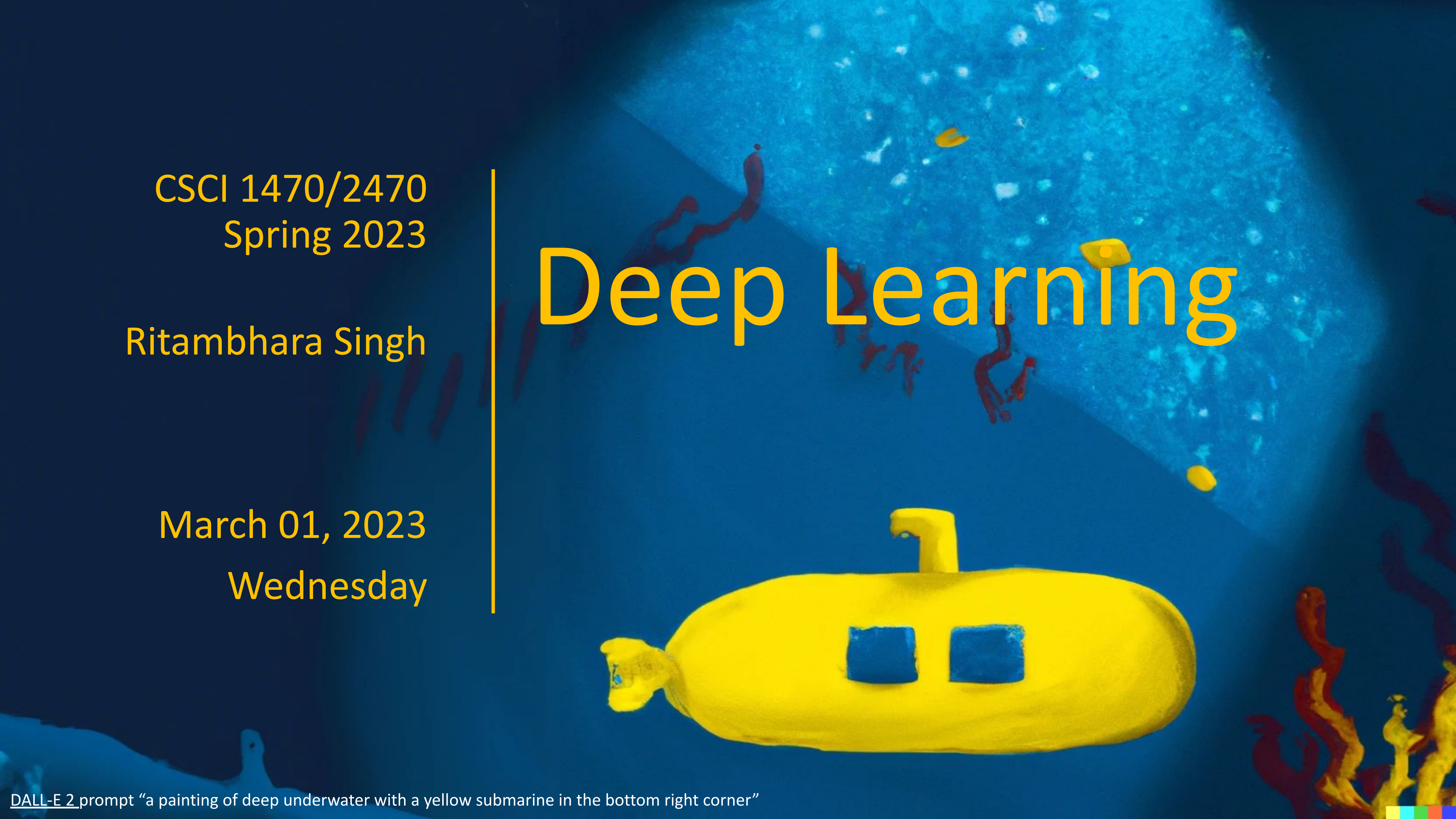


Deep Learning

CSCI 1470/2470
Spring 2023

Ritambhara Singh

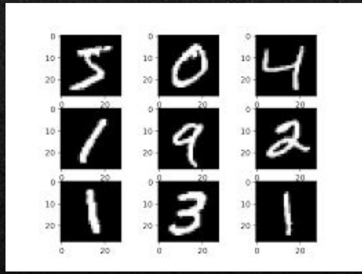
March 01, 2023
Wednesday

DALL-E 2 prompt “a painting of deep underwater with a yellow submarine in the bottom right corner”

Roadmap

Machine Learning Concepts

Perceptron

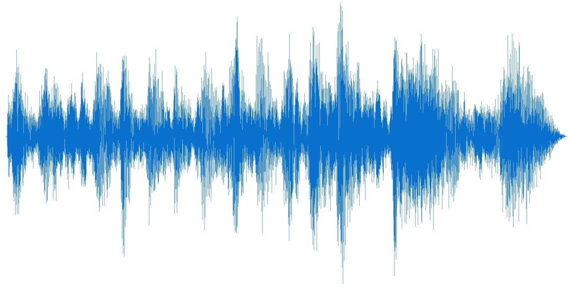


Fully Connected Neural Networks

Convolutional Neural Networks

New data type: sequences

- Audio



- DNA

- Stock market

- Weather

What is the data property here that we could leverage?

Natural Language

“language that has developed naturally in use”

Natural Language

"language that has developed naturally in use"

Compare to *constructed* or *formal* language

- code: `for i in range(50):`
- math: `52 + 94 = 147`
- logic: `A ^ B -> C` (if A and B, then C)

Natural Language

In this class: **sequence of *words***

"They went to the grocery store and bought bread, peanut butter, and jam."

Natural Language: Prediction tasks?

Example of prediction?

Input: X

I do not want sour cream in my burrito

Function: f

Output: Y

No quiero crema agrea en mi burrito

Natural Language: Prediction tasks?

Example of classification?

Input: X

“The story telling was erratic and, at times, slow”

“Loved the diverse cast of this movie”

Function: f

Output: Y

“Good review?”

Natural Language: Prediction tasks?

Example of prediction?

"They went to the grocery store and **bought… bread?**

milk?

rock?

Generating artificial sentences: Here each word is a discrete unit; predicting the next part of the sequence means predicting words

Language models

Definition: Probability distribution over strings in a language.

Exponentially-many strings means each string has very low probability

Relative probabilities are meaningful:

P("*they went to the store*") >> P("*butter dancing rock*")

Language models logic: leverage sentence structure

`P(any sequence)` is determined by `P(the words in the sequence).`

Said differently, we can represent a sequence as $w_1, w_2, \dots w_n$, and

$$P(w_1, w_2, \dots w_n) = P(w_1) * P(w_2|w_1) * P(w_3|w_1, w_2) * \cdots P(w_n|w_1 \dots w_{n-1})$$

P(“*they went to the store*”) = P(“*they*”)*P(“*went*”|“*they*”)*P(“*to*”|“*they went* ”)* ..

“The probability of a sentence is the product of the probabilities of each word given the previous words”

This is an application of the ***chain rule for probabilities***

Language models: weird & cool!

Model trained on the King James Bible, Structure and Interpretation of Computer Programs, and some of Eric S. Raymond's writings:

- ***The righteous shall inherit the land, and leave it for an inheritance unto the children of Gad according to the number of steps that is linear in b.***
- ***And this I pray, that your love may abound yet more and more like a controlled use of shared memory.***

Any questions?

(King James Programming)

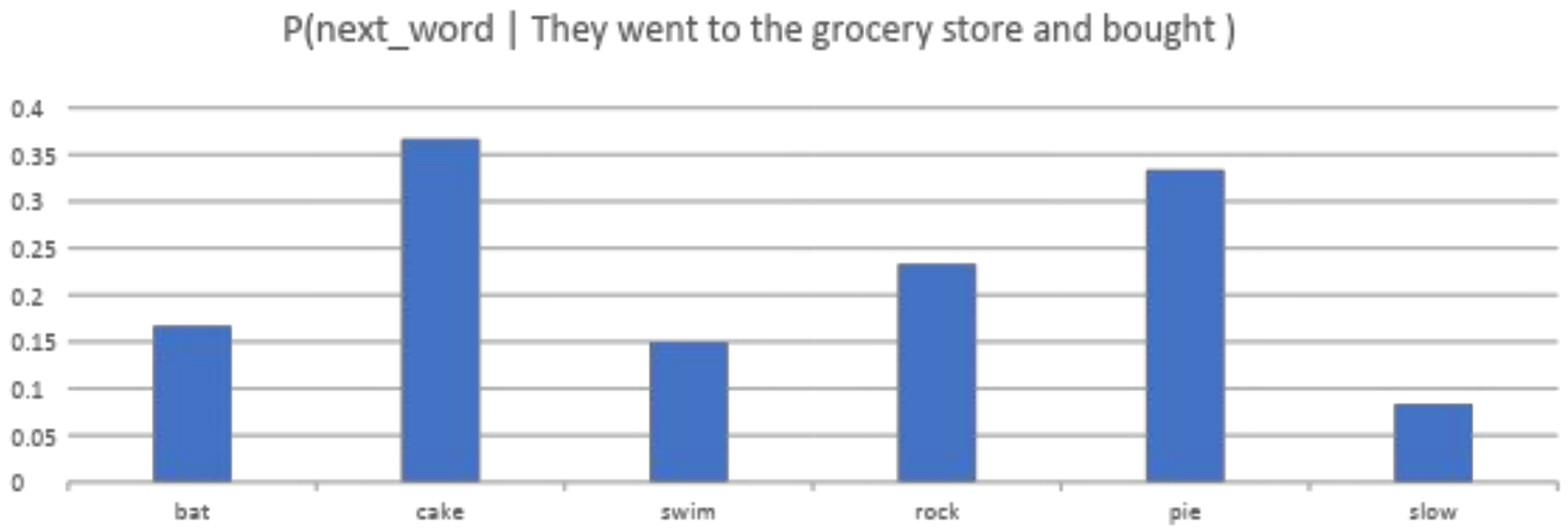
https://kingjamesprogramming.tumblr.com/

Language models: the math

But first, how do we represent sentence?

