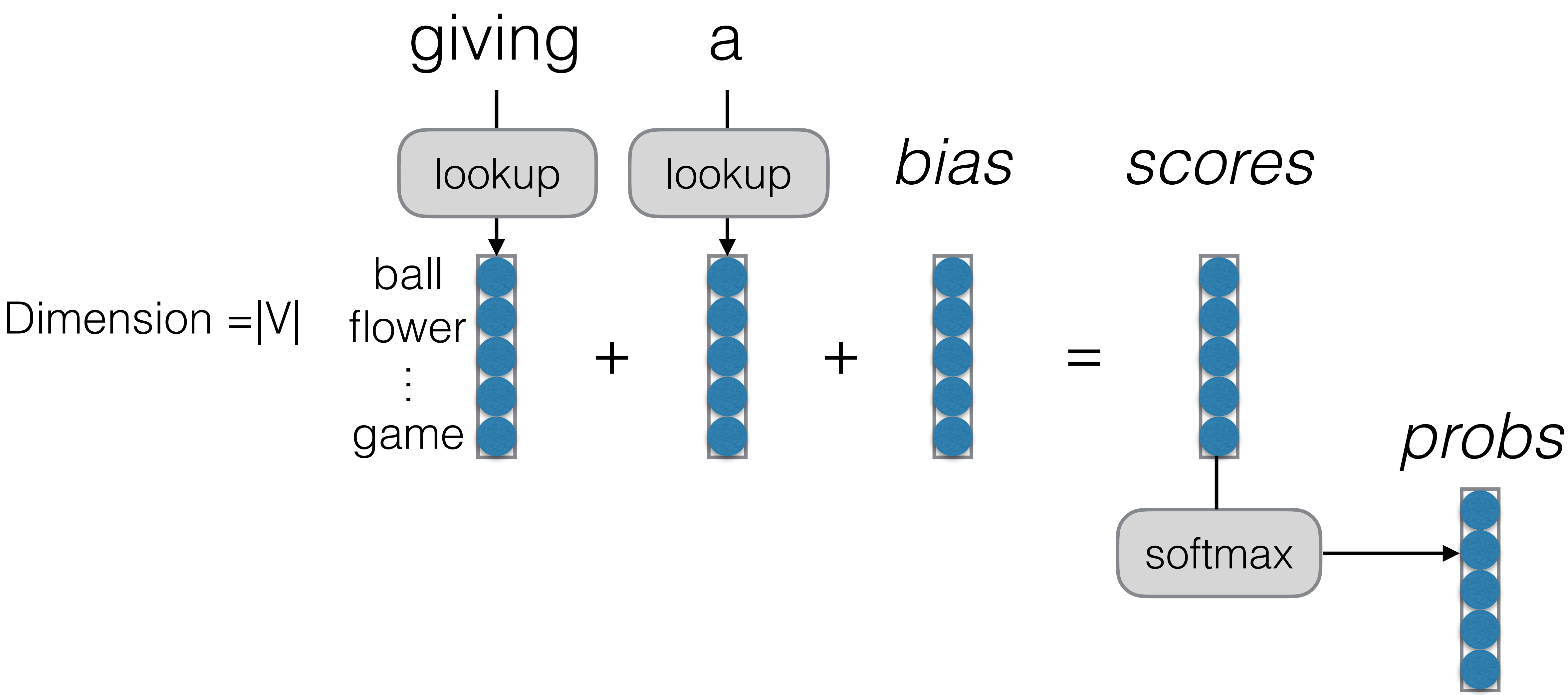
- Assume that we aim to learn a feature matrix W_0 where each column corresponds to a feature vector for each word.



- The word vector learns the similarity (coexistence) between the selected word (i.e., “giving”) and the other words, i.e., the likelihood of the next word coexisting with “giving” in the context

An Alternative: Featurized Models

- Combine with the bias vector (model parameter), compute the probability over the output vocabulary V
- Each word vector has the size of output vocabulary, where each element represents **the probability of the output word given the current word in the context**



- Feature weights optimized by SGD, etc.
- What are similarities/differences w/ BOW classifier?

