

Handling Text

Manoel Horta Ribeiro
manoel@cs.princeton.edu



MENU

APPETIZER
Textual Data

MAIN COURSE
Bag of Words

DESSERT
Modern Embeddings



MENU

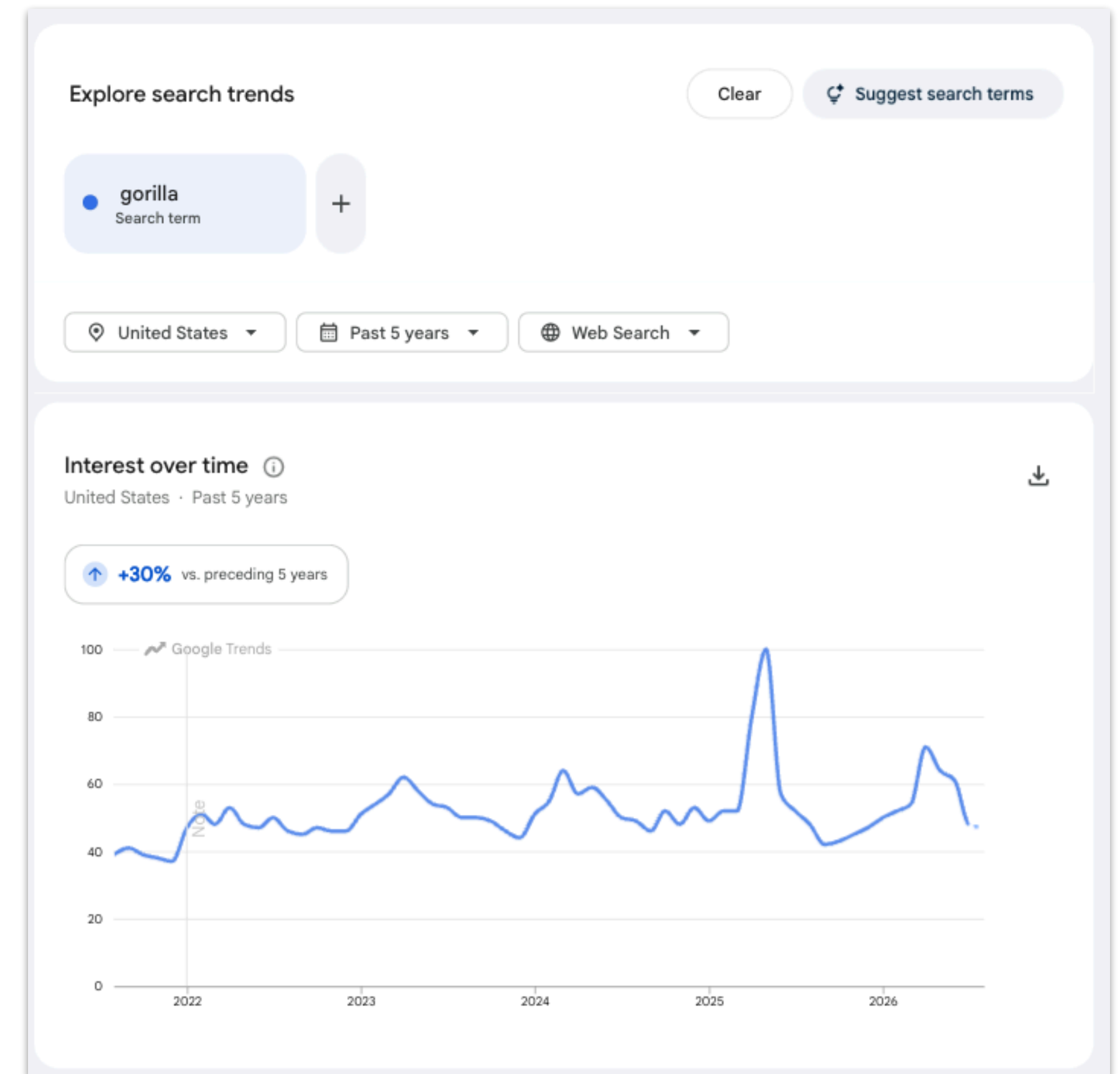
APPETIZER
Textual Data

MAIN COURSE
Bag of Words

DESSERT
Modern Embeddings

Textual data

- Much of modern data is unstructured text:
 - Web, social media, news...
- Frequently, “clean” datasets can be derived from “dirty” textual data”
 - e.g., search queries are short texts Google Trends time series for concepts (e.g., [Q36611 Gorilla](#)) aggregate search queries referring to the concept (e.g., “gorilla”, “big black Rwandan apes”)



How is text data encoded?