At each step, we look at a probability distribution for what the *next* word might be.

They went to the grocery store and bought ..



Natural language: tokenization

“They went to the grocery store and bought bread, peanut butter, and jam.”

[“they”, “went”, “to”, “the”, “grocery”, “store”, “and”, “bought”, “bread”, “peanut”, “butter”, “and”, “jam”]

Natural language: tokenization

"They went to the grocery store and bought bread, peanut butter, and jam."

- Consistent casing
- Strip punctuation
- One word is one token
- Split on spaces

```
[“they”, “went”, “to”, “the”,
“grocery”, “store”, “and”,
“bought”, “bread”, “peanut”,
“butter”, “and”, “jam”]
```

Aside: Tokenization itself can be challenging...

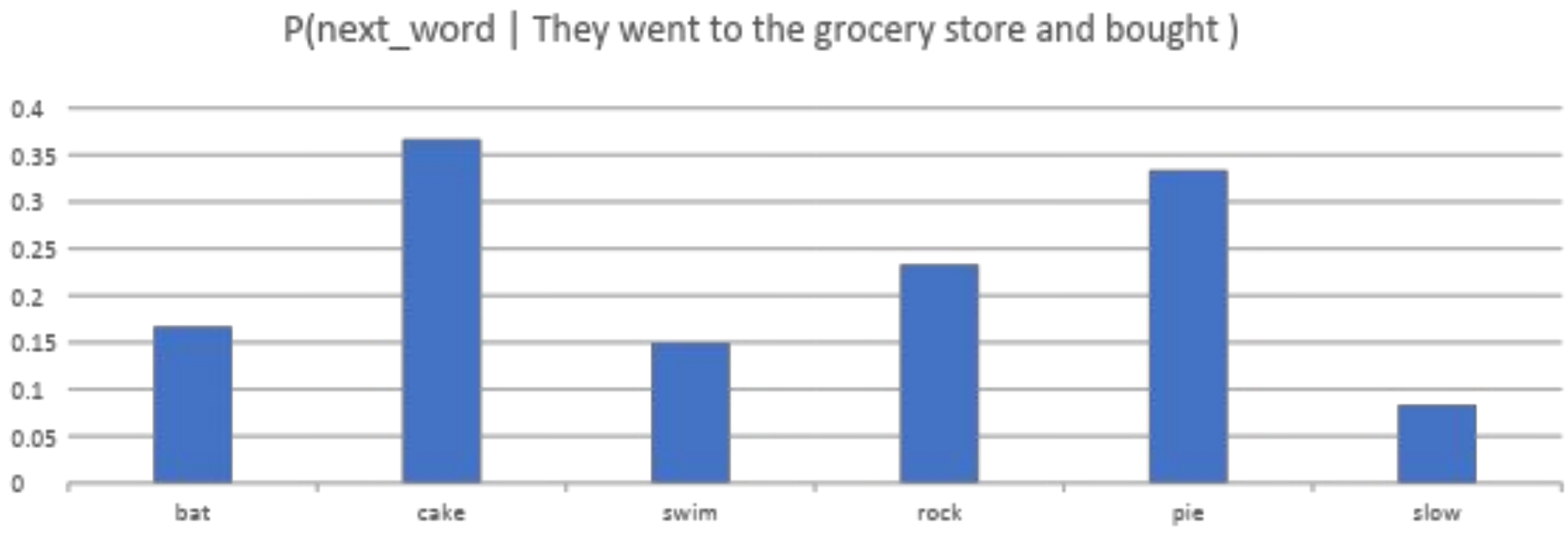
- A lot easier in English than other languages (e.g. Chinese)
  - Chinese is character-based; words & phrases have different character lengths
  - No spaces

Language models: the math

How do we know which words to calculate probabilities for?

At each step, we look at a probability distribution for what the *next* word might be.

They went to the grocery store and bought ..



Vocabularies: Defining a finite set of words

Vocabularies: the set of all words "known" to the model

Why?

- We need a finite set of words in order to define a discrete distribution over it.

How?

- Choose a hyperparameter **vocab_size** for how many words the model should know
- Keep only the **vocab_size** with most frequent words – replace everything else with "**UNK**"

Vocabularies: how

- Original sentence:
  - *“They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk.”*

Vocabularies: how

- Original sentence:
  - ***“They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk.”***
- Tokenized:
  - **`[“they”, “galloped”, “to”, “the”, “ratty”, “for”, “dinner”, “and”, “ate”, “exactly”, “seventy-three”, “waffle”, “fries”, “and”, “chocolate”, “peamilk”]`**

Vocabularies: how

- Original sentence:
  - *"They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk."*
- Tokenized:
  - `["they", "galloped", "to", "the", "ratty", "for", "dinner", "and", "ate", "exactly", "seventy-three", "waffle", "fries", "and", "chocolate", "peamilk"]`

Vocabularies: how

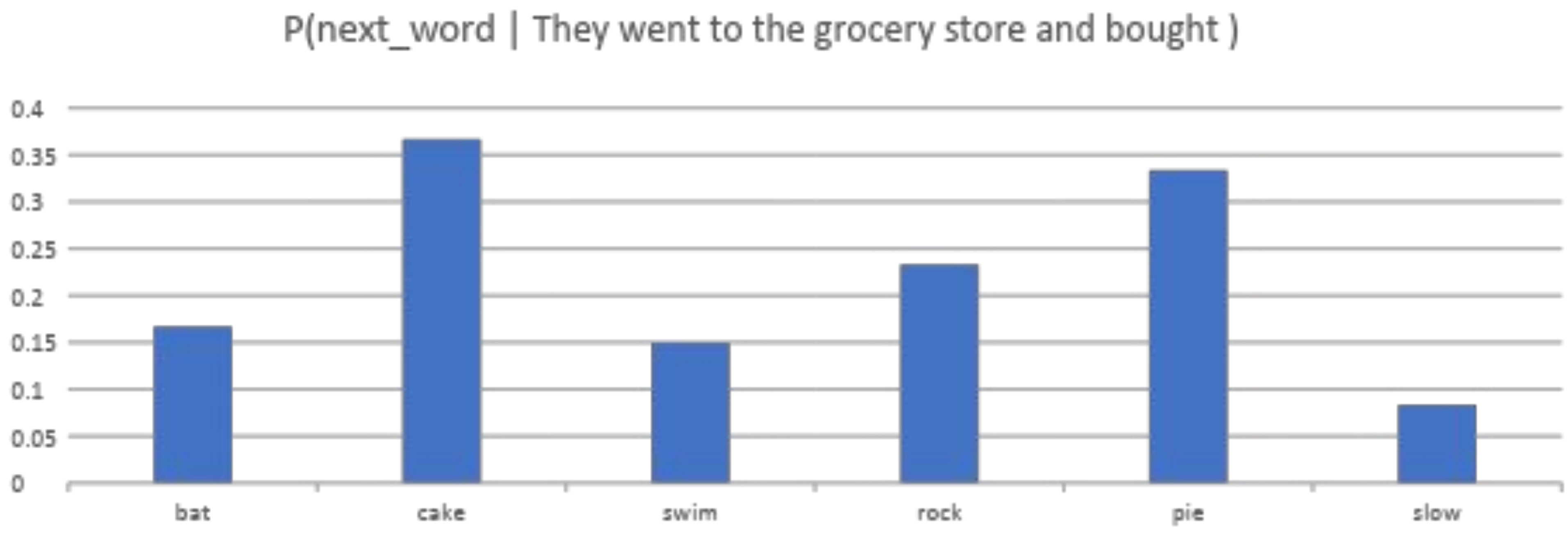
- Original sentence:
  - *"They galloped to the Ratty for dinner, and ate exactly seventy-three waffle fries and chocolate peamilk."*
- Tokenized:
  - `["they", "galloped", "to", "the", "ratty", "for", "dinner", "and", "ate", "exactly", "seventy-three", "waffle", "fries", "and", "chocolate", "peamilk"]`
- UNKed:
  - `["they", "UNK", "to", "the", "UNK", "for", "dinner", "and", "ate", "exactly", "UNK", "waffle", "fries", "and", "chocolate", "UNK"]`

Language models: the math

How to calculate the probability for words in our vocabulary?

At each step, we look at a probability distribution for what the *next* word might be.

They went to the grocery store and bought ..



LM implementation: counting

- Goal: predict next word given a preceding sequence
  - $P(\boldsymbol{word_n} | \, word_1, word_2, \dots word_{n-1}) = \frac{Count(word_1, word_2, \dots word_{n-1}, \boldsymbol{word_n})}{Count(word_1, word_2, \dots word_{n-1})}$

LM implementation: counting

- Goal: predict next word given a preceding sequence
  - $P(\boldsymbol{word_n} | \, word_1, word_2, \ldots word_{n-1}) = \frac{Count(word_1, word_2, \ldots word_{n-1}, \boldsymbol{word_n})}{Count(word_1, word_2, \ldots word_{n-1})}$
- Example task: predict the next word
  - ***he danced*** ___

LM implementation: counting

- Goal: predict next word given a preceding sequence
  - $P(\boldsymbol{word_n} | \, word_1, word_2, \ldots word_{n-1}) = \frac{Count(word_1, word_2, \ldots word_{n-1}, \boldsymbol{word_n})}{Count(word_1, word_2, \ldots word_{n-1})}$
- Example task: predict the next word
  - ***he danced*** ___
- Strategy: iterate through all words in vocabulary, and calculate

  $\frac{Count(he\ danced\ <word>)}{Count(he\ danced)}$ for each word

LM implementation: counting

- Our training sentences were:
  - *"She danced happily"*
  - *"They sang beautifully"*
  - *"He danced energetically"*
  - *"He sang happily"*
  - *"She danced gracefully"*
- *"He danced _ _ _"*
- *"He danced happily"*

Has 0 probability

$$\frac{Count(he\ danced\ < word >)}{Count(he\ danced)}$$

Why doesn't this work?