Example:

Previous words: “giving a"

$$
\begin{matrix} \text{a} \\ \text{the} \\ \text{talk} \\ \text{gift} \\ \text{hat} \\ \ldots \end{matrix}
\qquad
b=\begin{pmatrix} 3.0 \\ 2.5 \\ -0.2 \\ 0.1 \\ 1.2 \\ \ldots \end{pmatrix}
\quad
w_a=\begin{pmatrix} -6.0 \\ -5.1 \\ 0.2 \\ 0.1 \\ 0.5 \\ \ldots \end{pmatrix}
\quad
w_{giving}=\begin{pmatrix} -0.2 \\ -0.3 \\ 1.0 \\ 2.0 \\ -1.2 \\ \ldots \end{pmatrix}
\quad
s=\begin{pmatrix} -3.2 \\ -2.9 \\ 1.0 \\ 2.2 \\ 0.6 \\ \ldots \end{pmatrix}
$$

Vocabulary: 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

Reminder: Training Algorithm

- Calculate the **gradient of the loss function** with respect to the parameters

$$\frac{\partial \mathcal{L}_{\text{train}}(\theta)}{\partial \theta}$$

  - How? Use the chain rule / back-propagation. More in a second

- **Update** to move in a direction that decreases the loss

$$\theta \leftarrow \theta - \alpha \frac{\partial \mathcal{L}_{\text{train}}(\theta)}{\partial \theta}$$

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 😞

Neural Language Model

—Beyond Linear Models

Linear Models can't Learn Feature Combinations

students take tests→ **high** teachers take tests → **low**

students write tests → **low** teachers write tests → **high**

- These can't be expressed by linear features

- What can we do?
    - Remember combinations as features (individual scores for "students take", "teachers write")
      → Feature space explosion!
    - Neural networks!

Neural Language Models

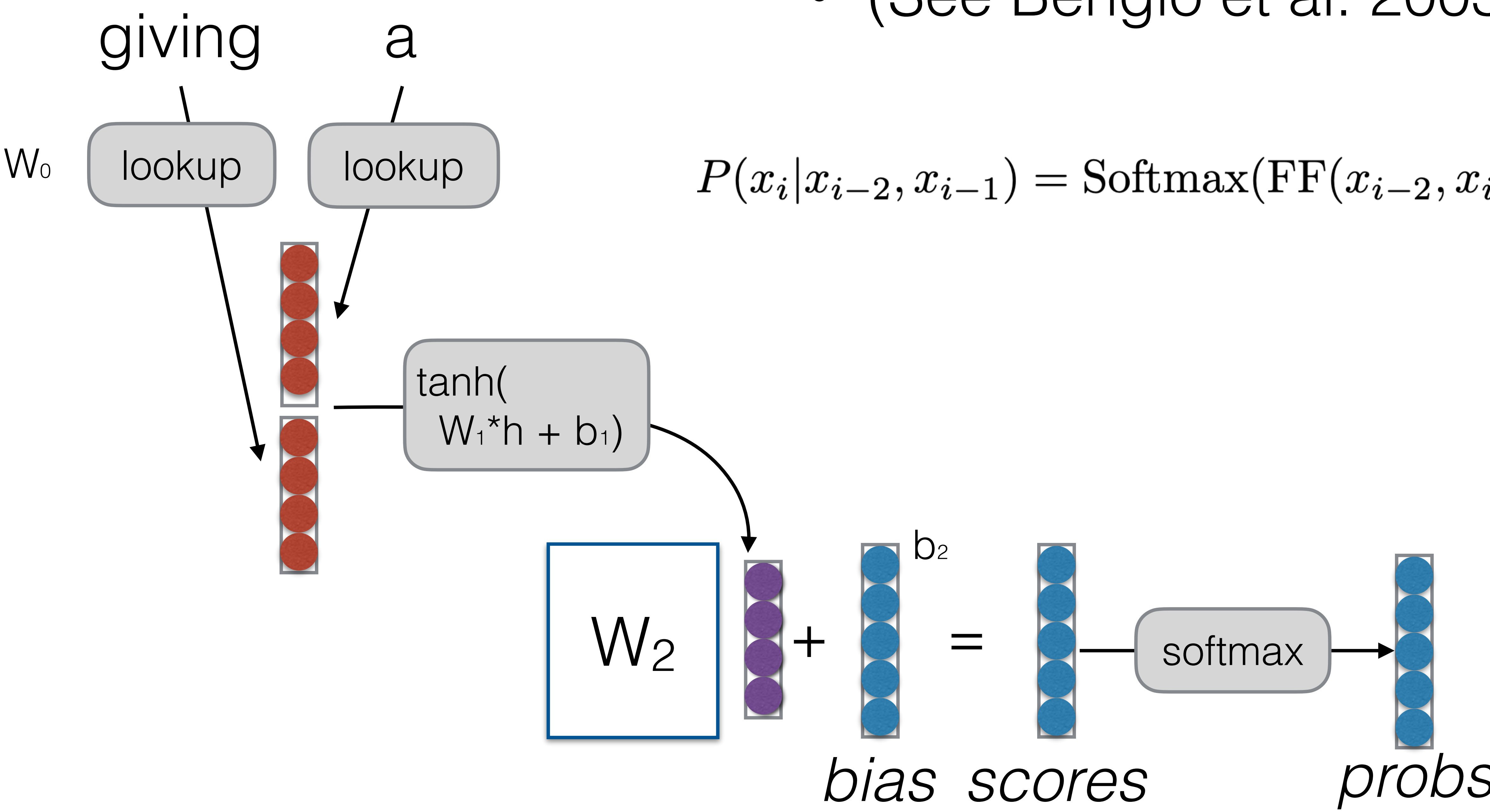
- Convert the word prediction problem to discriminative text classification
- The input is the n-1 previous words (context)
- The output is a word in the vocabulary (V-class classification)