- Simple: ASCII
 - 128 characters, 7 bits per character;
 - Essentially a look-up table;
- $A = 65 = 01000001$
- Problems: éá你م❤️
- José → JosÃ©

Bits					Column				Row			
b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀					
0	0	0	0	0	0	0	0	0	1	2	3	4
0	0	0	0	0	0	0	1	NUL	DLE	SP	0	@
0	0	0	0	0	0	1	1	SOH	DC1	!	1	A
0	0	0	0	0	1	0	2	STX	DC2	"	2	B
0	0	0	0	1	0	1	3	ETX	DC3	#	3	C
0	0	0	1	0	0	0	4	EOT	DC4	\$	4	D
0	0	1	0	0	0	1	5	ENQ	NAK	%	5	E
0	0	1	0	1	0	0	6	ACK	SYN	&	6	F
0	0	1	0	1	1	0	7	BEL	ETB	'	7	G
0	1	0	0	0	0	0	8	BS	CAN	(	8	H
0	1	0	0	0	1	0	9	HT	EM	)	9	I
0	1	0	0	1	0	1	10	LF	SUB	*	:	J
0	1	0	1	0	0	0	11	VT	ESC	+	;	K
0	1	0	1	0	1	0	12	FF	FS	,	<	L
0	1	0	1	1	0	0	13	CR	GS	—	=	M
0	1	1	0	0	0	0	14	SO	RS	.	>	N
0	1	1	0	0	1	0	15	SI	US	/	?	O

UTF-8 and Unicode

- **Unicode** assigns every character a unique “*code point*”
 - $A \rightarrow U+0041$
 - $\acute{e} \rightarrow U+00E9$
 - 你 $\rightarrow U+4F60$
 - 😊 $\rightarrow U+1F60A$
- **UTF-8** specifies how to store those numbers as bytes
 - $A \rightarrow 41$
 - $\acute{e} \rightarrow C3\ A9$
 - 你 $\rightarrow E4\ BD\ A0$
 - 😊 $\rightarrow F0\ 9F\ 98\ 8A$

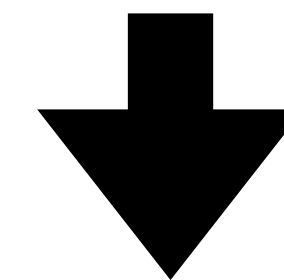
Not all characters use the same number of bytes!

From storing to analyzing

- E.g., suppose we are trying to train a “movie review classifier”
- We don’t have clear feature vectors!
Representation not amenable to ML methods!

The movie was surprisingly good...	+
This movie absolutely sucked...	-
...	...

The movie was surprisingly good...



54 68 65 20 6D 6F 76 69 65 ...



