How to get from linguistics to LLMs: Language in vector spaces

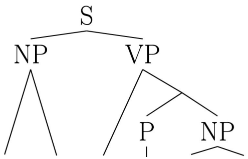
For the Study of Natural & Artificial Intelligence

Isabel Papadimitriou

Ling 83, Apr 21 2025

Motivating question:

How can we get everything that's going on in language…



… into a computational model?

A: Through learning how to represent language in large, continuous vector spaces

```
"red" = [0.6, 53.0, 2.4, 0.2 ...]

"...what light through yonder window breaks" =
[21.2, 112, 6.8, 22.0 ...]

  ...   ...
```

This class:

1) Introduction to vector spaces
2) Word vectors
    - Count-based word vectors
    - Dense word vectors
3) Introducing language models
4) LM demo
5) LM discussion and analysis

By the end of this class you'll:

- Understand some of the **basic ideas** behind modern computational linguistics (and why we do what we do)
- Have some familiarity with loading, using, and understanding a language model

This class:

1) **Introduction to vector spaces**
2) Word vectors
    - Count-based word vectors
    - Dense word vectors
3) Introducing language models
4) LM demo
5) LM discussion and analysis

Symbolic representation is great — but not enough for comp. ling

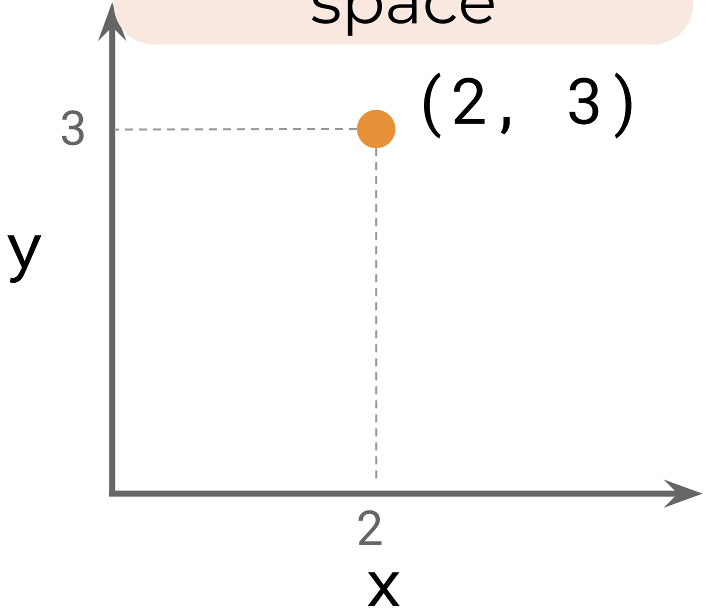
- Language is symbolic
- We take a complex, continuous world, and we **compress it into symbols** like “apple” or “jump”
- Language analysis is full of **abstract ideas**, like “noun phrase” or “phoneme”

- But, if we want to make a computational model, we simply can’t take into account every abstraction at once

What is a vector?

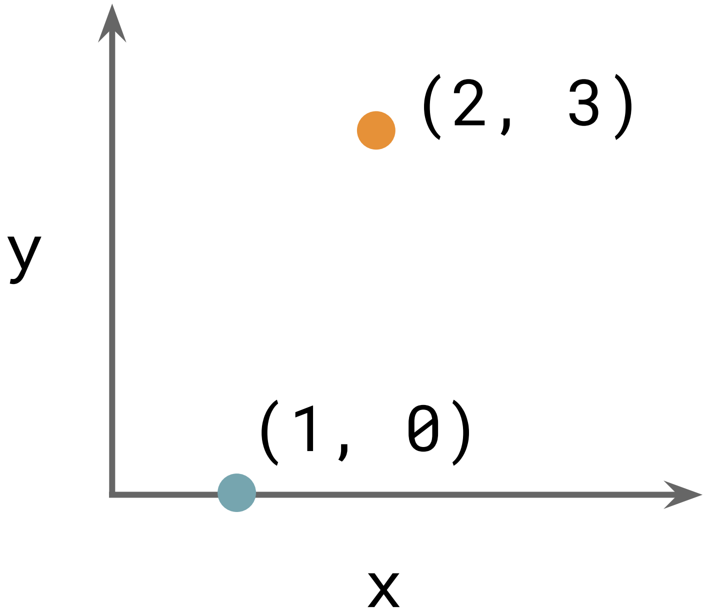
- A point in *n*-di for our purposes)

Back to continuous space



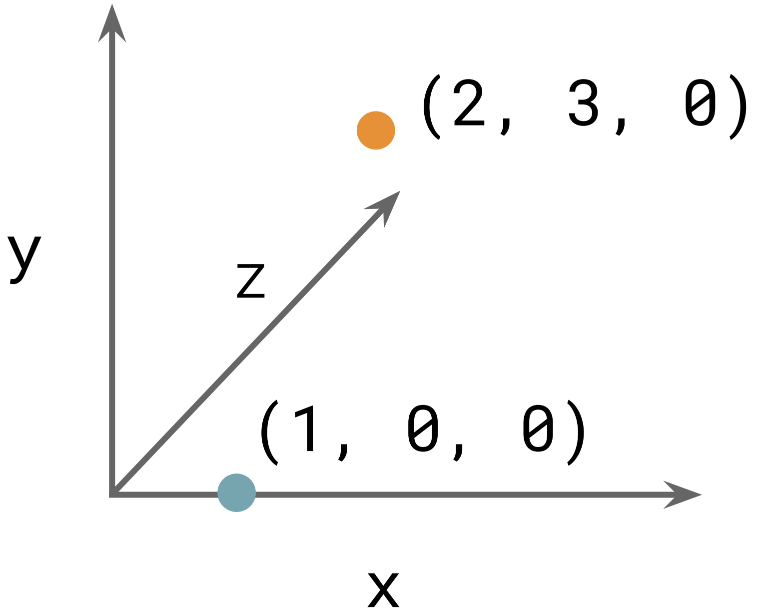
What is a vector?

- A point in *n*-dimensional space (for our purposes)



What is a vector?

- A point in *n*-dimensional space (for our purposes)



What is a vector?

- Our *perception* stops at 3 dimensions, but we don't have to
- Vectors are a very flexible abstraction for describing anything that has many features

A 10-feature vector:

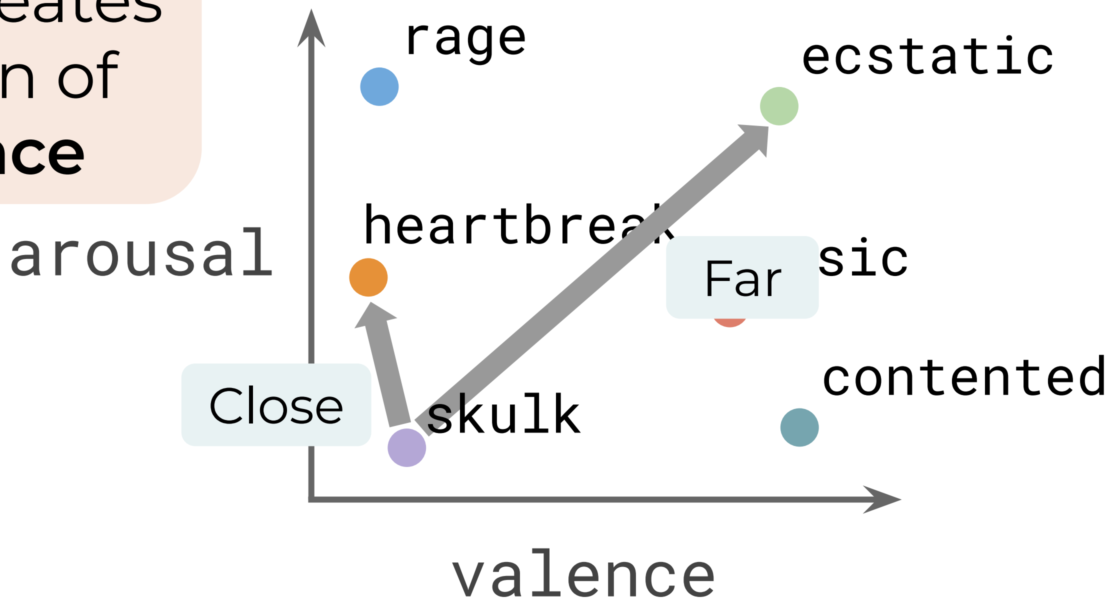
```
[1, 0, 8, 4, 10, 0, 5, 2, 2, 9]
```

eg, "A farm with 1 pig, 0 cows, 8 chickens, 4 ducks…"