This strategy depends on having instances of sentence prefixes.

LM implementation: N-gram counting

Improvement: **N-gram** model – only look at **N** words at a time

LM implementation: N-gram counting

Improvement: **N-gram** model – only look at **N** words at a time

(in this case, **bi**grams look at **2** words at a time)

- ***“She danced happily”***
- ***“They sang beautifully”***
- ***“He danced energetically”***
- ***“He sang happily”***
- ***“She danced gracefully”***

LM implementation: N-gram counting

Improvement: **N-gram** model – only look at **N** words at a time

(in this case, **bi**grams look at **2** words at a time)

- ***"danced happily"***
- ***"sang beautifully"***
- ***"danced energetically"***
- ***"sang happily"***
- ***"danced gracefully"***

"He danced <u>happily</u>" now has 1/3 probability!

But what if the answer was ***"He danced <u>beautifully</u>"*** ?

LM implementation

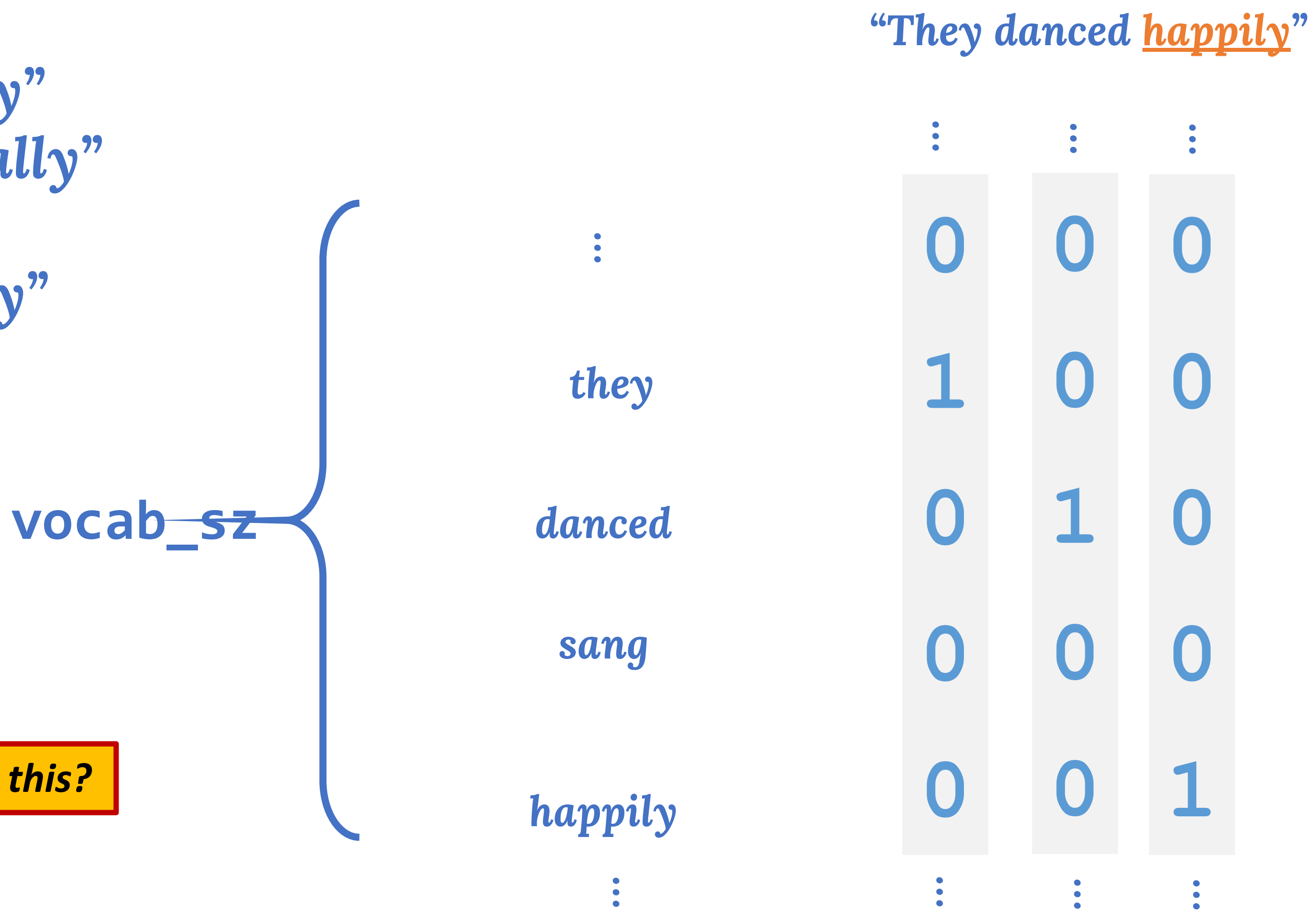
Problem: it's impossible for the training set to have *every possible valid* sequence of words!

Let's try to learn a better **numerical** representation

What is the simplest thing you can think of?

LM implementation: Simple approach

- *"She danced happily"*
- *"They sang beautifully"*
- *"He danced energetically"*
- *"He sang happily"*
- *"She danced gracefully"*



Any potential issues with this?

LM implementation

Problem: it's impossible for the training set to have *every possible valid* sequence of words!

Can we learn a better numerical representation **which associates *related* words with one another**?

Embedding matrix

vocab_sz

⋮						
they	2	0	1	3	0	4
danced	0	1	1	0	2	1
sang	0	0	2	0	1	3
happily	0	1	1	1	0	2
gleefully	4	0	0	1	1	0
⋮						

Embedding matrix

Any questions?

vocab_sz

⋮						
they	2	0	1	3	0	4
danced	0	1	1	0	2	1
sang	0	0	2	0	1	3
happily	0	1	1	1	0	2
gleefully	4	0	0	1	1	0
⋮						

Embedding matrix

$$\text{vocab\_sz}\left\{\overbrace{\begin{matrix} 2 & 0 & 1 & 3 & 0 & 4 \\ 0 & 1 & 1 & 0 & 2 & 1 \\ 0 & 0 & 2 & 0 & 1 & 3 \\ 0 & 1 & 1 & 1 & 0 & 2 \\ 4 & 0 & 0 & 1 & 1 & 0 \end{matrix}}^{\text{embedding\_sz}}\right.$$

- 2d matrix: **`vocab_sz x embedding_sz`**

Embedding matrix

$$\texttt{vocab\_sz}\left\{\overbrace{\begin{matrix}2&0&1&3&0&4\\0&1&1&0&2&1\\0&0&2&0&1&3\\0&1&1&1&0&2\\4&0&0&1&1&0\end{matrix}}^{\texttt{embedding\_sz}}\right.$$

- 2d matrix: **`vocab_sz` x `embedding_sz`**
- each word corresponds to an index, or word ID – hence the **`vocab_sz`** dimension

Embedding matrix

How to build this embedding matrix?

$$\texttt{vocab\_sz}\left\{\overbrace{\begin{matrix} 2 & 0 & 1 & 3 & 0 & 4 \\ 0 & 1 & 1 & 0 & 2 & 1 \\ 0 & 0 & 2 & 0 & 1 & 3 \\ 0 & 1 & 1 & 1 & 0 & 2 \\ 4 & 0 & 0 & 1 & 1 & 0 \end{matrix}}^{\texttt{embedding\_sz}}\right.$$

- 2d matrix: **`vocab_sz` x `embedding_sz`**
- each word corresponds to an index, or word ID – hence the **`vocab_sz`** dimension
- **`embedding_sz`** is a hyperparameter

LM implementation: deep learning

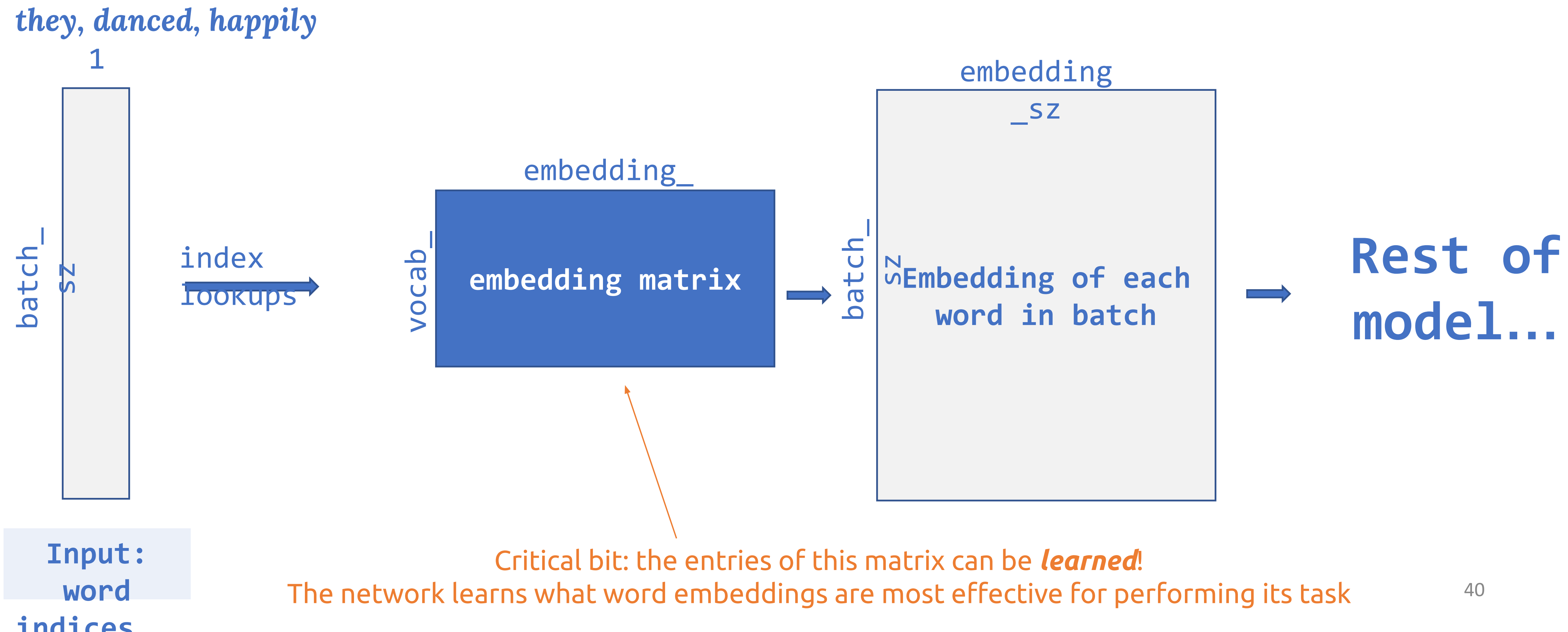
Deep learning helps solve this!