MENU

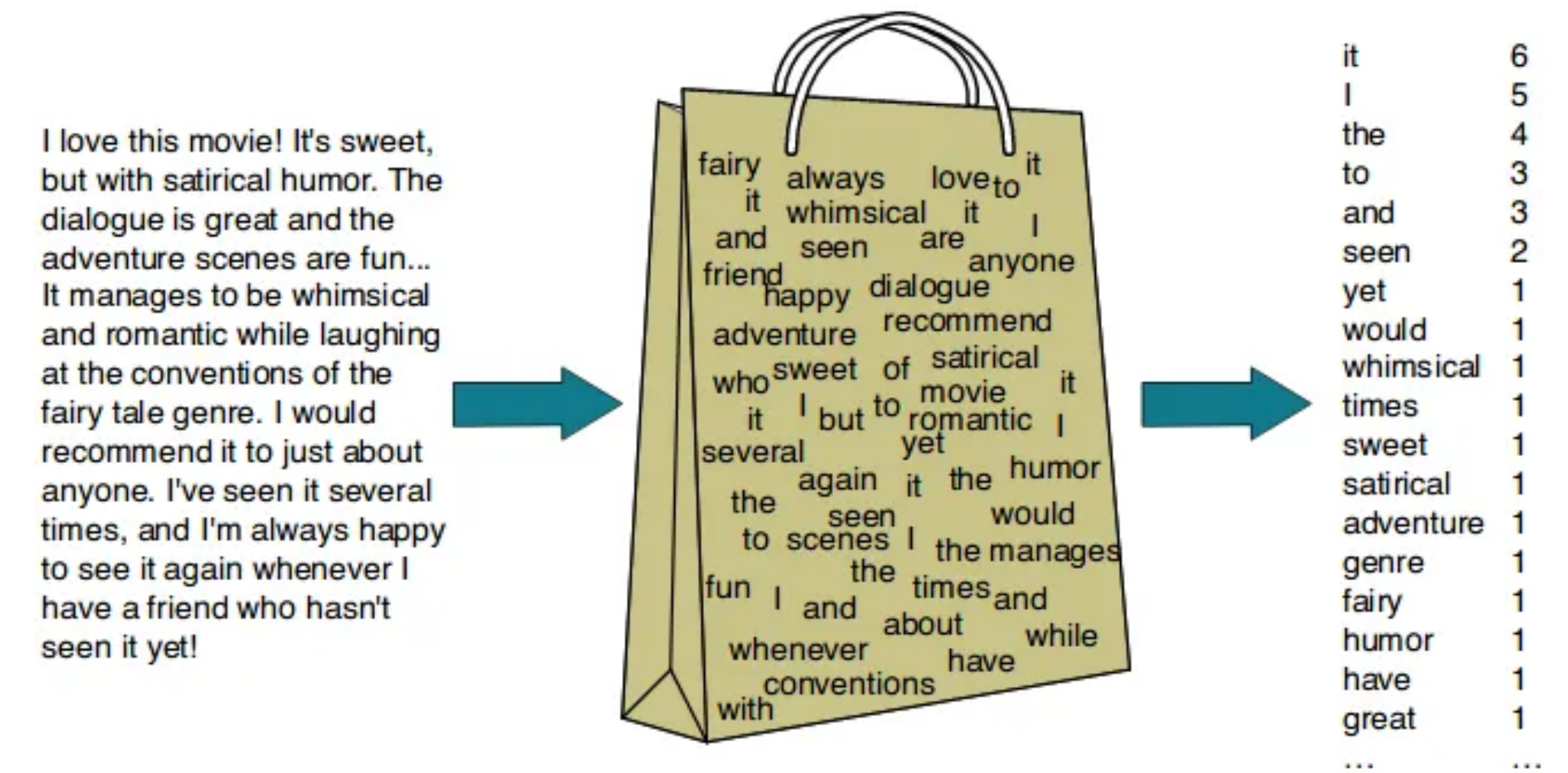
APPETIZER
Textual Data

MAIN COURSE
Bag of Words

DESSERT
Modern Embeddings

“Bag-of-words” matrix

- Bag = multiset
 - “Multi-”: *keep the multiplicity of words*, “-set”: *don’t keep the order!*
- Fixed-length representation:
 - Vector with one entry for each unique word in the vocabulary
 - Bag-of-word vectors are very high dimensional (1e6ish) and sparse



Tokenization

- **Tokenization:** maps a character string into a sequence of tokens ($\approx$ words)
- Surprisingly consequential
 - *New York* $\rightarrow$ *_New York_* or *_New_York_*
 - *Word-level* $\rightarrow$ *enormous vocabulary!*
 - *Character-level* $\rightarrow$ *enormous sequences!*
- Tokens that correspond to meaningful linguistic units are often more useful. (Even for LLMs)

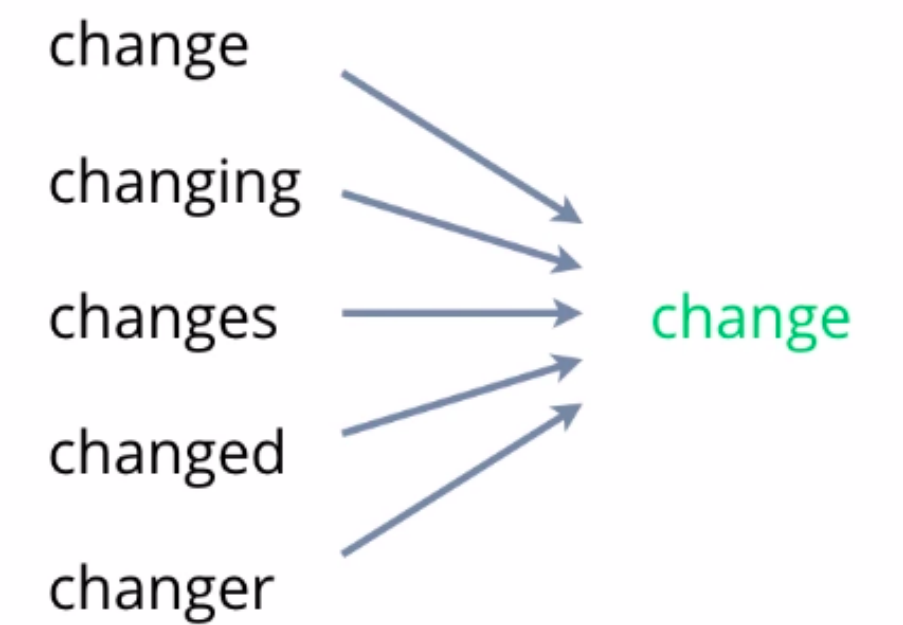
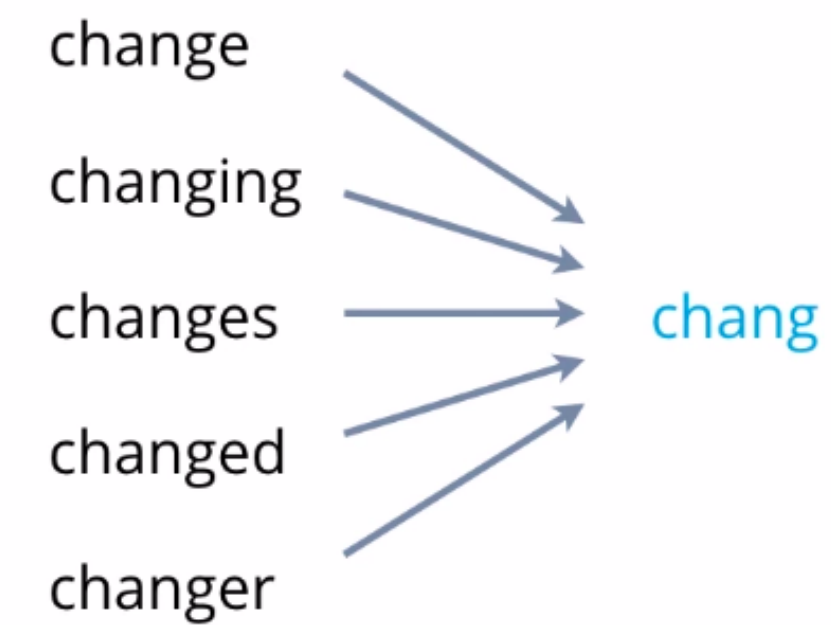
Large Language Models (LLMs), such as GPT-3 and GPT-4, utilize a process called tokenization. Tokenization involves breaking down text into smaller units, known as tokens, which the model can process and understand. These tokens can range from individual characters to entire words or even larger chunks, depending on the model. For GPT-3 and GPT-4, a Byte Pair Encoding (BPE) tokenizer is used. BPE is a subword tokenization technique that allows the model to dynamically build a vocabulary during training, efficiently representing common words and word fragments. Although the core tokenization process remains similar across different versions of these models, the specific implementation can vary based on the model's architecture and training objectives.