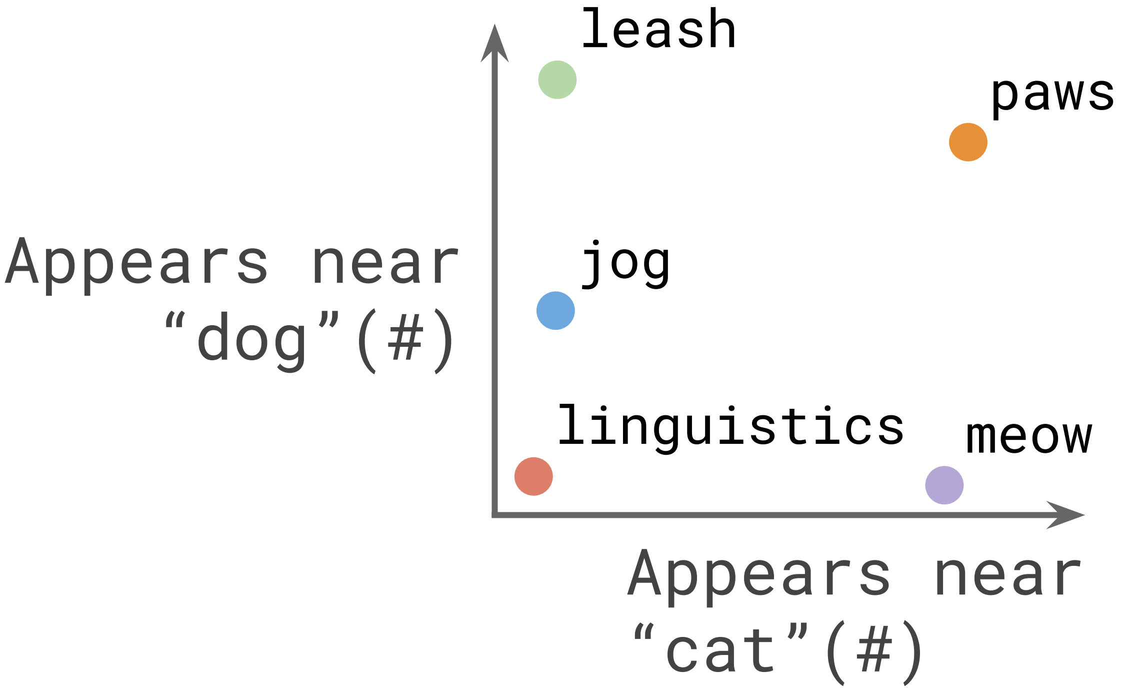
Example: a famous 2-D word vector space

[Osgood et al 1957]

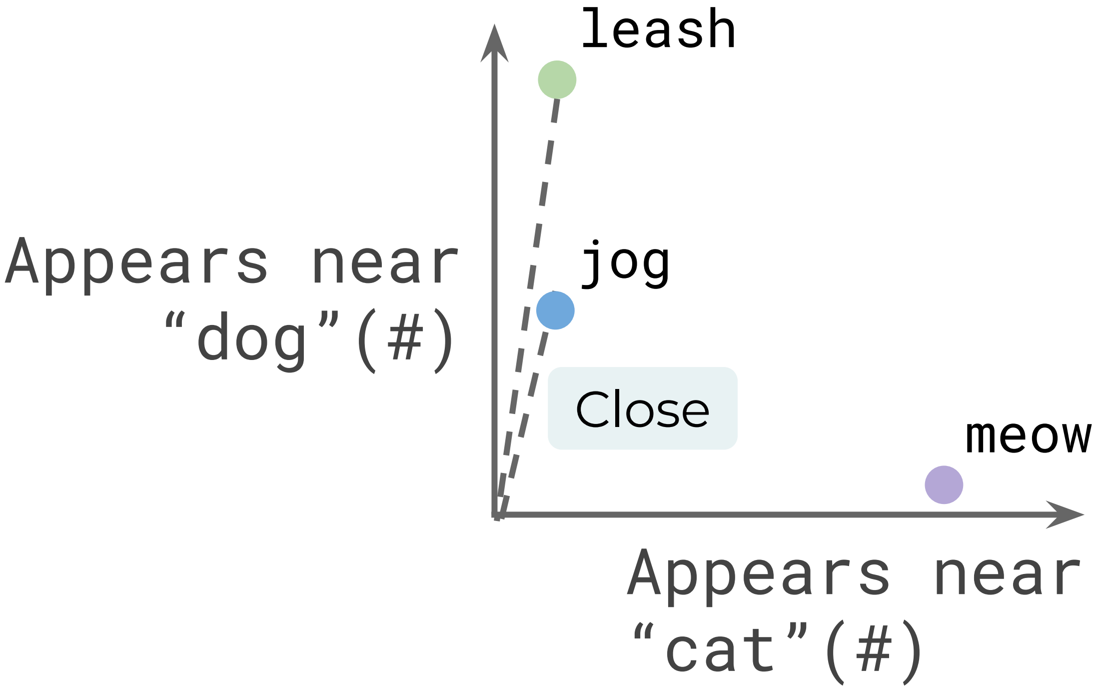
The vector space creates a notion of **distance**



Example: another 2-D word vector space

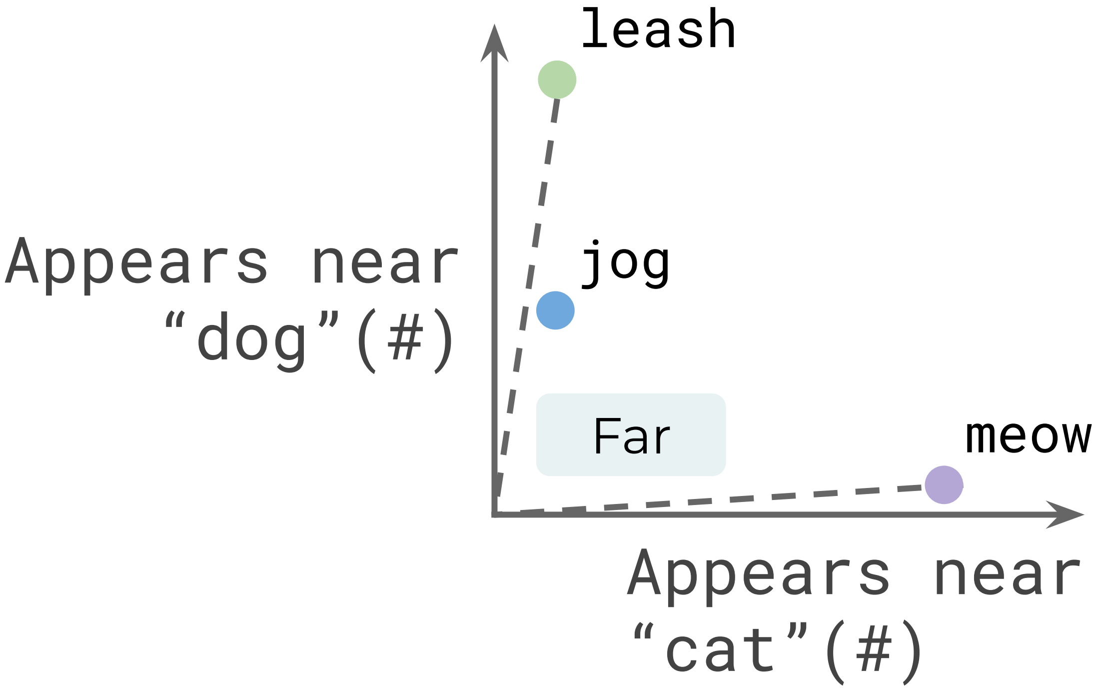


Example: another 2-D word vector space



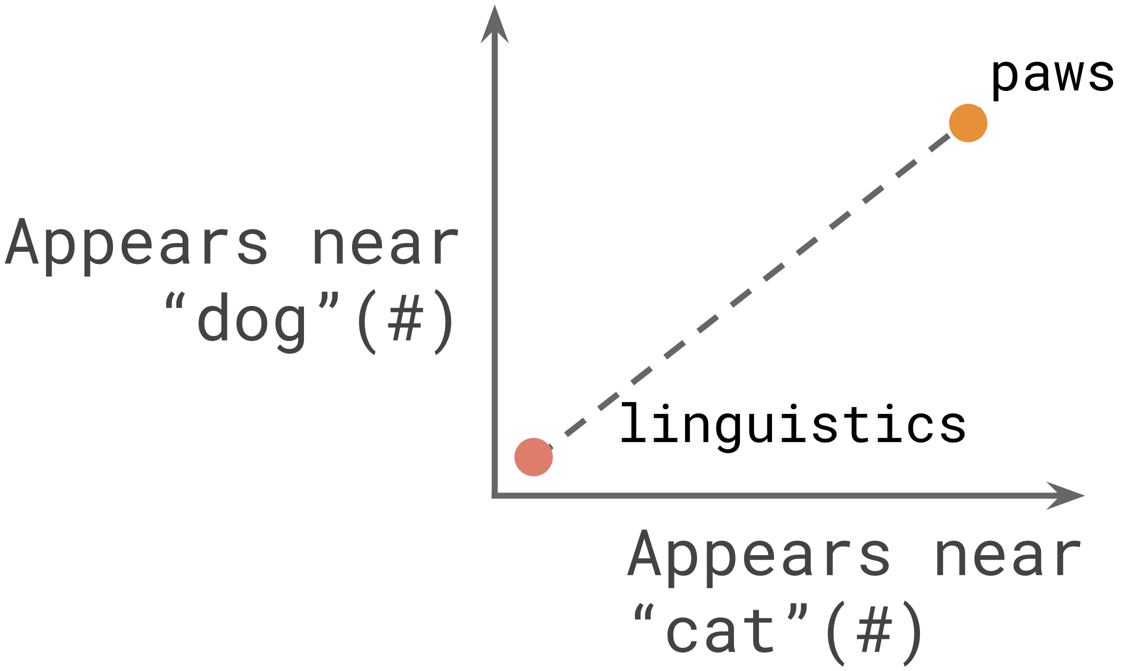
A relevant distance metric here is **angle** – the dog:cat ratio

Example: another 2-D word vector space



A relevant distance metric here is **angle** – the dog:cat ratio

Example: another 2-D word vector space



Distance is still a relevant metric!