How?

We can learn an ***embedding matrix*** that associates *related* words with one another for solving a prediction task.

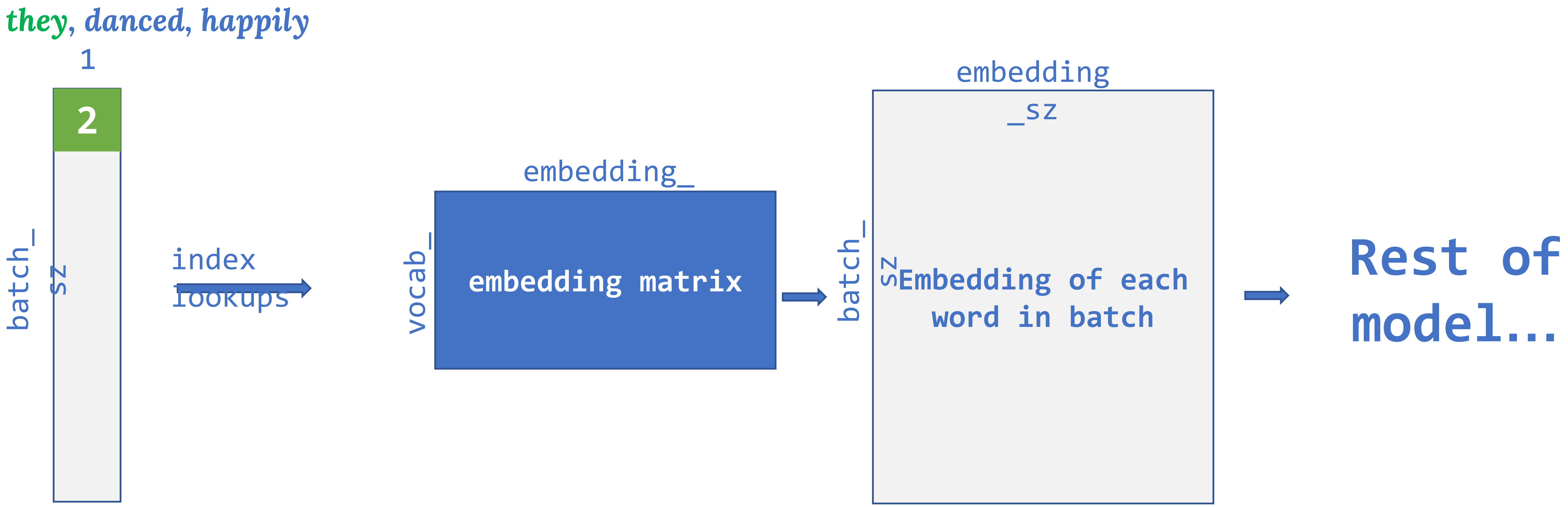
Using the Embedding Matrix in a Network

If you want to input a [batch of] words into a neural net, this is how:



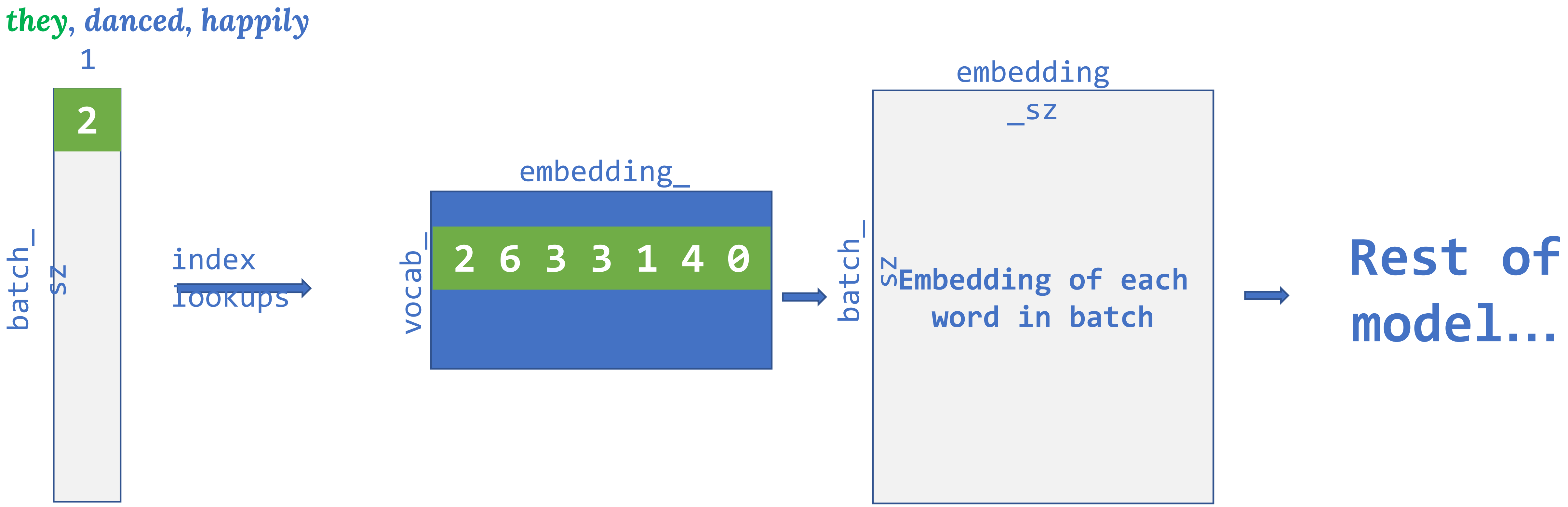
Using the Embedding Matrix in a Network

Let's look at the 0^{th} word in this batch; its ID in the vocab is 2.



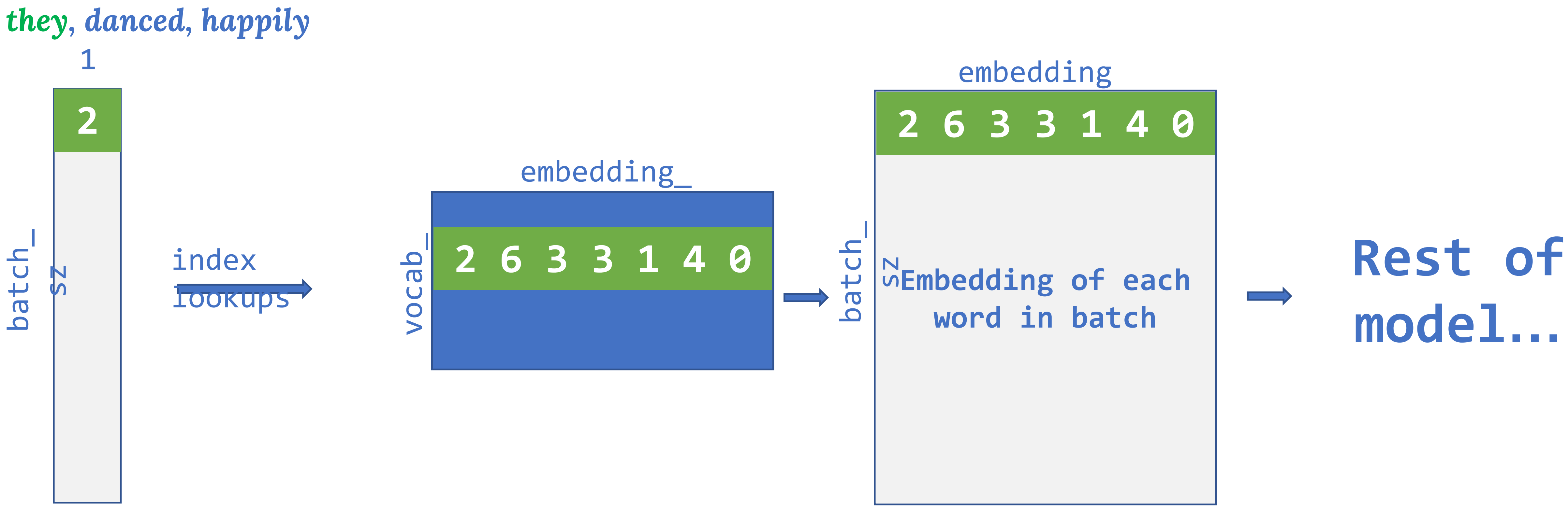
Using the Embedding Matrix in a Network

So we look at row 2 of the embedding matrix.



Using the Embedding Matrix in a Network

We can then pull out this embedding so we can use it in the rest of the model!