LLMs use BPE (byte-pair encoding), a tokenization scheme learned from a training corpus.

Word Normalization

- Different strings can represent essentially the same thing.
 - *Dogs* → *dog*; *Running* → *run*; *Don't* → *do not*
- Common approaches: lowercasing, stemming, lemmatization.

Stemming vs Lemmatization



Word Normalization

- Normalization throws information away!
 - *US → us; Apple → apple; Polish → polish!*
- Why do it, then?
 - *Reduces the number of distinct features*
 - *Combines evidence across related forms*
- Normalization trades information for statistical efficiency!

**We loved the actors! We
really enjoyed the movie.**

**I love the acting and really
enjoy this movie.**

Stopword removal

- Often, “small words” carry little information and draw out information!
- **Idea:** removing stop words!
 - *Using a list of words (many such online!)*
 - *Using words that appear in at least $p\%$ of the documents*

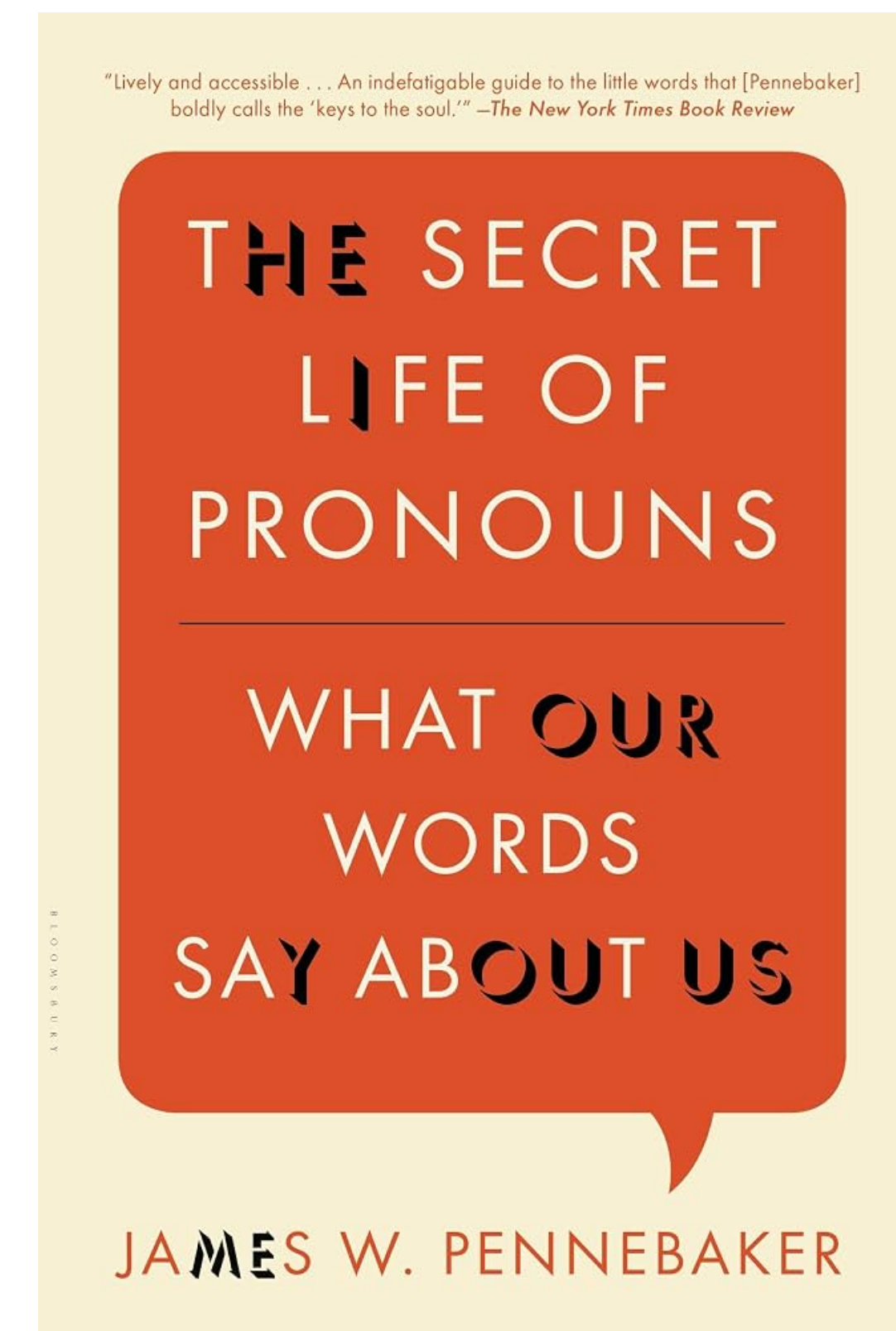
The movie was dull **and** predictable.