So, we can get some semantic structure from putting words in vector spaces

- This is great, because vectors can be **inputs** to computational models
- Vector representations give models some sense of the nebulous idea of **semantic relatedness**

Vector semantics mean that models don’t learn every meaning separately

Language Model Task:

Predict the next word, the previous word was “**good**”

Predict the next word, the previous word was “**great**”

Classic ngram model: no relation…

Vector semantics mean that models don't learn every meaning separately

Language Model Task:

~~Predict the next word, the previous word was "**good**"~~

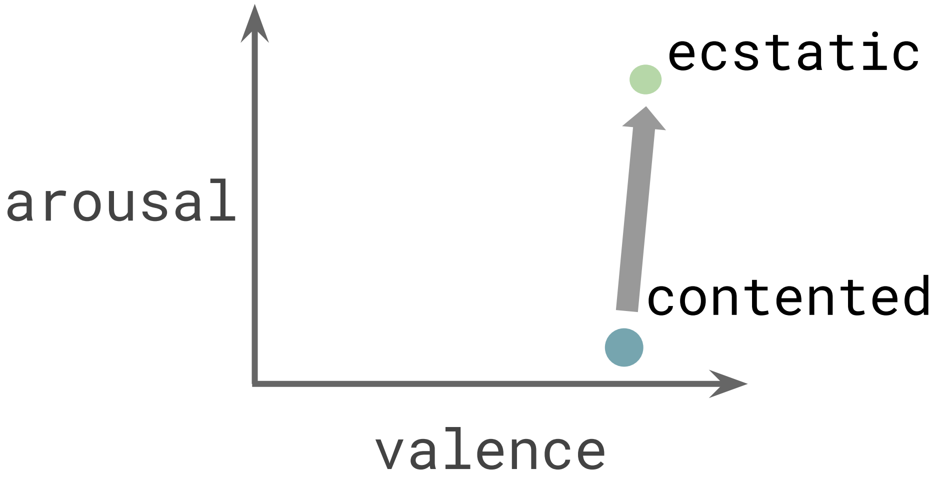
~~Predict the next word, the previous word was "**great**"~~

Predict the next wor
previous word was
[1, 6, 3, 18, ...]

Vector distance

ict the next word, the
ous word was
[2, 6, 2, 15, ...]

Vector spaces capture many **different types** of relationships simultaneously



Lots of information:
1) Medium distance
2) Similar valence
3) Different arousal

This is very powerful as we scale up to **many dimensions**!

Semantic relationships are multifaceted

- Similarity (desk ↔ table)
- Relatedness (dog ↔ leash)
- Semantic frames (buy ↔ sell)
- Register (automobile ↔ couture)
- Affect (beaming ↔ great)

Can we make a **high-dimensional** vector space to capture this complexity?

This class:

1) Introduction to vector spaces
2) **Word vectors**
   - **Count-based word vectors**
   - Dense word vectors
3) Introducing language models
4) LM demo
5) LM discussion and analysis

We can't think of every feature...

- It's **hard to hand-construct** enough interesting features!

  (This is a big theme of computational linguistics)

- Let's see if we can use some things we know to make a computational model where we don't choose the features

Distributional semantics

If two words have almost identical **environments**,
they have almost identical **semantics**

"oculist", "eye doctor" [Zellig Harris, 1954]

- We wanted a vector representation of **meaning...**

💡 Idea: represent the **environment** as a vector

Idea: Count word co-occurrences

- If we have a corpus of language data...

context *word*

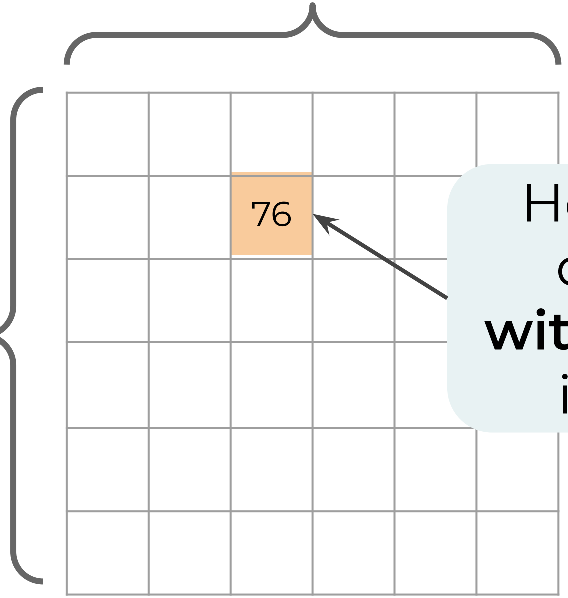
`… Every time [ I drive my car I hear that ] noise…`

- Count: how many times does a word **appear in the context window** of a center word?

Counting word-word co-occurrences

Word B:
50,000 rows

Word A:
50,000 rows

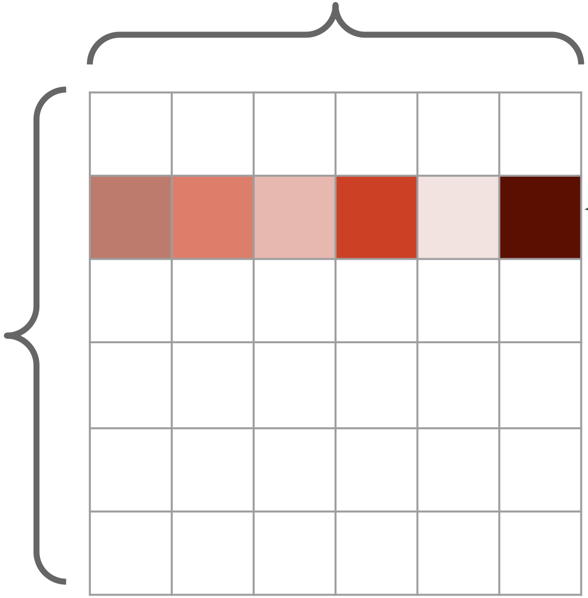


How many times does B appear **within 4 words** of A in our corpus?

Co-occurrence rows are vectors!

Word B:
50,000 rows

Word A:
50,000 rows



50K-dimensional **vector of the environments** that A appears in

Co-occurrence rows are vectors!

	aardvark	...	computer	data	result	pie	sugar
	0	...	2	8	9	442	25
	0	...	1,670	1,683	85	5	4

"cherry"

"digital"

[Jurafsky and Martin 2023, from Wikipedia corpus]

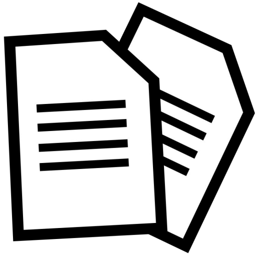
Co-occurrence rows are vectors!

	aardvark	…	computer	data	result	pie	sugar
cherry	0	…	2	8	9	442	25
digital	0	…	1,670	1,683	85	5	4

- We can ask: **How** similar are these words? **Why** are they similar?
- There's ways to improve these vectors, like by lessening the weight of common words like "the"

[Jurafsky and Martin 2023, from Wikipedia corpus]

Our count vectors make a good representation that computers can use



cherry

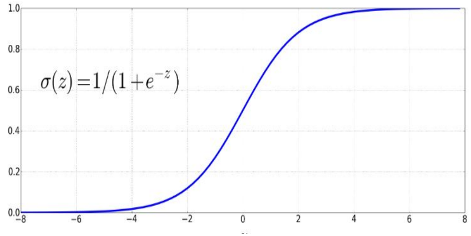
2	8	9	442	25

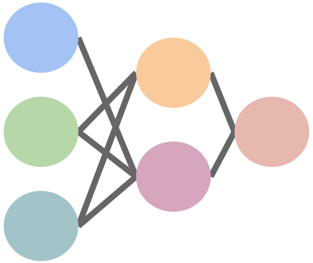
digital

1670	1683	85	5	4

?

1	6	12	512	30





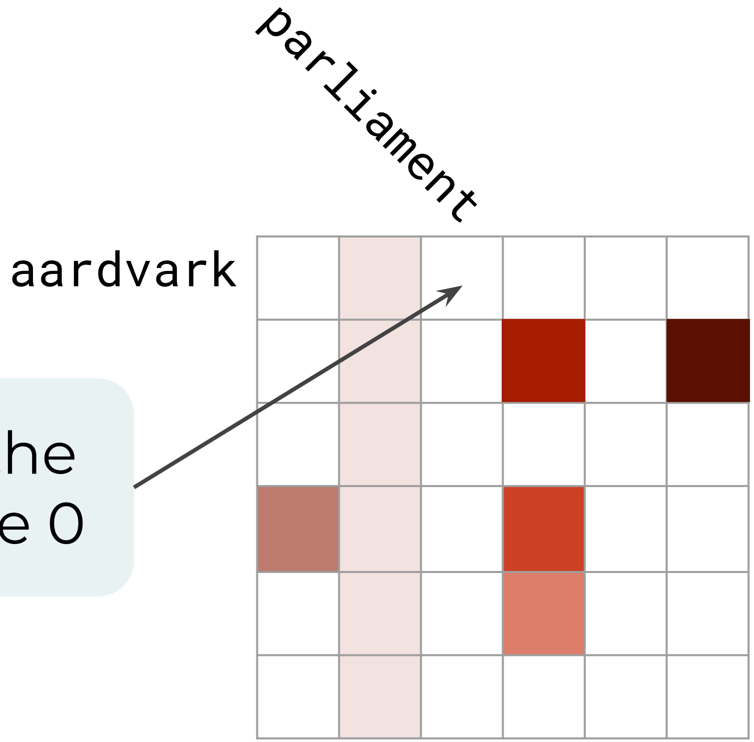
Except… 50K is a lot of dimensions!

Q: What is the meaning of "spoon"?