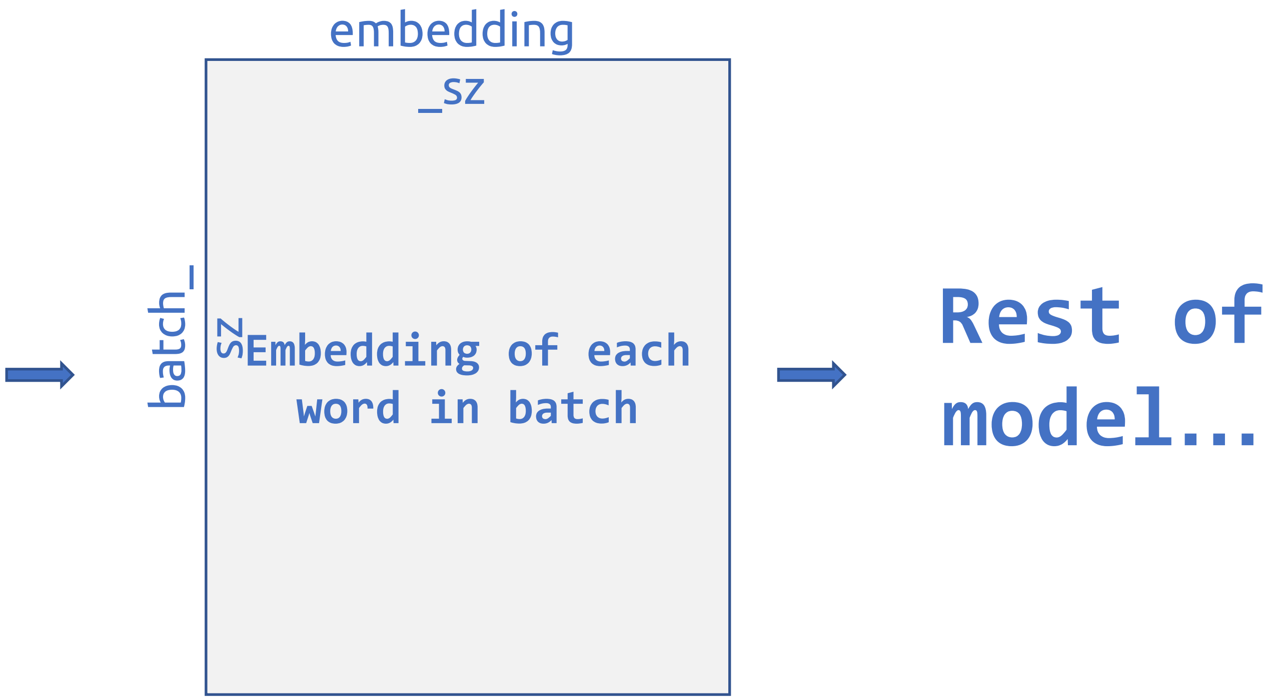


Using the Embedding Matrix in a Network

In tensorflow, we can use

`tf.nn.embedding_lookup`

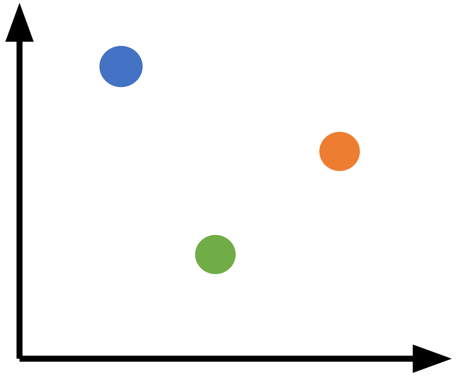
which takes in an embedding matrix and a list of indices, and returns the embedding corresponding to each index.



What does the embedding matrix represent?

- Each row in the matrix can be viewed as a vector in vector space

Example 2-D vector space:



Vocab size: 3

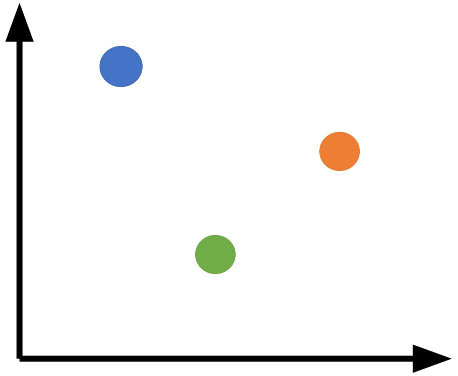
Embed size: 2

1	3
2	1
3	2

What does the embedding matrix represent?

- Each row in the matrix can be viewed as a vector in vector space
- “Embedding”: We’re **embedding** a non-Euclidian entity [a word] into Euclidian space

Example 2-D vector space:



Vocab size: 3

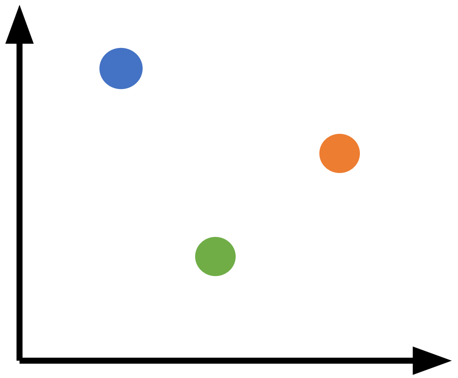
Embed size: 2

$$\begin{matrix} 1 & 3 \\ 2 & 1 \\ 3 & 2 \end{matrix}$$

What does the embedding matrix represent?

- Each row in the matrix can be viewed as a vector in vector space
- "Embedding": We're **embedding** a non-Euclidian entity [a word] into Euclidian space
- Each row represents the "embedding" for a single word

Example 2-D vector space:



Vocab size: 3

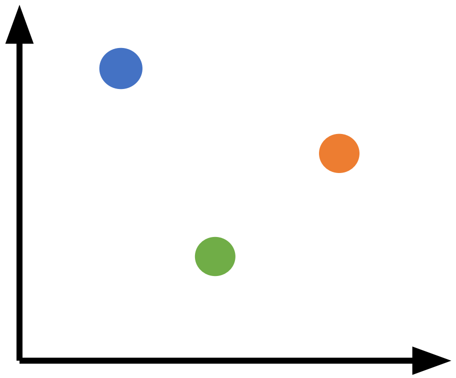
Embed size: 2

1	3
2	1
3	2

What does the embedding matrix represent?

- Each row in the matrix can be viewed as a vector in vector space
- “Embedding”: We’re **embedding** a non-Euclidian entity [a word] into Euclidian space
- Each row represents the “embedding” for a single word
- This has pretty remarkable properties!

Example 2-D vector space:



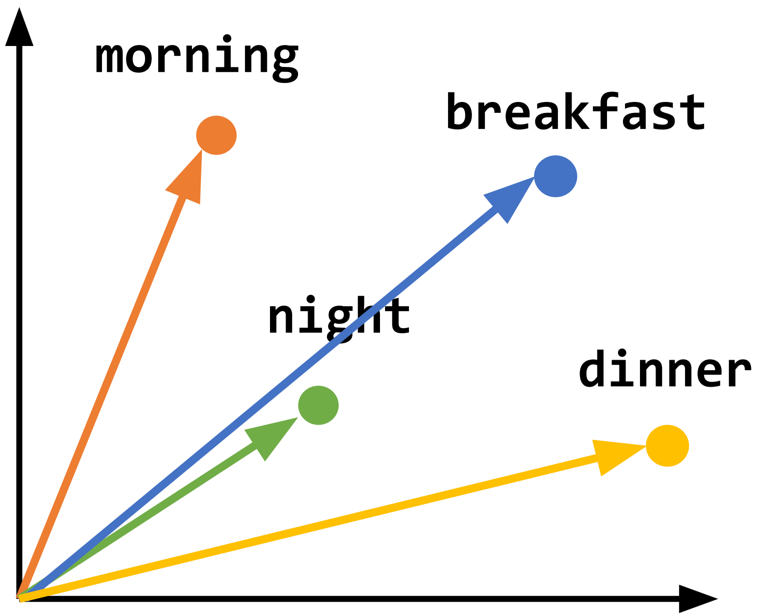
Vocab size: 3

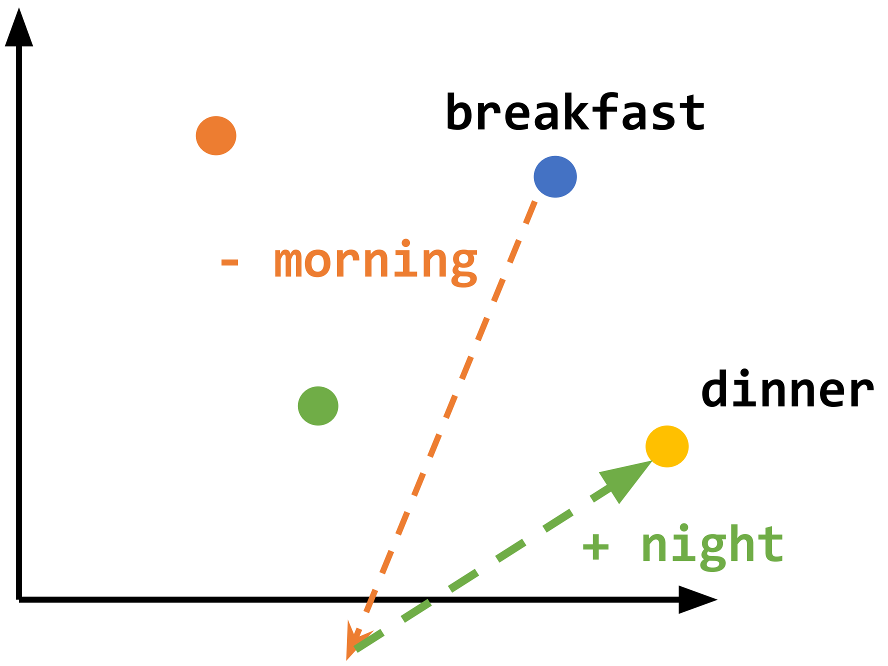
Embed size: 2

1	3
2	1
3	2

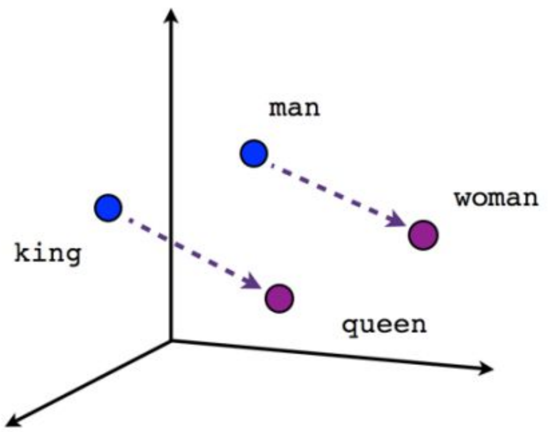
Vector arithmetic in the embedding matrix