$$P(x_i|x_{i-n+1}, \ldots, x_{i-1}) = \mathrm{Softmax}(\mathrm{NeuralNet}(x_{i-n+1}, \ldots, x_{i-1}))$$

Feed-forward Neural Language Models

- (See Bengio et al. 2003)

$$P(x_i|x_{i-2}, x_{i-1}) = \text{Softmax}(\text{FF}(x_{i-2}, x_{i-1}))$$



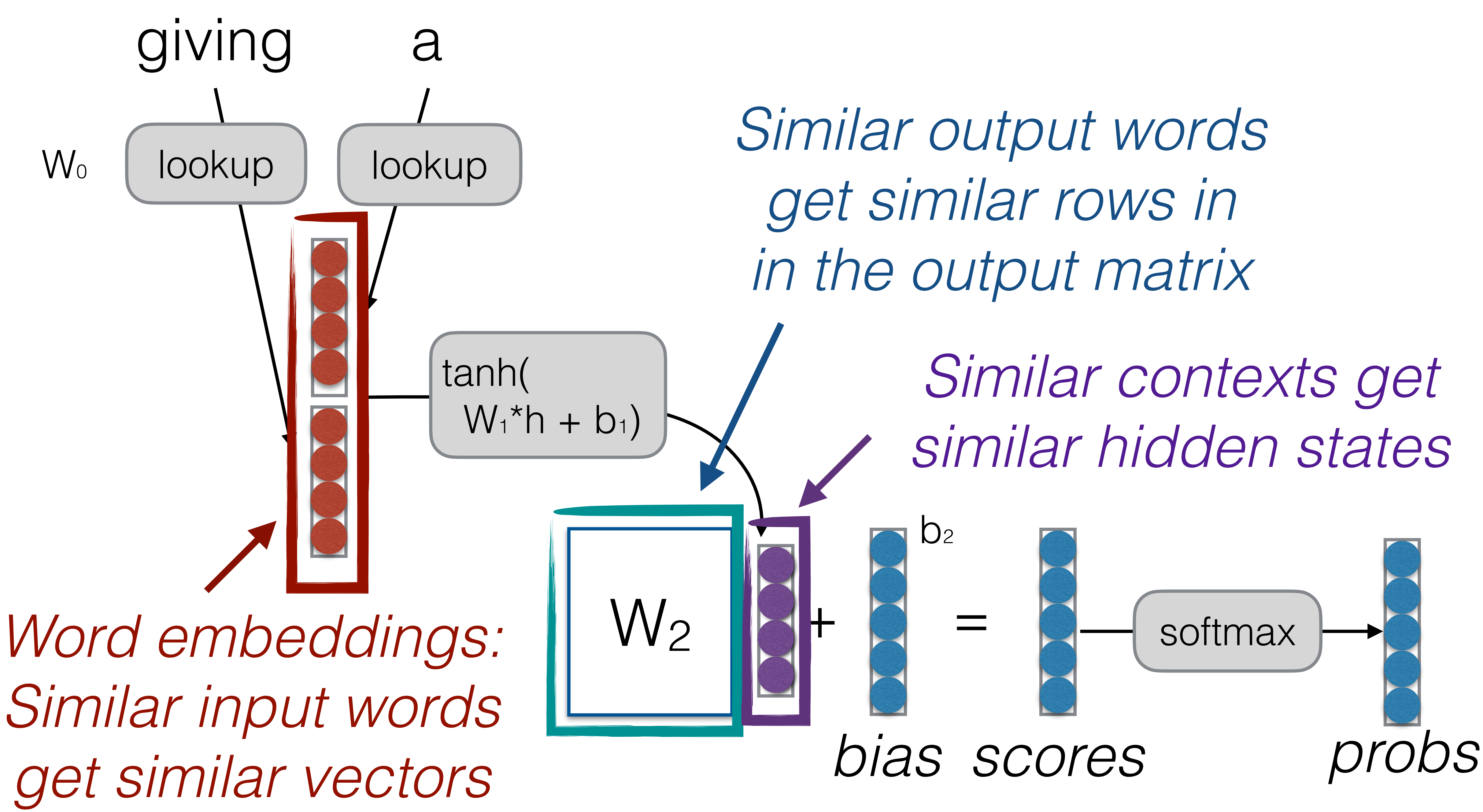
Example of Combination Features

- Word embeddings capture features of words
  - e.g. feature 1 indicates verbs, feature 2 indicates determiners
- A row in the weight matrix (together with the bias) can capture particular *combinations* of these features
  - e.g. the 34th row in the weight matrix looks at feature 1 in the second-to-previous word, and feature 2 in the previous word