The movie was **exciting** **and** **original**.

Exciting, original,
absolutely thrilling!

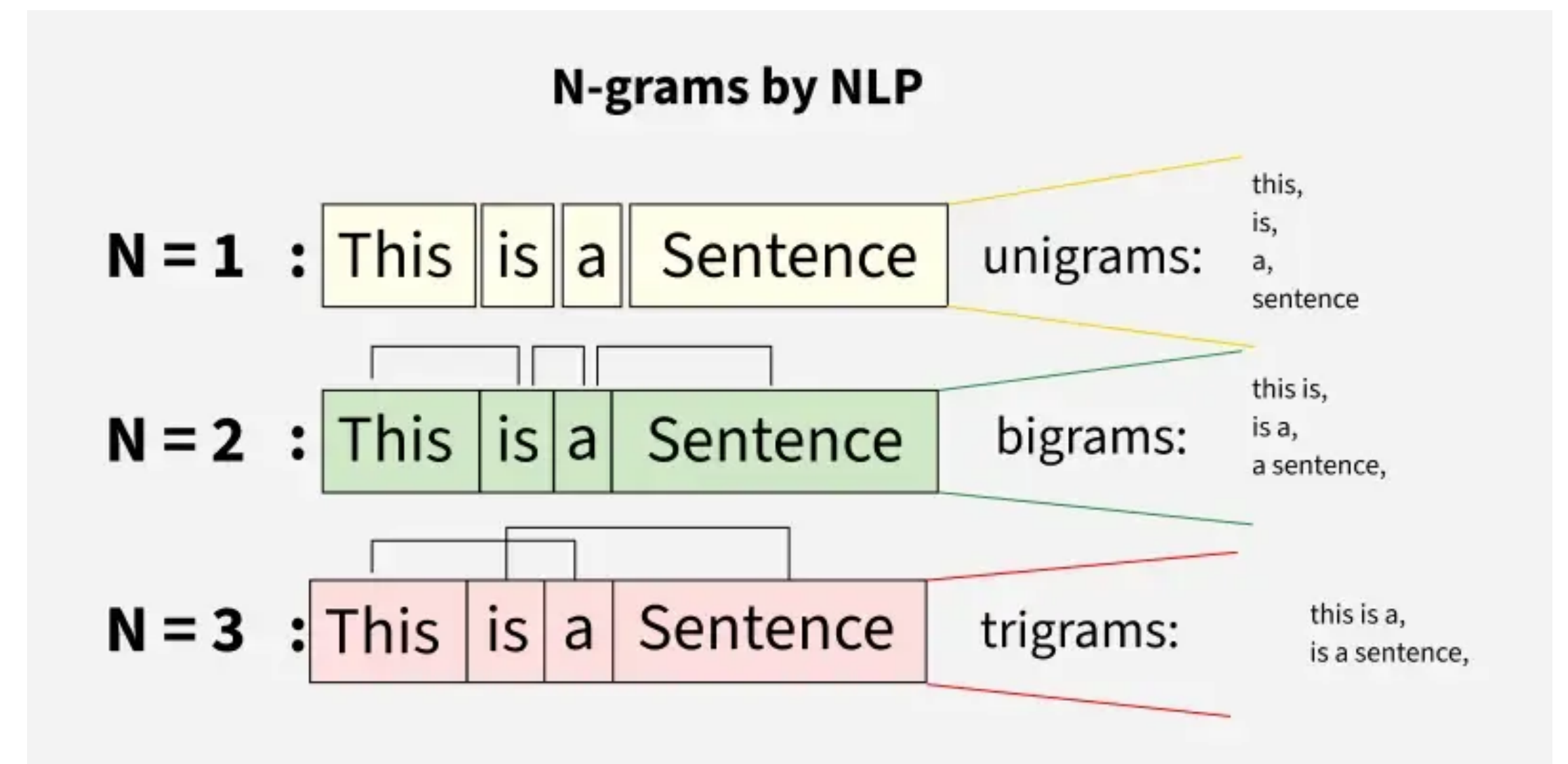
Don't throw out the baby with the bathwater!