Demo [here]





More ‘semantic directions’ in embedding space

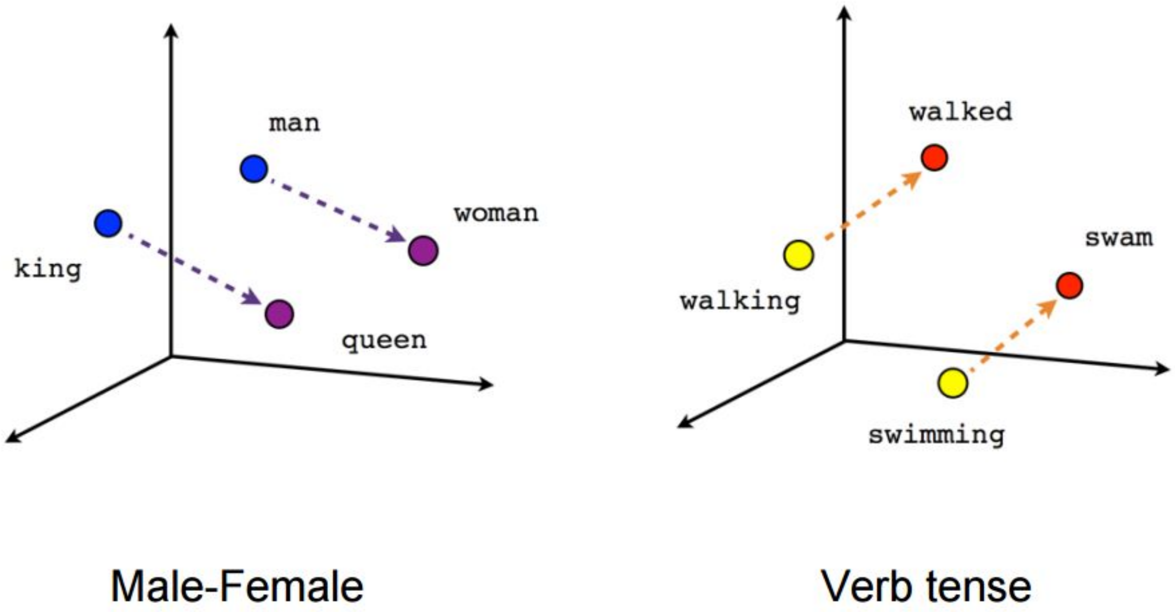


Male-Female

E(queen) – E(king) ≈
E(woman) – E(man)

Semantic: relating to meaning in language

More ‘semantic directions’ in embedding space



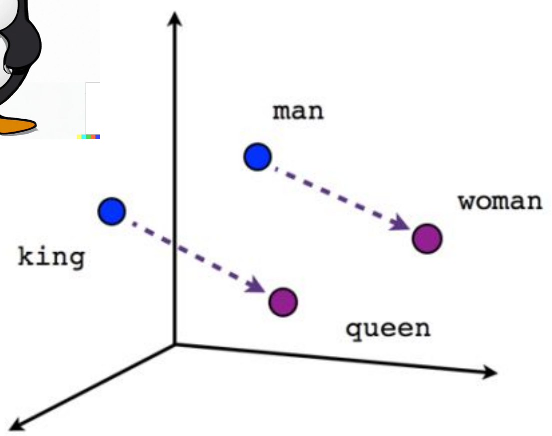
E(queen) – E(king) ≈ E(woman) – E(man)

E(walked) – E(walking) ≈ E(swam) – E(swimming)

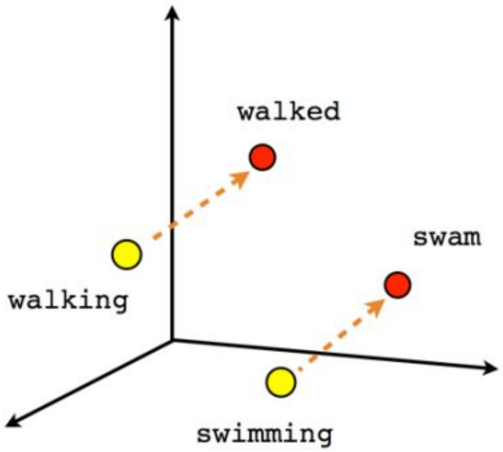
Semantic: relating to meaning in language

More ‘semantic directions’ in embedding space

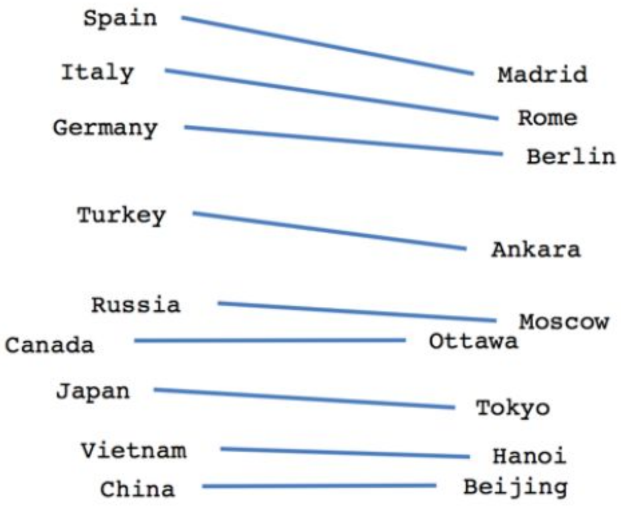
Any questions?



Male-Female



Verb tense



Country-Capital

E(queen) – E(king) ≈ E(woman) – E(man)

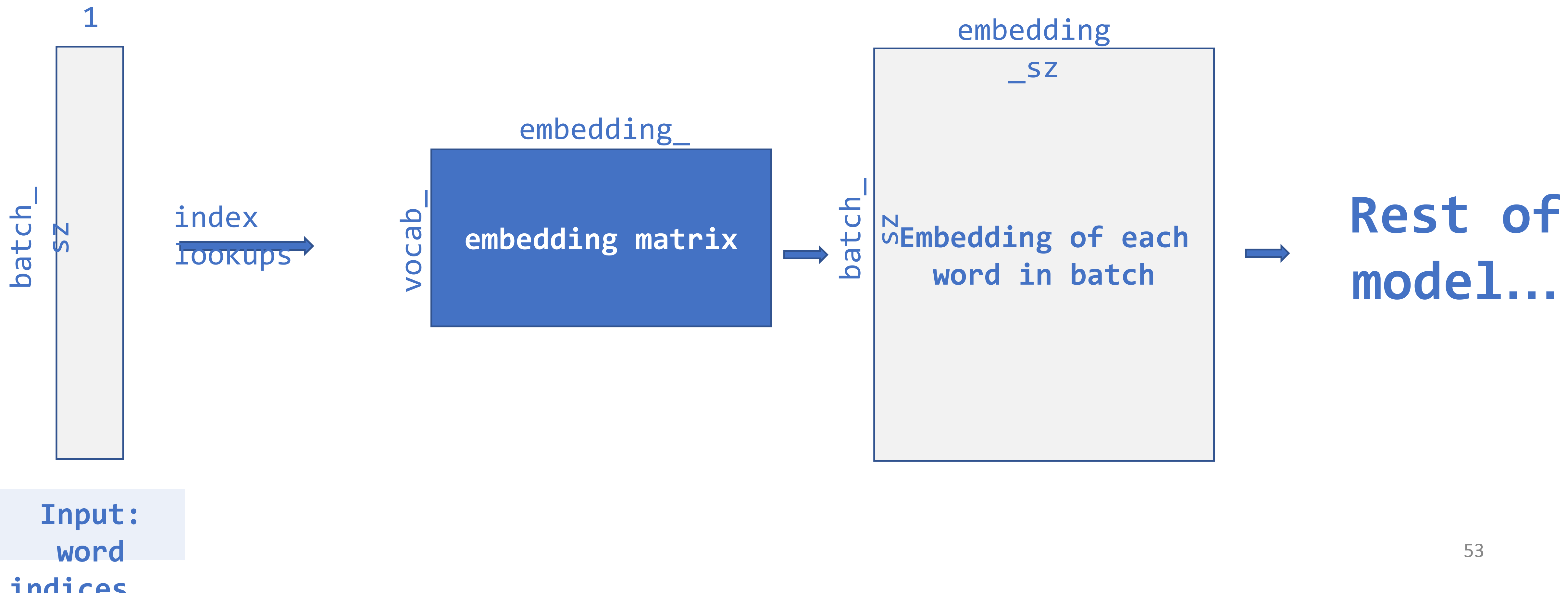
E(walked) – E(walking) ≈ E(swam) – E(swimming)

E(Spain) – E(Madrid) ≈ E(Vietnam) – E(Hanoi)

Semantic: relating to meaning in language

Why do embedding matrices work like this?

When the language model is trained, it's incentivized to put words with similar context near each other in the embedding space.



Why do embedding matrices work like this?

Let’s say in the middle of training…

`P(“happily”| “they danced”) = hi`

`P(“gleefully”| “they danced”) = low`

Then, the model sees a lot of **`“danced gleefully”`**

How do we increase **`P(“gleefully” | “they danced”)`**?