A: Well, it never appears with "aardvark" … it often appears with "food"… …

This is clearly an inefficient way to describe meaning

Co-occurrence vectors are sparse



Many of the entries are 0

Intuitively: many **more numbers** in each vector than the **information** they contribute

This class:

1) Introduction to vector spaces
2) **Word vectors**
    - **Count-based word vectors**
    - Dense word vectors
3) Introducing language models
4) LM demo
5) LM discussion and analysis

This class:

1) Introduction to vector spaces
2) **Word vectors**
    - Count-based word vectors
    - **Dense word vectors**
3) Introducing language models
4) LM demo
5) LM discussion and analysis

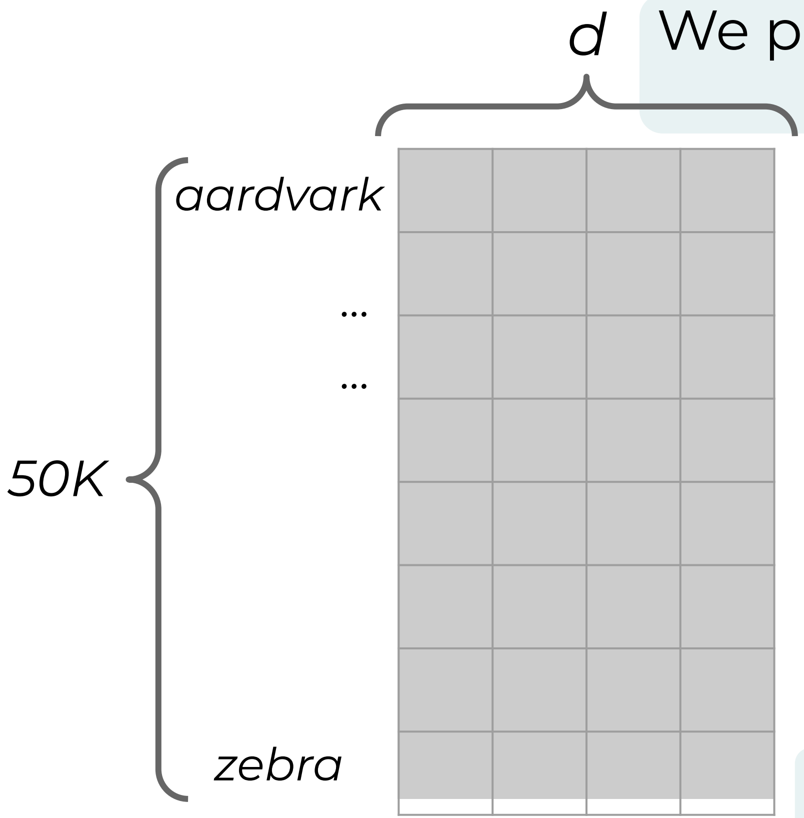
Can we use distributional information to learn more succinct embeddings?

- Same data: corpus of word co-occurrences

`Every time [ I drive my car I hear that ] noise…`

- Main idea: **train classifiers** to predict this distributional information

Dense vectors: word2vec



We pick a dimension, eg 300

Randomly initialize all vectors – Start with no information

Main idea:
Train these vectors to reflect distributional information

Machine learning

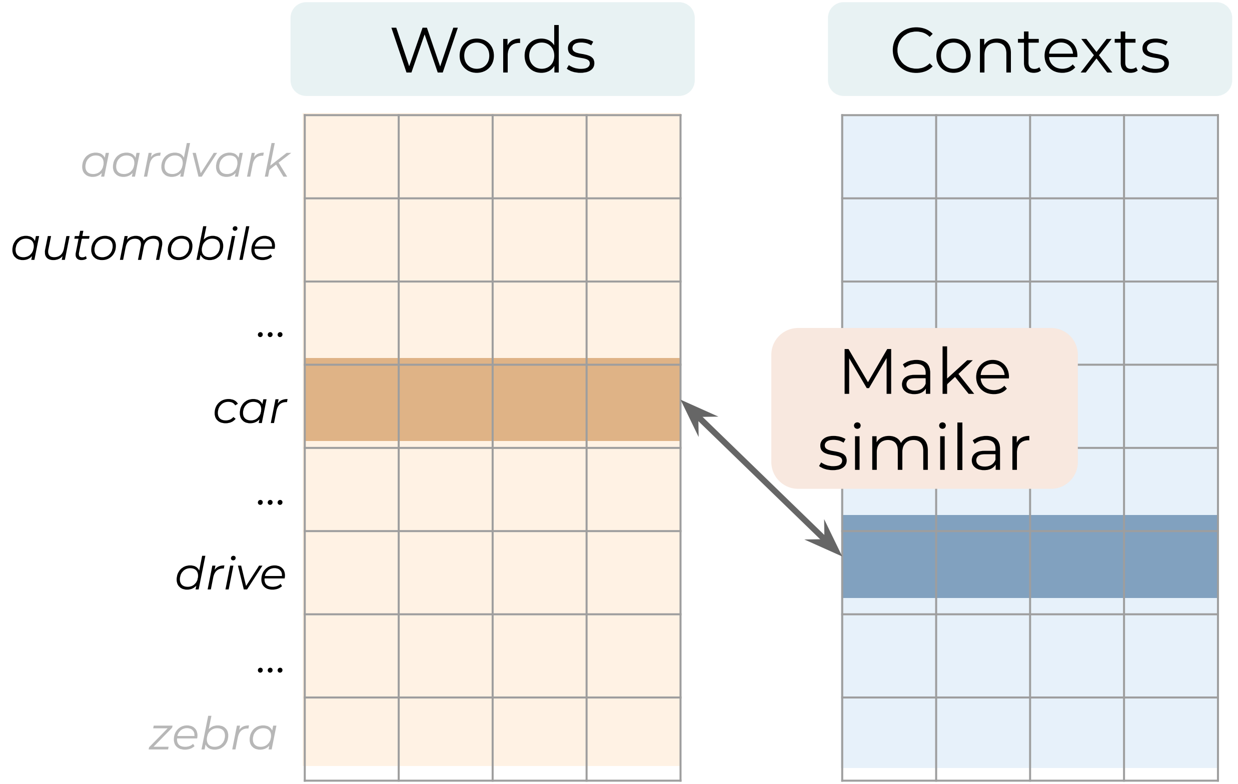
Learning representations for a word from the words in its context

Corpus

context *word*

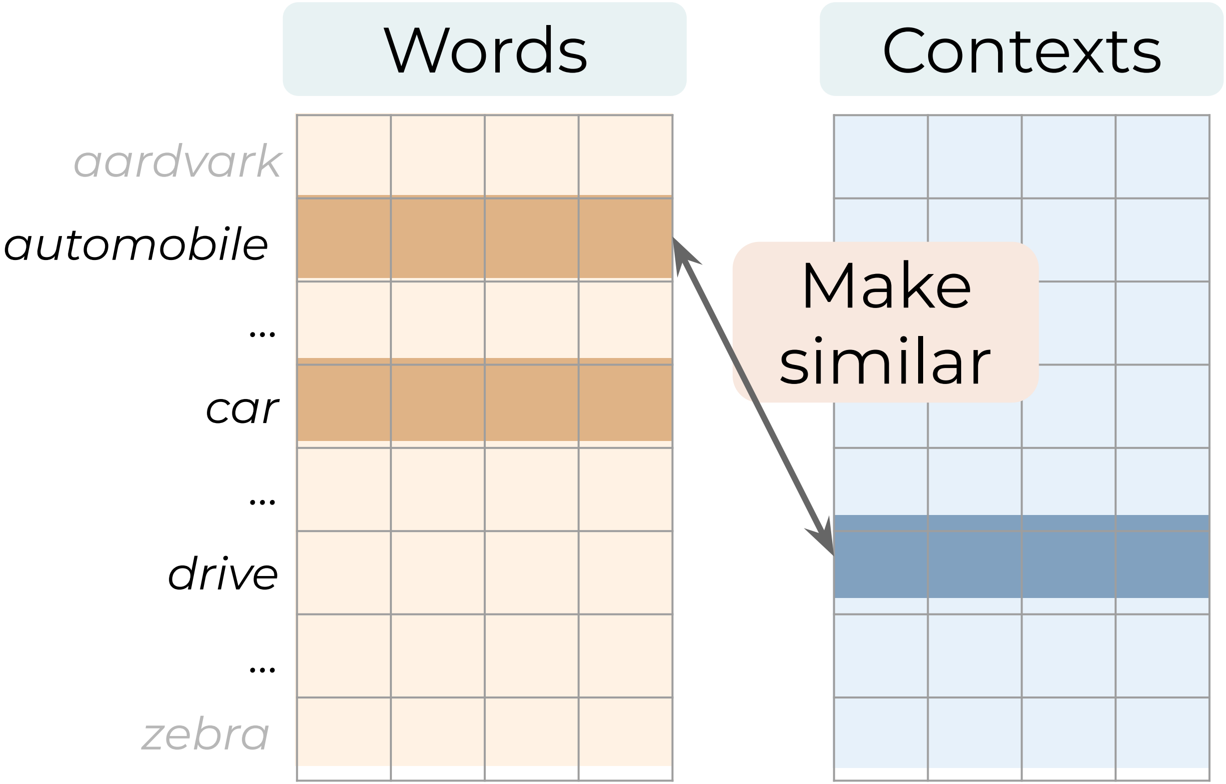
`Every time [ I drive my car I hear that ] noise…`