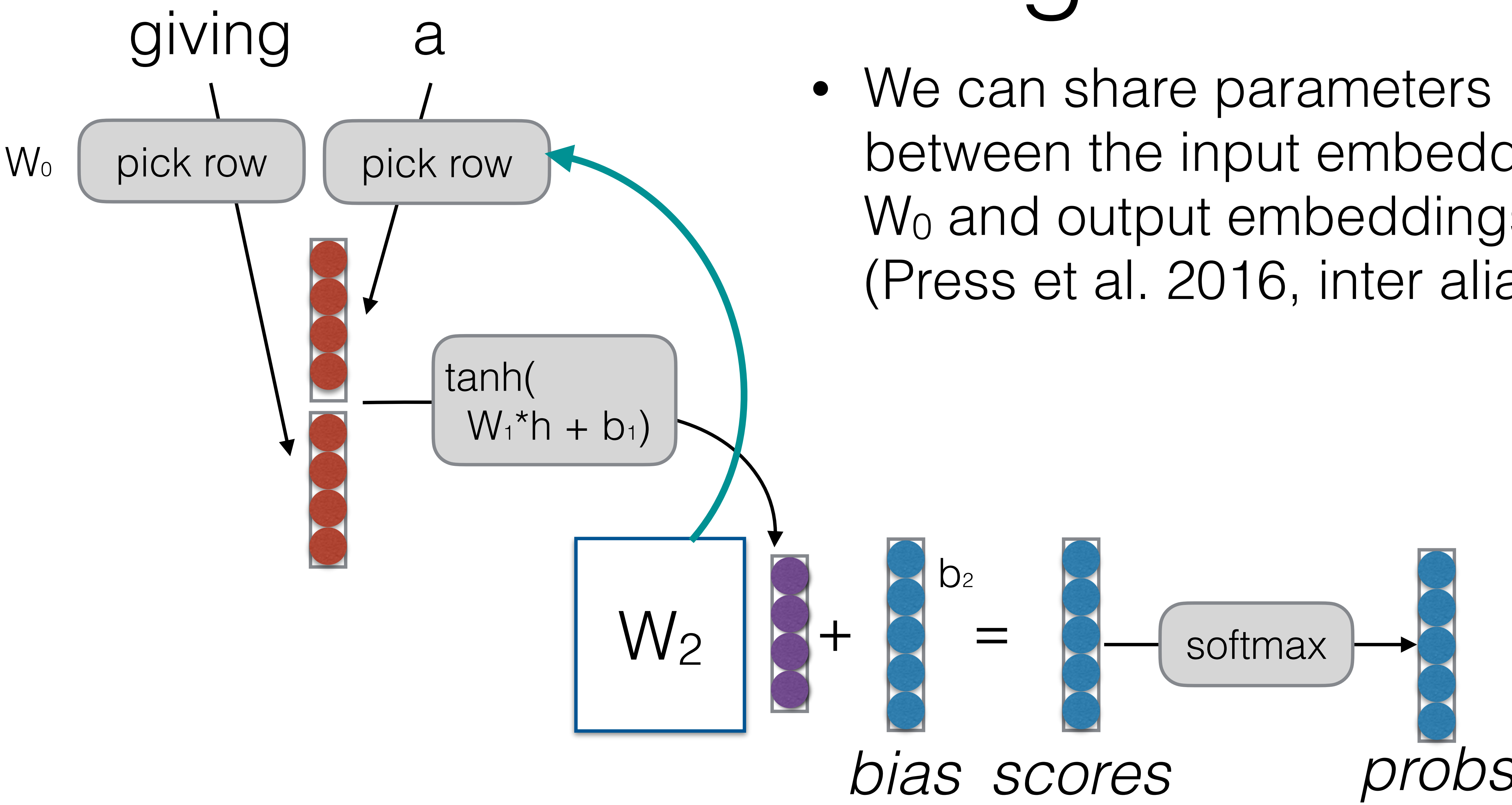
$$
\begin{array}{rc}
\text{giving} & \begin{bmatrix} 1.2 \\ -0.1 \\ 0.7 \\ -2.1 \\ 0.5 \end{bmatrix} \\
\text{a} & \begin{bmatrix} -0.3 \\ 2.0 \\ 0.6 \\ -0.8 \\ -0.4 \end{bmatrix}
\end{array}
\;*\;
\overset{\boldsymbol{w}_{34}}{\begin{bmatrix} 1.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1.3 \\ 0 \\ 0 \\ 0 \end{bmatrix}}
\;+\; \overset{b_{34}}{-2} \;=
$$

positive number if the previous word is a determiner and second-to-previous word is a verb

Where is Strength Shared?



Tying Input/Output Embeddings



- We can share parameters between the input embeddings W_0 and output embeddings W_2 (Press et al. 2016, inter alia)

Want to try? Delete the input embeddings W_0, and instead pick a row from the output matrix W_2.

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 😞

Many Other Potential Designs!

- Neural networks allow design of arbitrarily complex functions!

- In future classes, we can replace Feedforward models by more powerful sequential NNs:

    - **Recurrent neural network LMs**

    - **Transformer LMs**

LM Evaluation

Evaluation of LMs

- **Log-likelihood:**

$$LL(\mathcal{D}_{\text{test}}) = \sum_{X \in \mathcal{D}_{\text{test}}} \log P(X)$$

- **Per-word Log Likelihood:**

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

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

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

- **Perplexity:**

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

Unknown Words

- Necessity for UNK words
  - We won’t have all the words in the world in training data
  - Larger vocabularies require more memory and computation time
- Common ways:
  - Limit vocabulary by frequency threshold (usually UNK <= 1) or rank threshold
  - Model characters or subwords

Evaluation and Vocabulary

- **Important:** the vocabulary must be the same over models you compare

- Or more accurately, all models must be able to generate the test set (it's OK if they can generate *more* than the test set, but not less)

    - e.g. Comparing a character-based model to a word-based model is fair, but not vice-versa

LM Problem Definition

Count-based LMs

Evaluating LMs

Log-linear LMs

Neural Net Basics

Feed-forward NN LMs

Questions?

Quiz 1: https://forms.gle/bV72hMZy3qd6UbKr7
Survey: https://forms.gle/3RsuRYqi1BdakTyJA