Why do embedding matrices work like this?

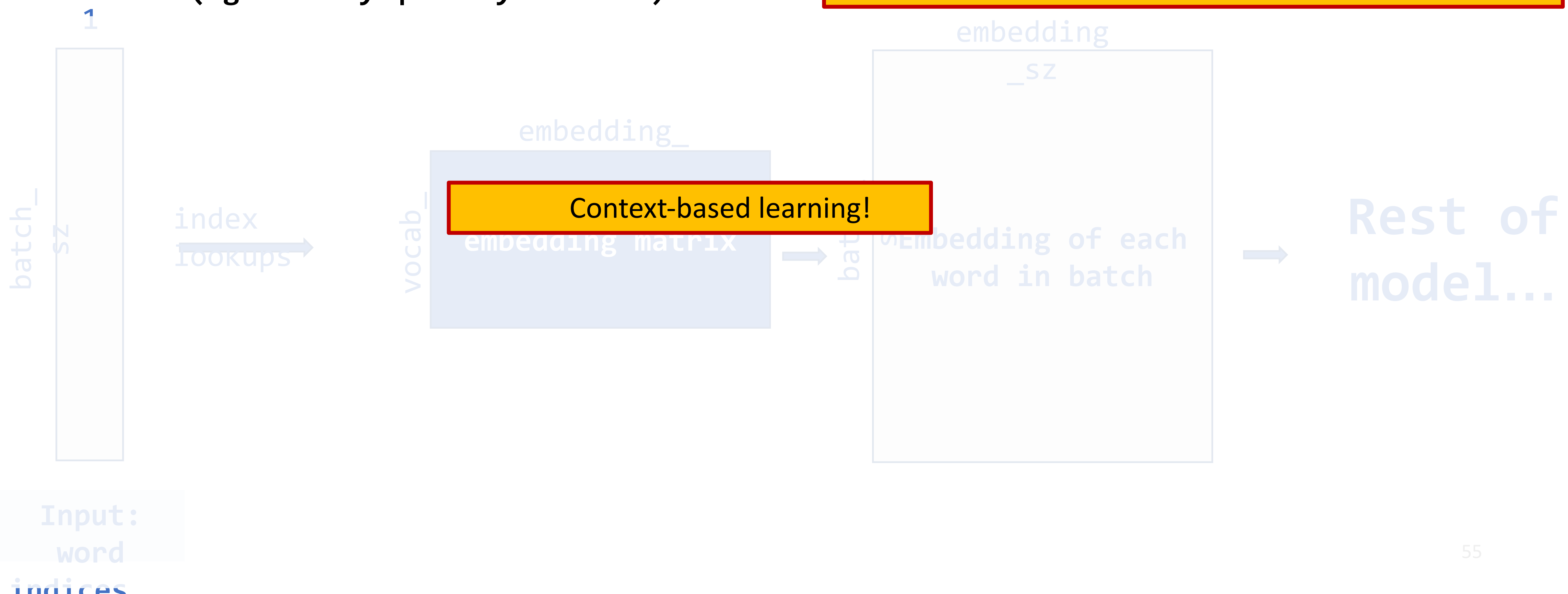
Let's say in the middle of training…

`P("happily"| "they danced") = high`

`P("gleefully"| "they danced") = low`

Since probability is calculated based on the embedding matrix…

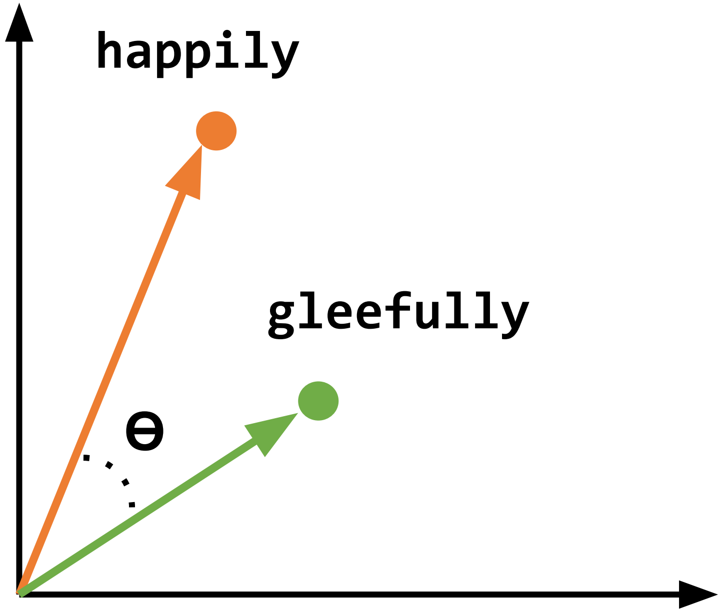
Modify the embedding of **"gleefully"** so that it's similar to the embedding of **"happily"**!



Context-based learning!

Quantifying "similarity"

- $$cosine\ similarity = \cos(\theta) = \frac{A \cdot B}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$



$\cos(0°) = 1$

$\cos(90°) =$ -0.448

$\cos(180°) =$ -0.598

Limitations of the context-based approach

- Context is correlated with meaning, but context != meaning
- Synonyms typically have similar context:
  - **`P("happily" | "they danced")`**
  - **`P("gleefully" | "they danced")`**
- ...but often antonyms do, too:
  - **`P("happily" | "they danced")`**
  - **`P("unwillingly" | "they danced")`**
- "happily" and "unwillingly" might be used in similar contexts, but have the ***opposite*** meaning a language model might (erroneously) give them similar embeddings

Other failure modes are even more dire

What happens when your dataset reflects historical / societal biases?

Other failure modes are even more dire

What happens when your dataset reflects historical / societal biases?

Google News word2vec:

- Large set of *pretrained* word embeddings, published 2013
- Dataset: news articles aggregated by Google News (100 billion words)

Other failure modes are even more dire

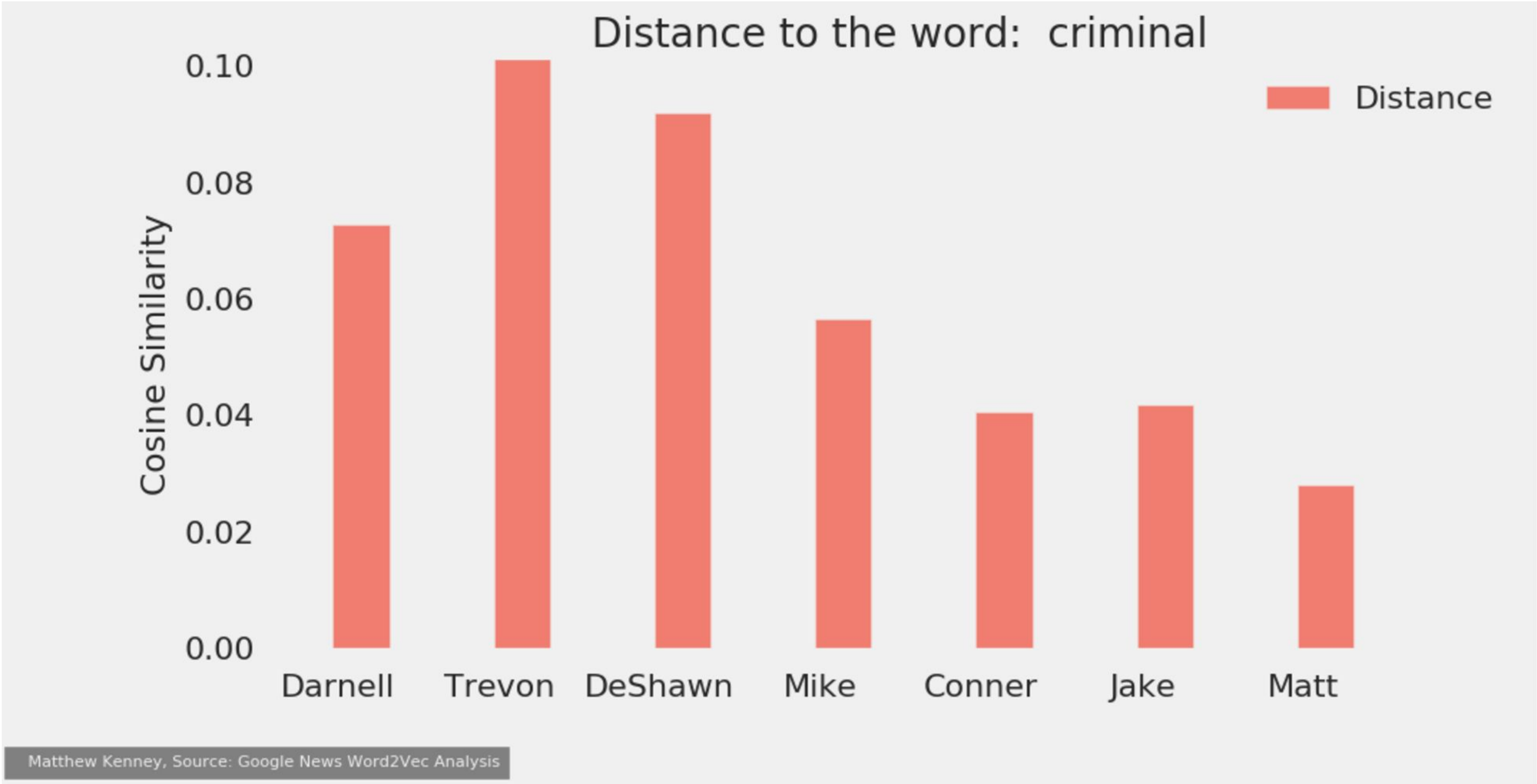
What happens when your dataset reflects historical / societal biases?

Google News word2vec:

- Large set of *pretrained* word embeddings, published 2013
- Dataset: news articles aggregated by Google News (100 billion words)

What kinds of relationships do these embeddings contain?