Training: make word vectors close to the context vectors they co-occur with

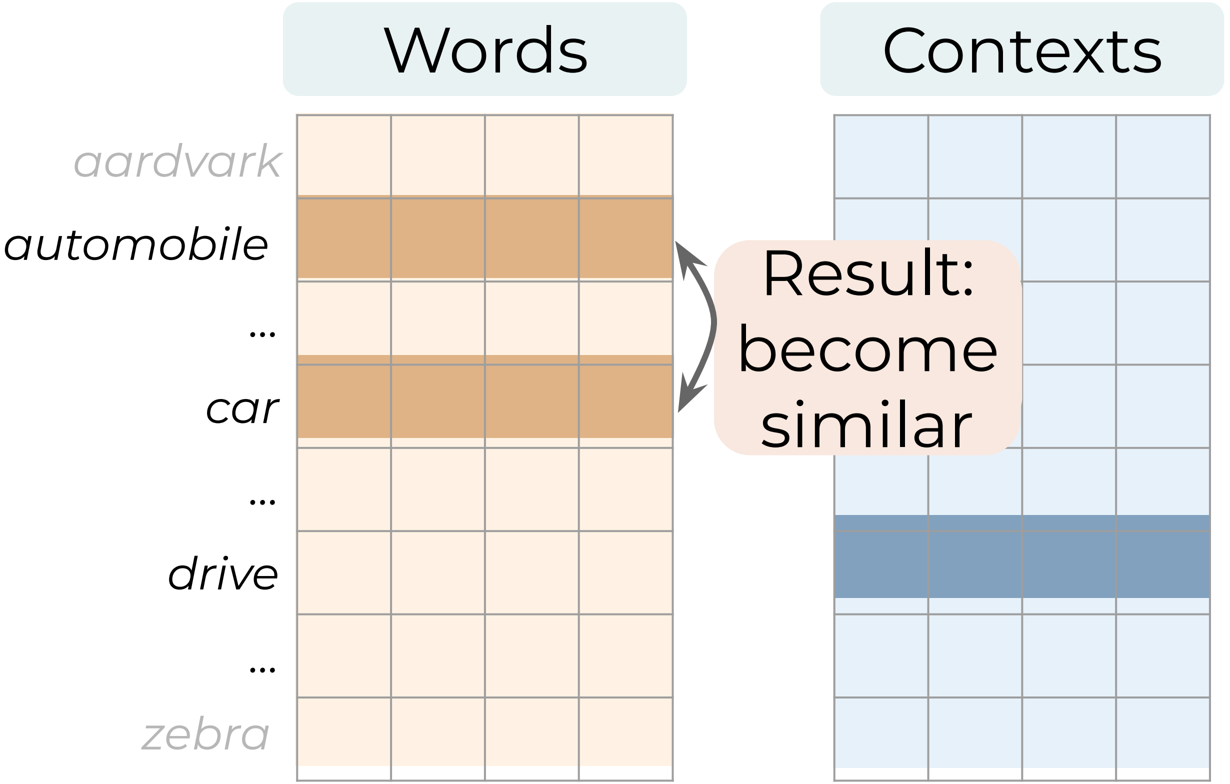


- How do we ‘make’ two vectors similar?
- We **incentivize** the iterative machine learning process

Training: make word vectors close to the context vectors they co-occur with



Training: make word vectors close to the context vectors they co-occur with



Distributional hypothesis!

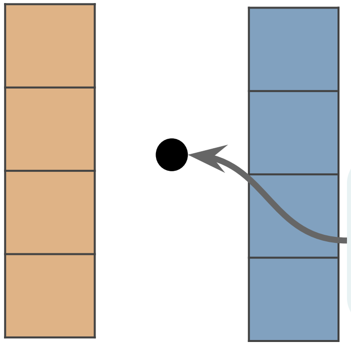
Training: make word vectors close to the context vectors they co-occur with

Corpus

context *word*

`Every time [ I drive my car I hear that ] noise…`

car *drive*



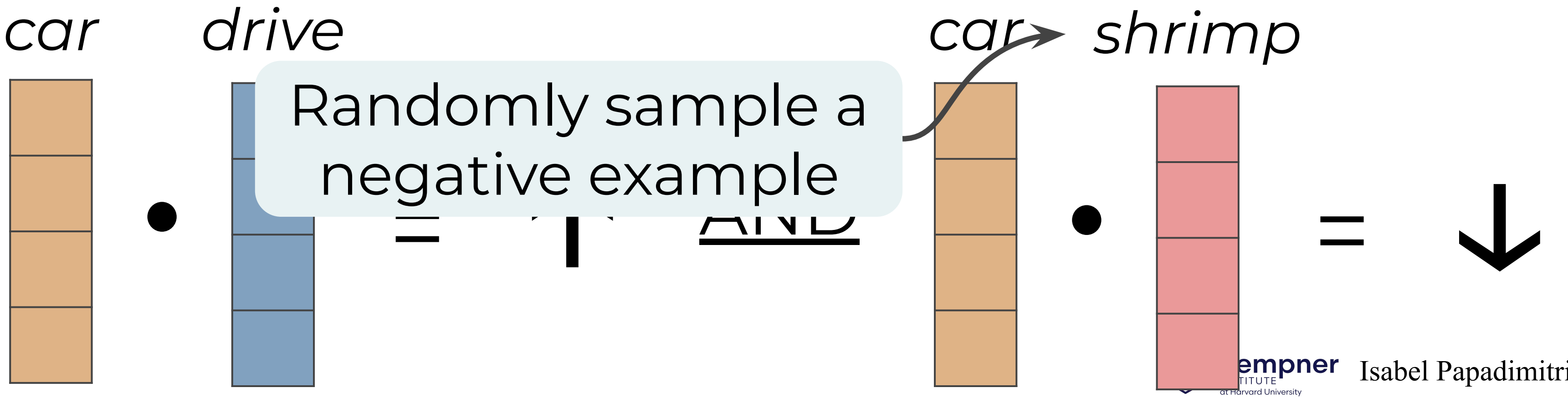
Dot product:
∝ cosine of **angle**

Training: make word vectors close to the context vectors they co-occur with

Corpus

context *word*

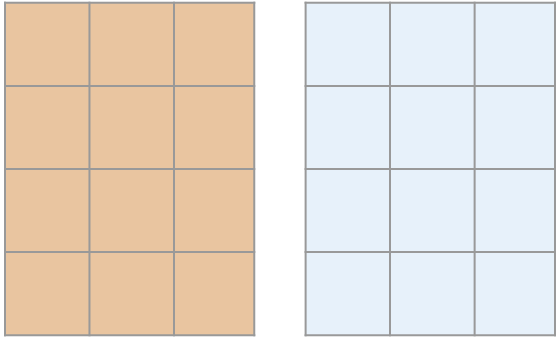
`Every time [ I drive my car I hear that ] noise…`



Now we have a classic machine learning problem:

Parameters

Words *Contexts*



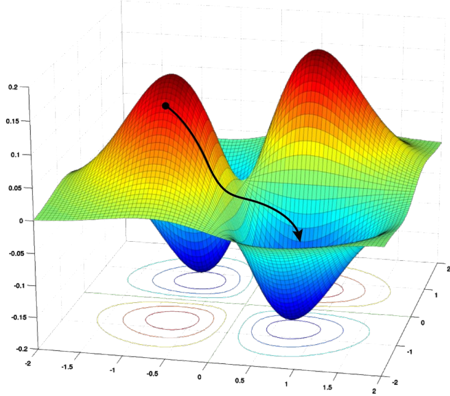
Loss function

For every word, context in corpus...

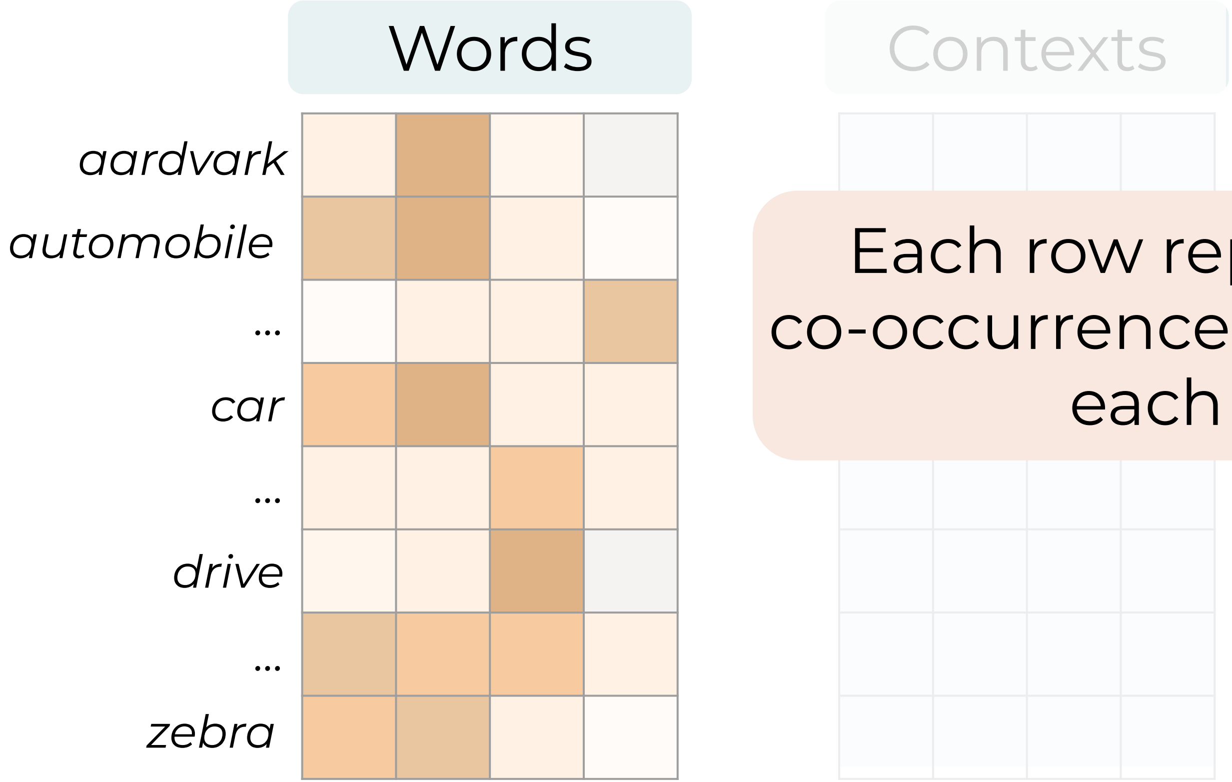
word • context should be high

word • negative should be low

Stochastic gradient descent

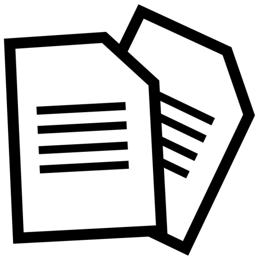


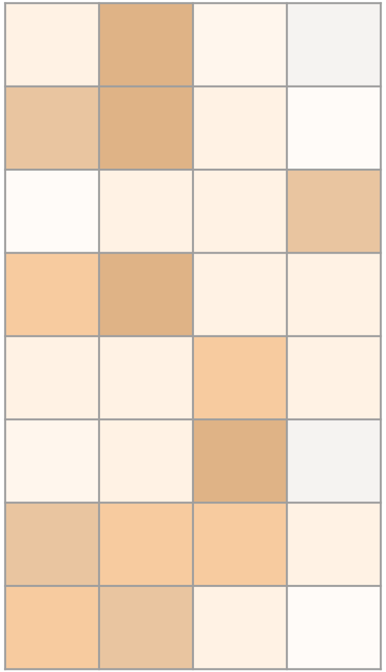
Dense word embeddings

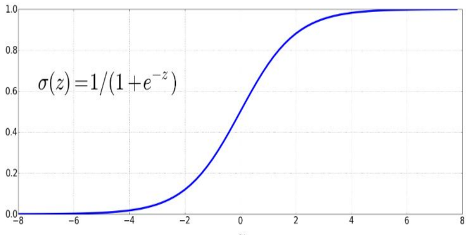


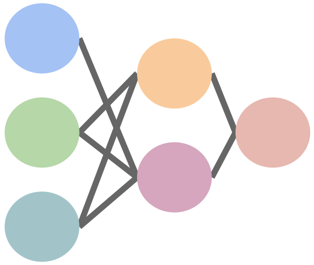
Each row represents the co-occurrence information of each word