- The least noticeable words (pronouns) are most psychologically revealing!
- Patterns in words such as **I**, **we**, **you**, **the**, **a**, **but**, and **because** correlate with social relationships, status, personality, emotional state, deception, and more.



n -grams

- Bag-of-words ignores word order!
 - Dog bites man = Man bites dog
- n -grams are sequences of consecutive tokens!
 - **Pro:** Capture context!
 - **Con:** Increase sparsity!



Weighting the Matrix

- Basic BoW uses raw counts!
 - **Problem #1:** longer documents will have larger counts!
 - **Problem #2:** common words dominate but are less informative!
- **Idea:** give more weight to more distinctive words

Bag of Words Matrix

Sentences	i	the	to	and	in	for	here	that	at	who
Sentence 1	0	0	0	0	0	0	0	0	0	0
Sentence 2	1	0	1	0	1	0	0	1	0	0
Sentence 3	0	0	0	0	0	0	0	1	0	0
Sentence 4	0	0	0	1	0	0	0	0	1	0
Sentence 5	0	0	0	0	0	1	0	1	0	0
Sentence 6	1	1	1	1	1	0	1	0	0	1
Sentence 7	1	1	0	1	1	0	1	0	0	1
Sentence 8	0	0	1	1	0	0	0	0	0	0
Sentence 9	1	1	1	1	0	1	1	0	1	0
Sentence 10	1	1	1	1	0	0	0	1	1	0
Sentence 11	0	1	0	0	1	1	1	0	0	1
Sentence 12	1	1	1	0	1	0	0	0	0	0
Sentence 13	1	1	0	0	0	0	0	0	1	0

Frequent Words

Term Frequency-Inverse Document Frequency

- Weigh words inversely proportional to how many documents they appear in!

$$\underbrace{TF(t,d)}_{f_{t,d}} \cdot \underbrace{IDF(t,d)}_{\log N/n_t}$$

- $f_{t,d}$: frequency of token t in a document d
- N : number of documents in the matrix
- n_t : number of documents where t occurs

Bag of Words Matrix

Sentence 1	0	0	0	0	0	0	0	0	0	0
Sentence 2	1	0	1	0	1	0	0	1	0	0
Sentence 3	0	0	0	0	0	0	0	1	0	0
Sentence 4	0	0	0	1	0	0	0	0	1	0
Sentence 5	0	0	0	0	0	1	0	1	0	0
Sentence 6	1	1	1	1	1	0	1	0	0	1
Sentence 7	1	1	0	1	1	0	1	0	0	1
Sentence 8	0	0	1	1	0	0	0	0	0	0
Sentence 9	1	1	1	1	0	1	1	0	1	0
Sentence 10	1	1	1	1	0	0	0	1	1	0
Sentence 11	0	1	0	0	1	1	1	0	0	1
Sentence 12	1	1	1	0	1	0	0	0	0	0
Sentence 13	1	1	0	0	0	0	0	0	1	0
	i	the	to	and	in	for	here	that	at	who

Frequent Words

TF-IDF Example

“The cinematography was gorgeous, but the movie was very slow.”

- the, but, was, very → *Low IDF*
 - *cinematography, gorgeous* → *High IDF*
- Note that TF-IDF handles very frequent words without removing them!

Application #1: Sentiment Analysis

- **Goal:** infer attitudes expressed in a piece of text!
 - Positive 😊
 - Negative 😡
 - Neutral 😐
- A BoW approach can be surprisingly effective!

Text	Sentiment
Just got the job!! Can't believe it 🥳🎉	😊
Going to the train station now	😐
The new iPhone is very neat!	😊
Worst customer service I've ever experienced!	😡
...	...