Google News word2vec

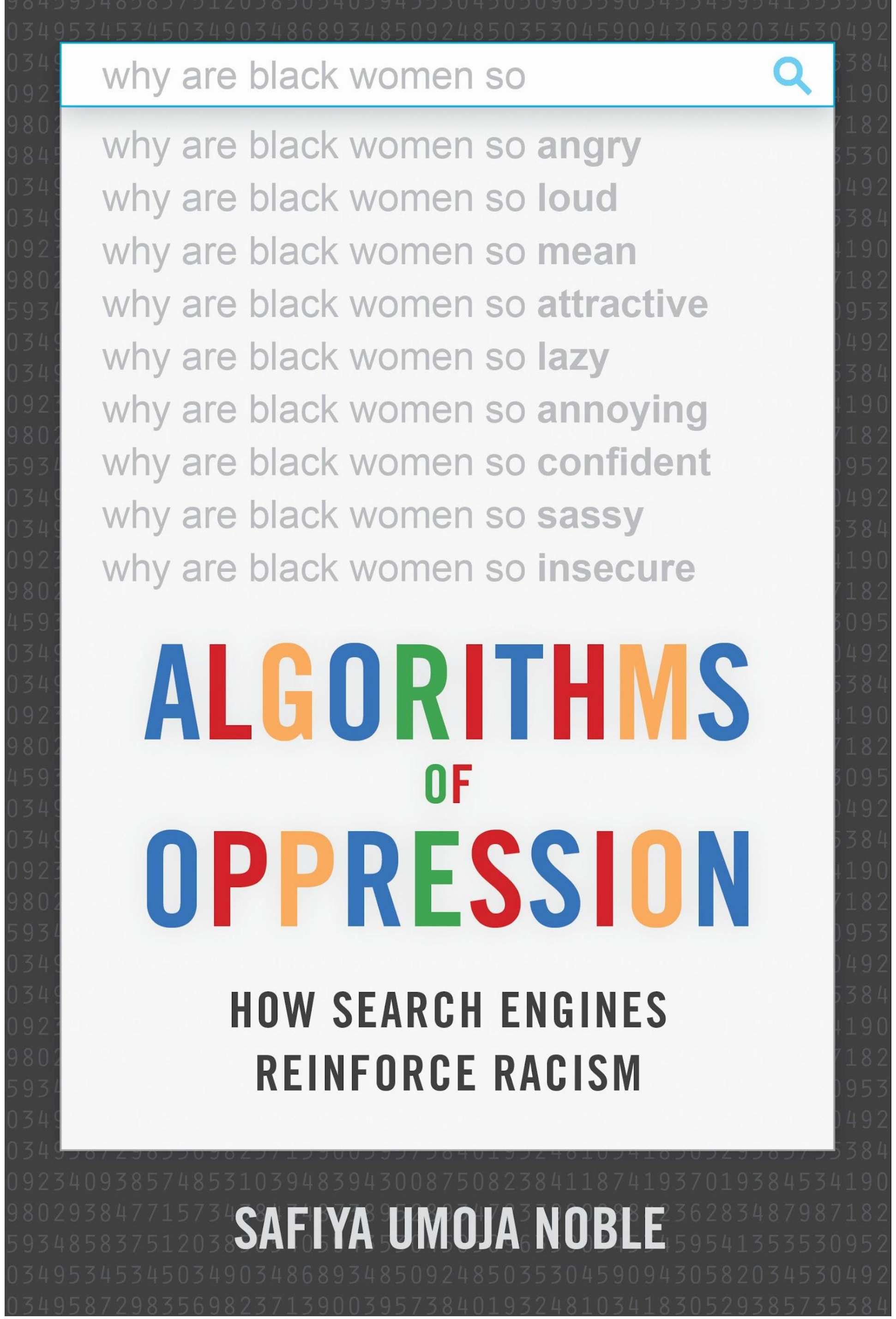


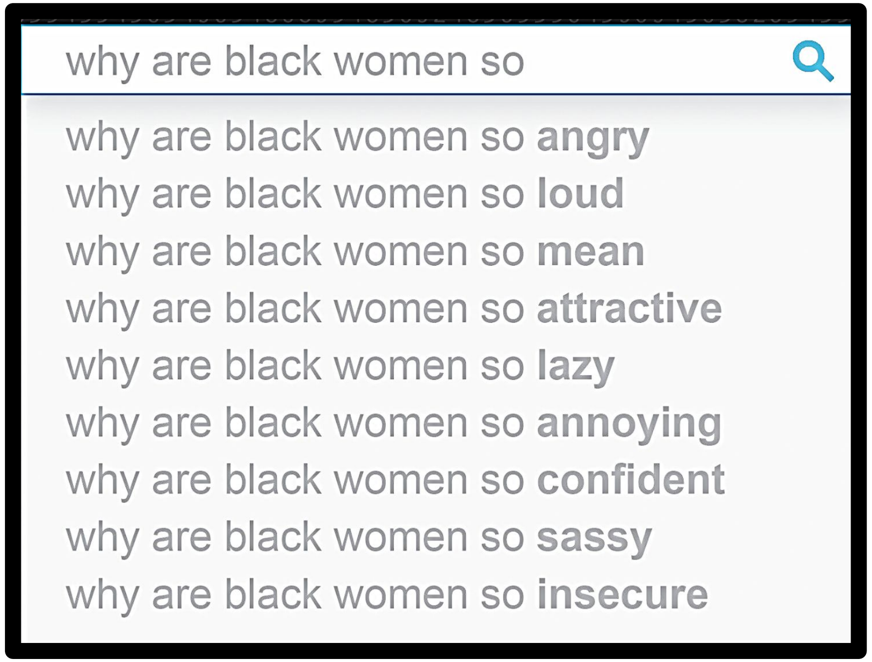
http://www.mattkenney.me/google-word2vec-biases/

Google News word2vec

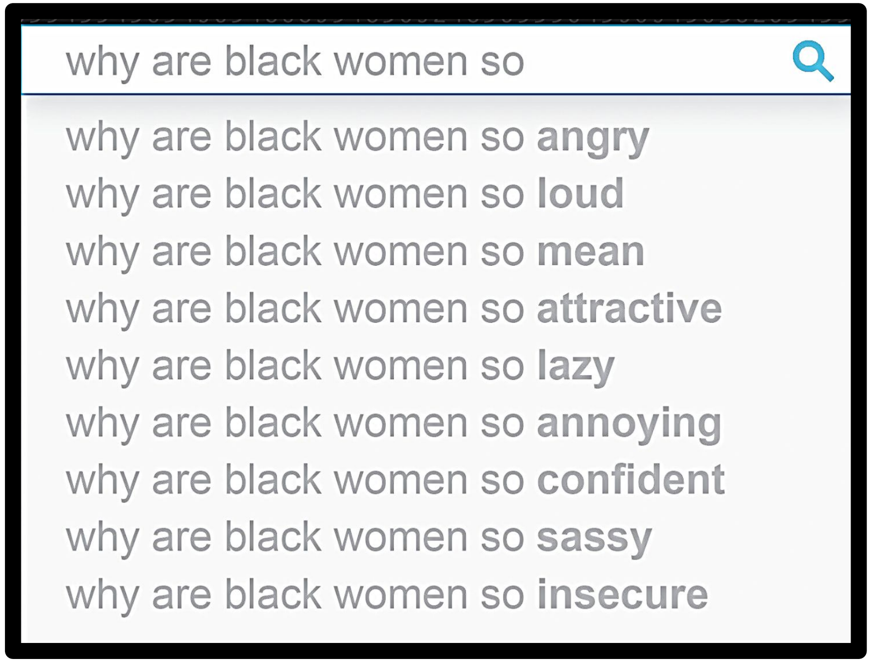
- **Why did this happen?**
  - The training dataset (news articles) was biased.
  - The news cycle **over-represents** crimes by black perpetrators
    - (Entman 94, Gilliam et.al. 96, Dixon 08, Dixon 15) – this is true over time as well
  - Viewers respond more strongly to news stories about crimes by black perpetrators.
    - (Dixon and Maddox 06, Dixon and Azocar 07, Hurley et.al. 15)
    - (News outlets optimize for clicks, therefore report crime by black people more)

http://www.mattkenney.me/google-word2vec-biases/


why are black women so
why are black women so angry
why are black women so loud
why are black women so mean
why are black women so attractive
why are black women so lazy
why are black women so annoying
why are black women so confident
why are black women so sassy
why are black women so insecure
ALGORITHMS
OF
OPPRESSION
HOW SEARCH ENGINES
REINFORCE RACISM
SAFIYA UMOJA NOBLE



- In ~2010, when Noble started working on this book, these were the real Google autocomplete suggestions
- ***Takeaway: language models reproduce the biases of the data on which they are trained***
  - ...unless special care is taken—we have an upcoming lab on this!



- Think about the algorithms behind autocomplete, or ad recommendation…

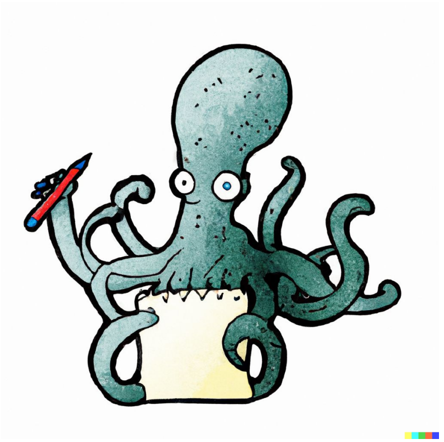
- ***The math might be cool, but there's more to algorithms than math. It is important to consider their potential ethical and social implications once deployed***

Recap

Language modeling

- Natural Language
- Counting/N-grams
- Limitations of traditional methods

And this I pray, that your love may abound yet more and more like a controlled use of shared memory.



Language modeling using Deep Learning

- Learning embedding matrix
- Useful properties of embeddings
- Limitations of context-based learning