Dense embeddings are more succinct, and our methods can use them better







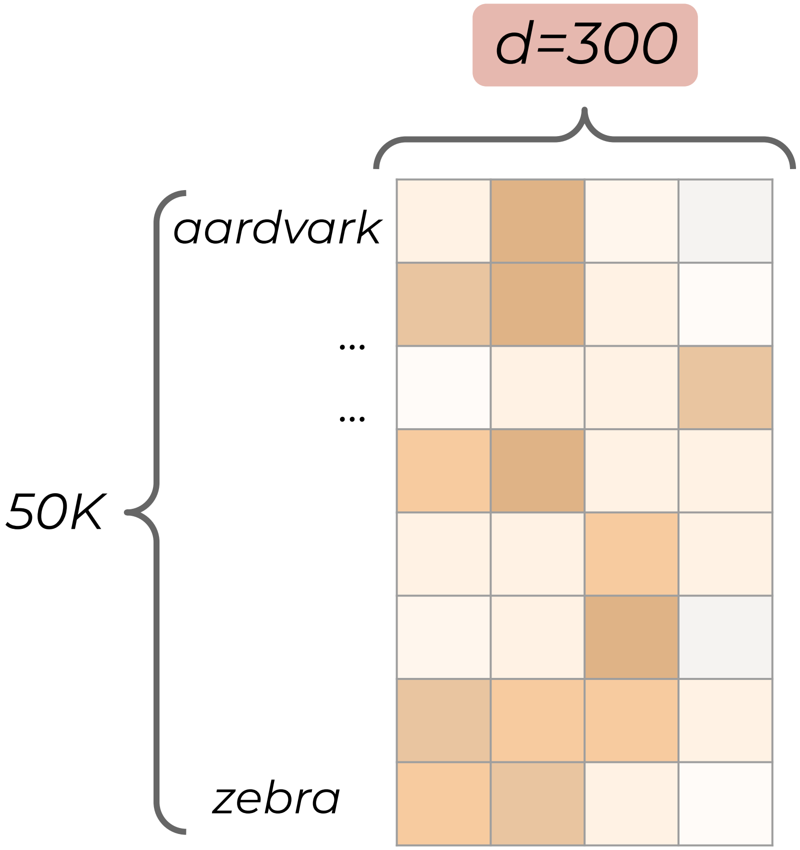


Note: dense embeddings are not **interpretable**

- With co-occurrence word vectors:

  "These two vectors are close because they both co-occur with the word 'marsupial'"

- With word2vec vectors, column dimensions are a mystery


d=300
aardvark
...
...
50K
zebra

Note: dense embeddings are not **interpretable**

- With co-occurrence word vectors:

  “These two vectors are close because they both co-occur with the word ‘marsupial’”

- With word2vec vectors, column dimensions are a mystery

A **tradeoff**: more effective methods in CL are often less interpretable

Should each word just get one meaning vector?

Word meaning is complex, and varies depending on the context

Classic polysemy: **bank** (river) vs **bank** (financial)

[Pustejovsky 1996]

Should each word just get one meaning vector?

Word meaning is complex, and varies depending on the context:

I dove into the **water**

I bought you a **water**

(mass/count)

[Pustejovsky 1996]

Should each word just get one meaning vector?

Word meaning is complex, and varies depending on the context

The **newspaper** fired its editor

John spilled coffee on the **newspaper**

(producer/product)

[Pustejovsky 1996]

Should each word just get one meaning vector?

Word meaning is complex, and varies depending on the context

A **good** knife

A **good** review

A **good** meal

(sharp/favorable/tasty)

[Pustejovsky 1996]

This class:

1) Introduction to vector spaces
2) Word vectors
   - Count-based word vectors
   - Dense word vectors
3) **Introducing language models**
4) LM demo
5) LM discussion and analysis

So, it's possible to represent rich **lexical** information with vectors using machine learning...

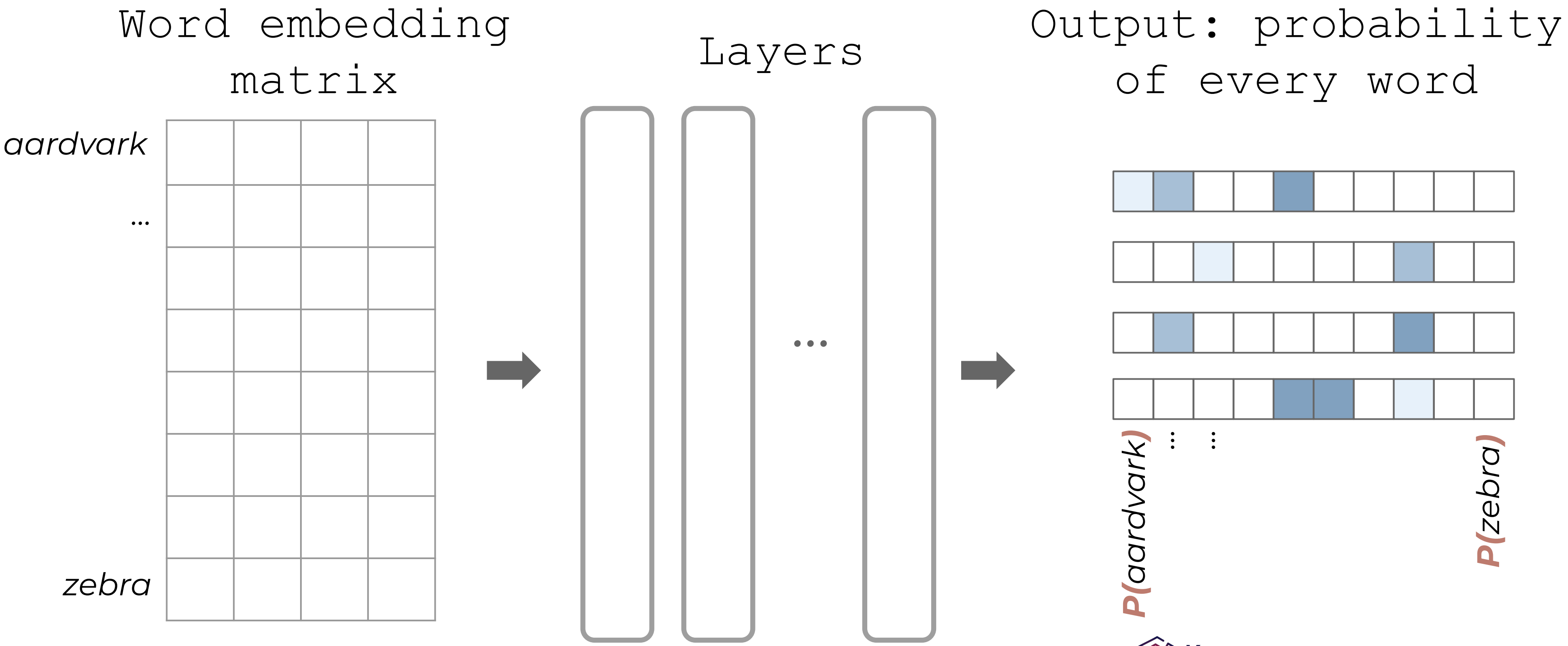
Next step: what about **everything else*** in language?

* I'll focus on **text models**, but there's more "everything" in **speech models**!