Application #1: Sentiment Analysis

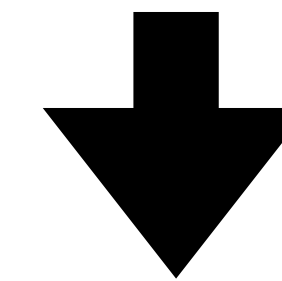
Text	Sentiment		worst	amazing	...	the	Y
Just got the job!! Can't believe it! Amazing!	😊	➡	0	1	...	1	1
Going to the train station now	😐		0	0	...	1	0
The new iPhone is amazing!	😊		0	1	...	0	1
Worst customer service I've ever experienced!	😡		1	0	...	0	-1
...	...		...	...	...		...

Application #2: Document Retrieval

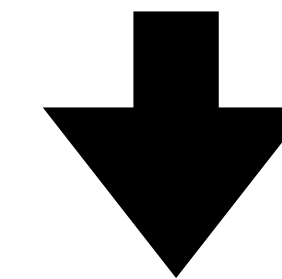
- **Goal:** find precedent in previous legal cases
- Compare “query” vectors with document vectors
- Find the k most “similar” documents

$$\text{sim}_{\text{cos}}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}$$

Car search without warrant



Query vector



Document vector #1

Document vector #2

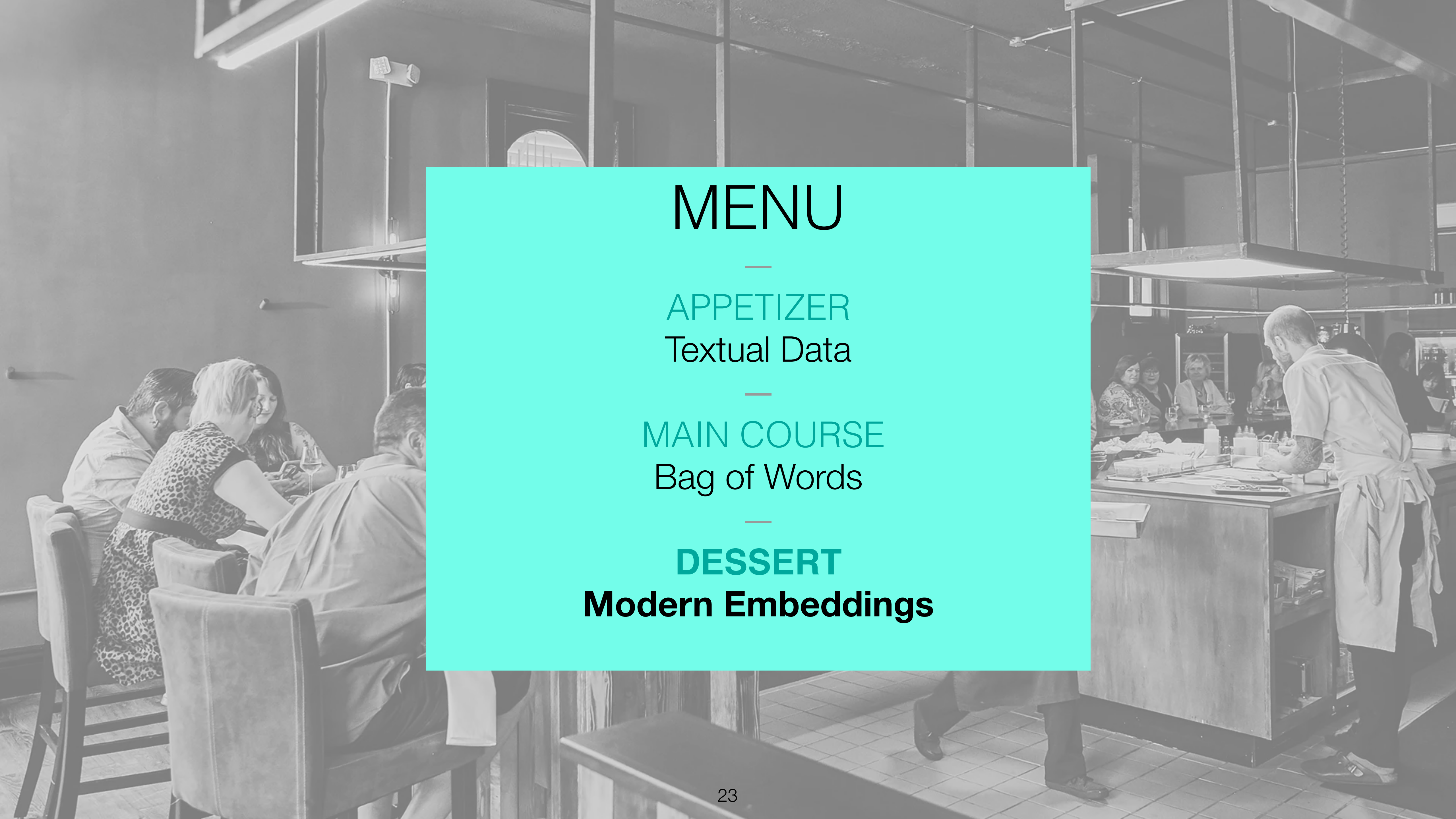
...