Language model: a big neural network trained on one task:

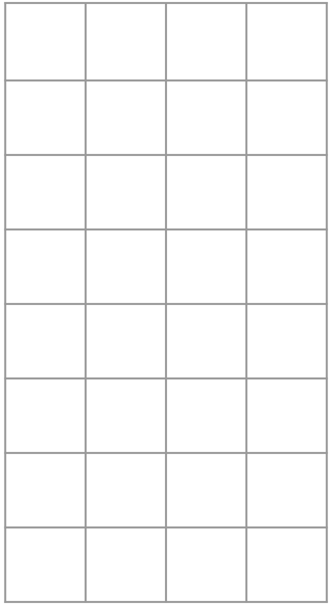
next word prediction

The anatomy of a language model



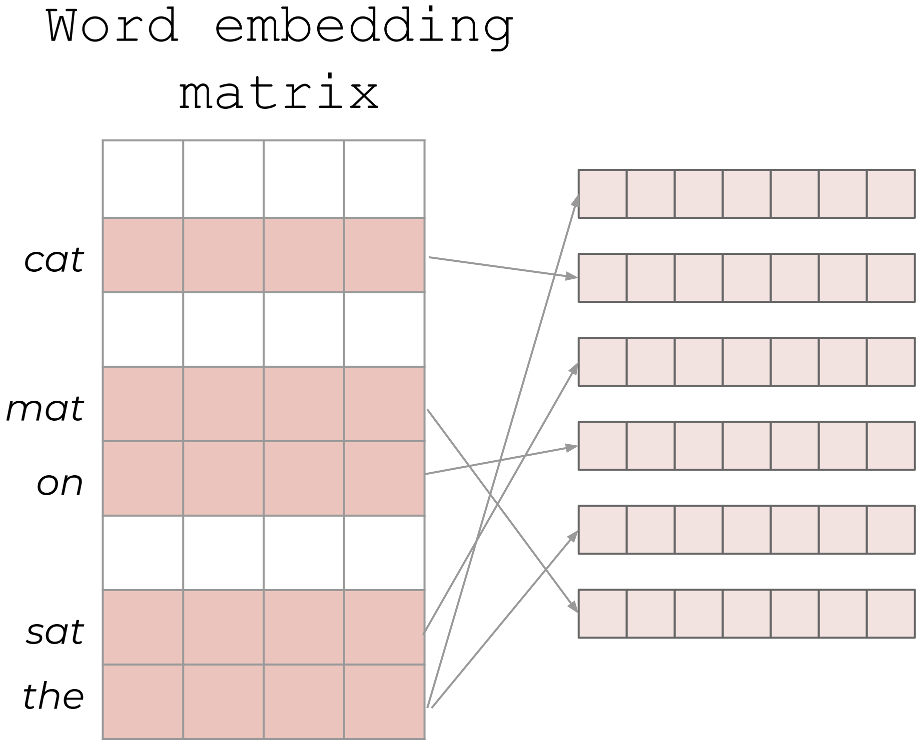
The anatomy of a language model

Word embedding matrix

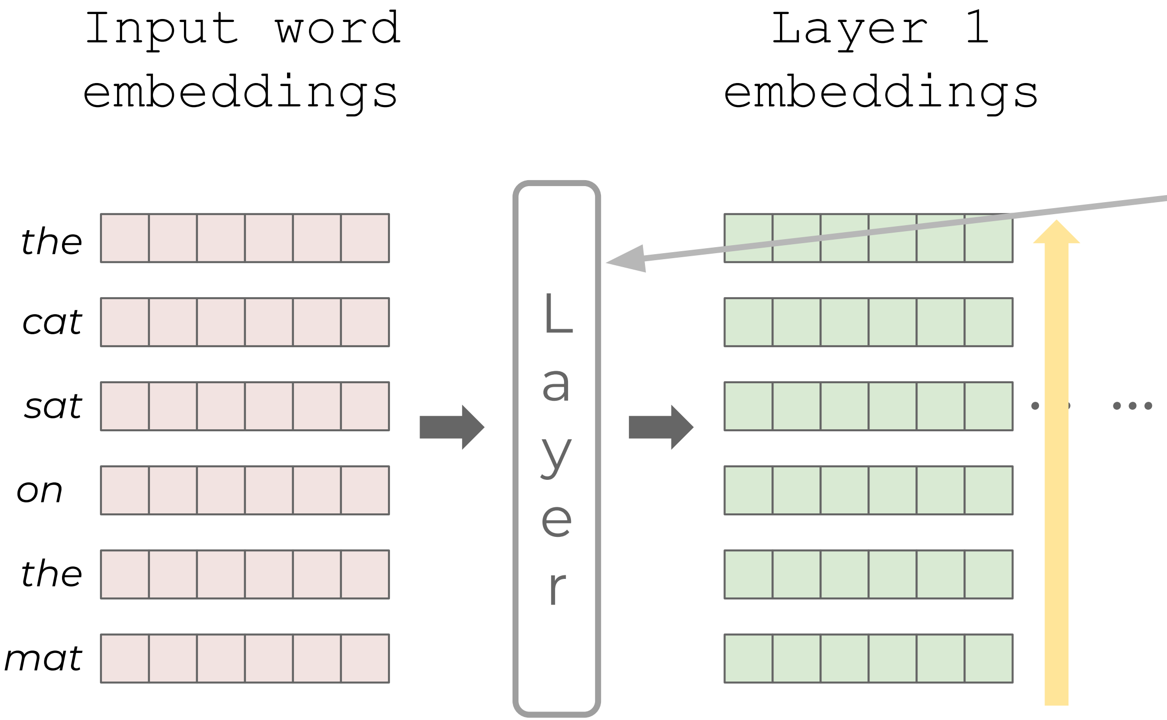


Input: “the cat sat on the mat”

The anatomy of a language model



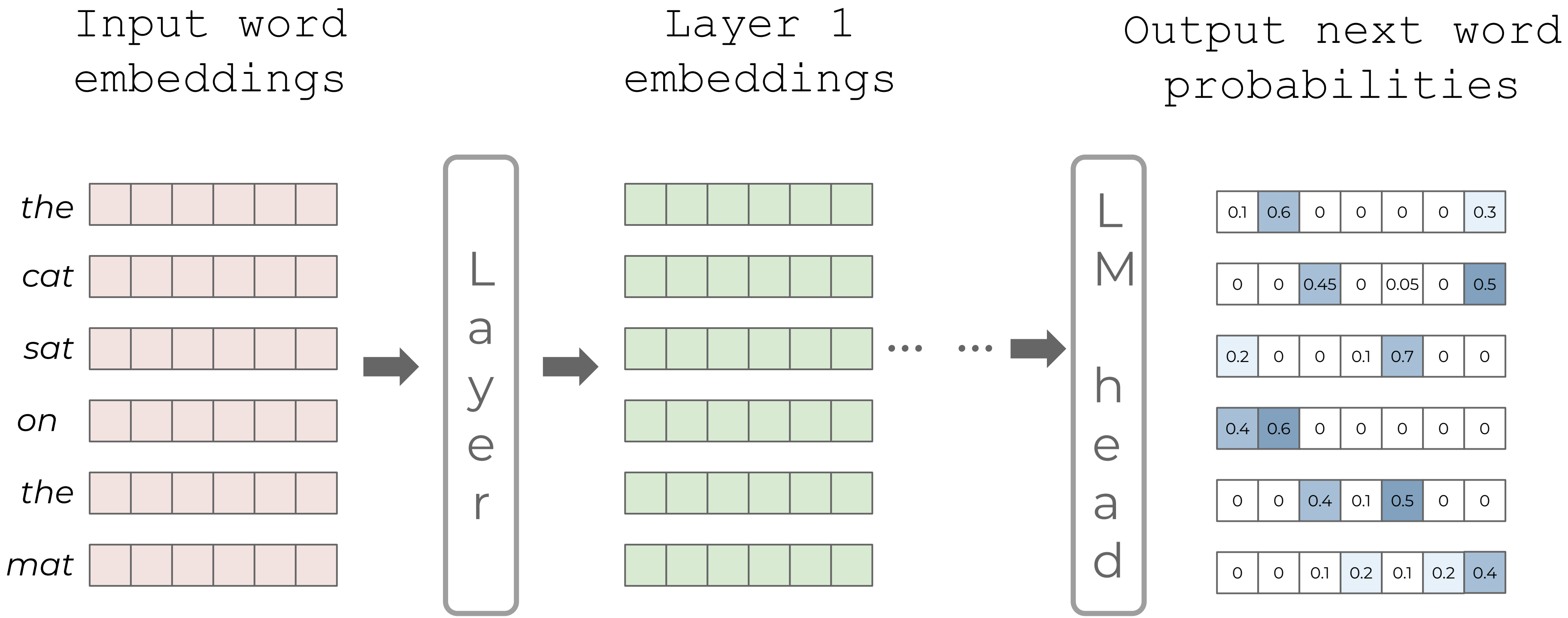
The anatomy of a language model



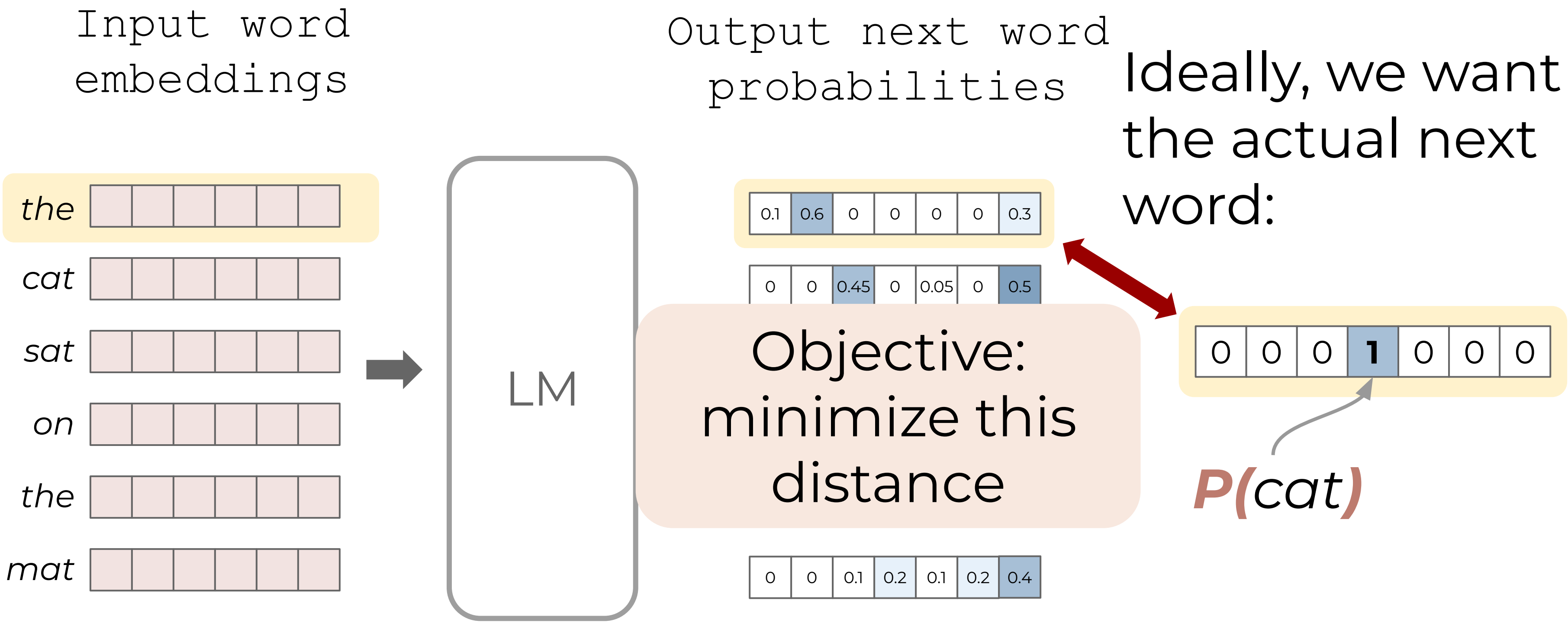
A layer is just operations on vectors

Only lets each vector see the ones **before** it

The anatomy of a language model

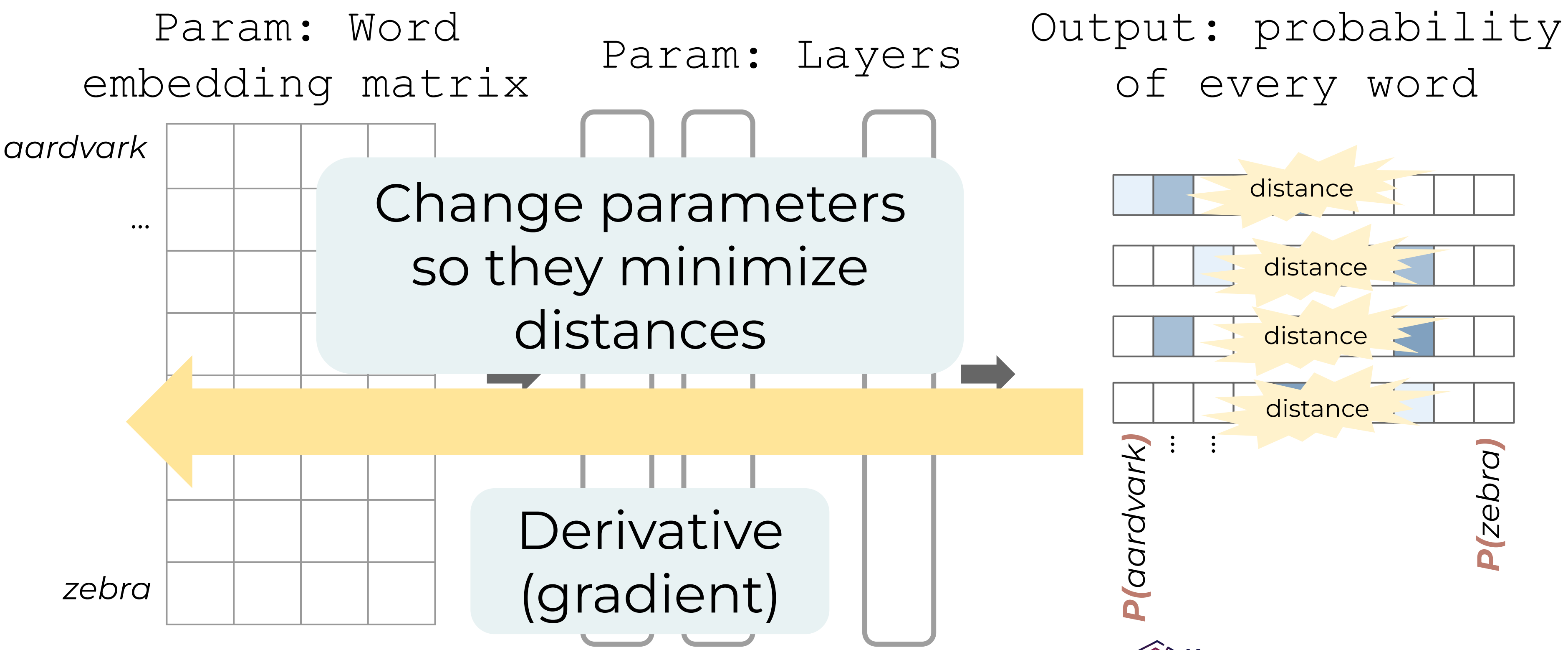


The anatomy of a language model

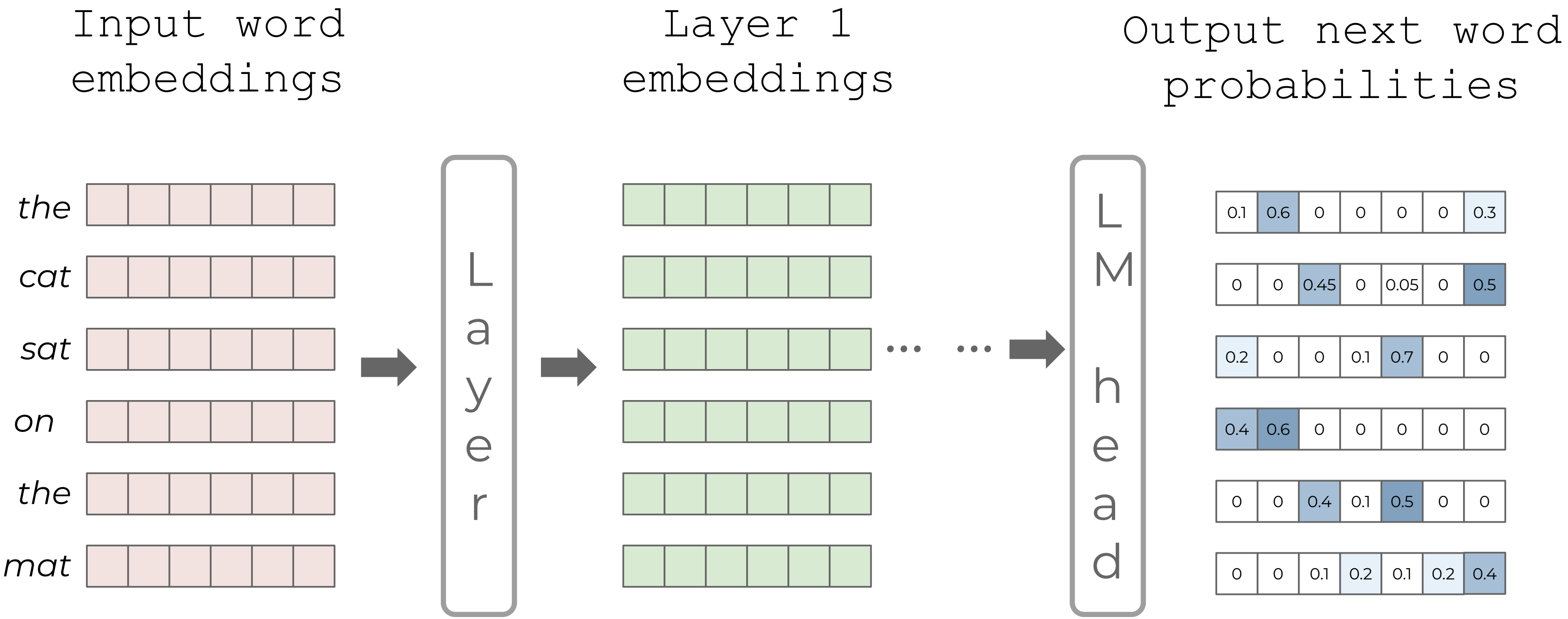


Ideally, we want the actual next word:

Training: change parameters to minimize



The anatomy of a language model



This class:

1) Introduction to vector spaces
2) Word vectors
   - Count-based word vectors
   - Dense word vectors
3) Introducing language models
4) **LM demo**
5) LM discussion and analysis

Language model demo

https://colab.research.google.com/drive/12bS65Yvg8qO6t5--w6a6Mo2GpyV3JyXk?usp=sharing

(you can access this too)

This class:

1) Introduction to vector spaces
2) Word vectors
   - Count-based word vectors
   - Dense word vectors
3) Introducing language models
4) LM demo
5) **LM discussion and analysis**

Why does this simple idea work so well?

- Next-word prediction is difficult to do well
- There really are no shortcuts
    - There's a lot of shortcuts in other tasks, eg sentiment analysis
- It wasn't a task that people used to attempt
- We now have the data, compute, and models to try, and it has revolutionized the field

Discussion: Is next-word prediction realistic?

- How does LM training relate to what **babies** do?
- Social: not trying to predict, trying to **be involved**
- **Grounded**: “look at the doggy!”
- Much **less data**: 10 million vs trillions of words
  - But: what about **replay** in humans?

How do LMs learn different aspects of language?

- Every* piece of text is created by a human who:
  - Has a **grammar** system
  - Knows the real world and **meaning**
  - Is writing with a **communicative intent**
  - Is writing in a **social context**
- Implicit information

Short primer on LM interpretability

(my research!)

- Two main approaches:
  - Look at those layer embeddings: when are they close/far, and why?
  - Intervene on training: what is necessary or sufficient in different cases?
- Do language models learn and represent language like we think humans do?

Discussion: can we learn something about language?

- This is kind of controversial
- Language models are not the human brain
- But we can learn about:
  - The **information** in language
  - General **learnability** under different conditions
  - **Possibilities** for how it can be done!

This class:

1) Introduction to vector spaces
2) Word vectors
   - Count-based word vectors
   - Dense word vectors
3) Introducing language models
4) LM demo
5) LM discussion and analysis

Thanks!