Document vector #N

What is missing?

- Bag of words doesn't know what words mean!
 - The movie was excellent
 - The movie was fantastic
 - The movie was horrible
- Improvements to bag of words typically represent meaning geometrically!





MENU

APPETIZER
Textual Data

MAIN COURSE
Bag of Words

DESSERT
Modern Embeddings

Where does meaning come from?

Distributional Hypothesis: Words occurring in similar contexts tend to have similar meanings.

I fed my **dog** this morning

I took my **dog** for a walk

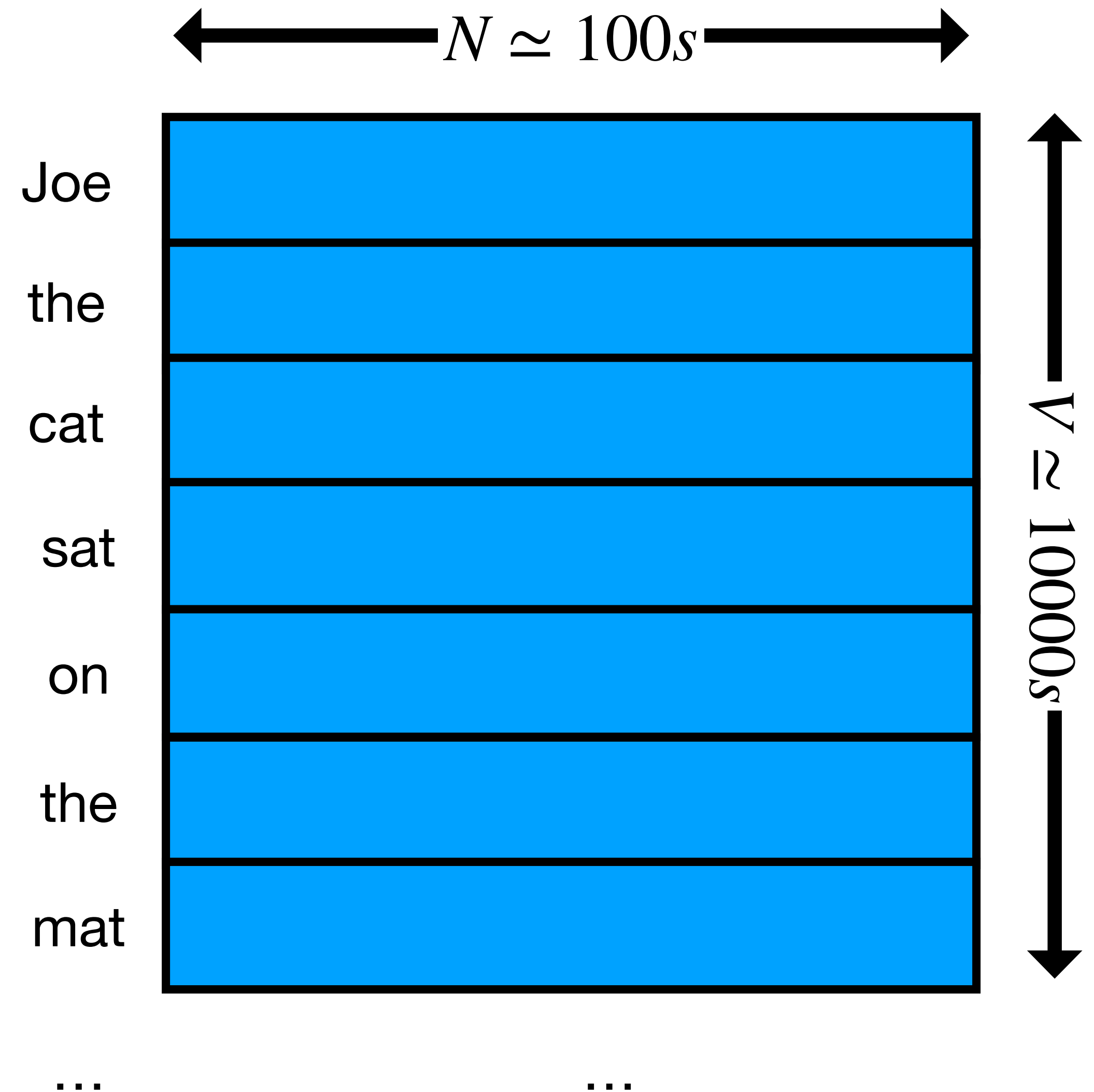
I fed my **puppy** this morning

I took my **puppy** for a walk

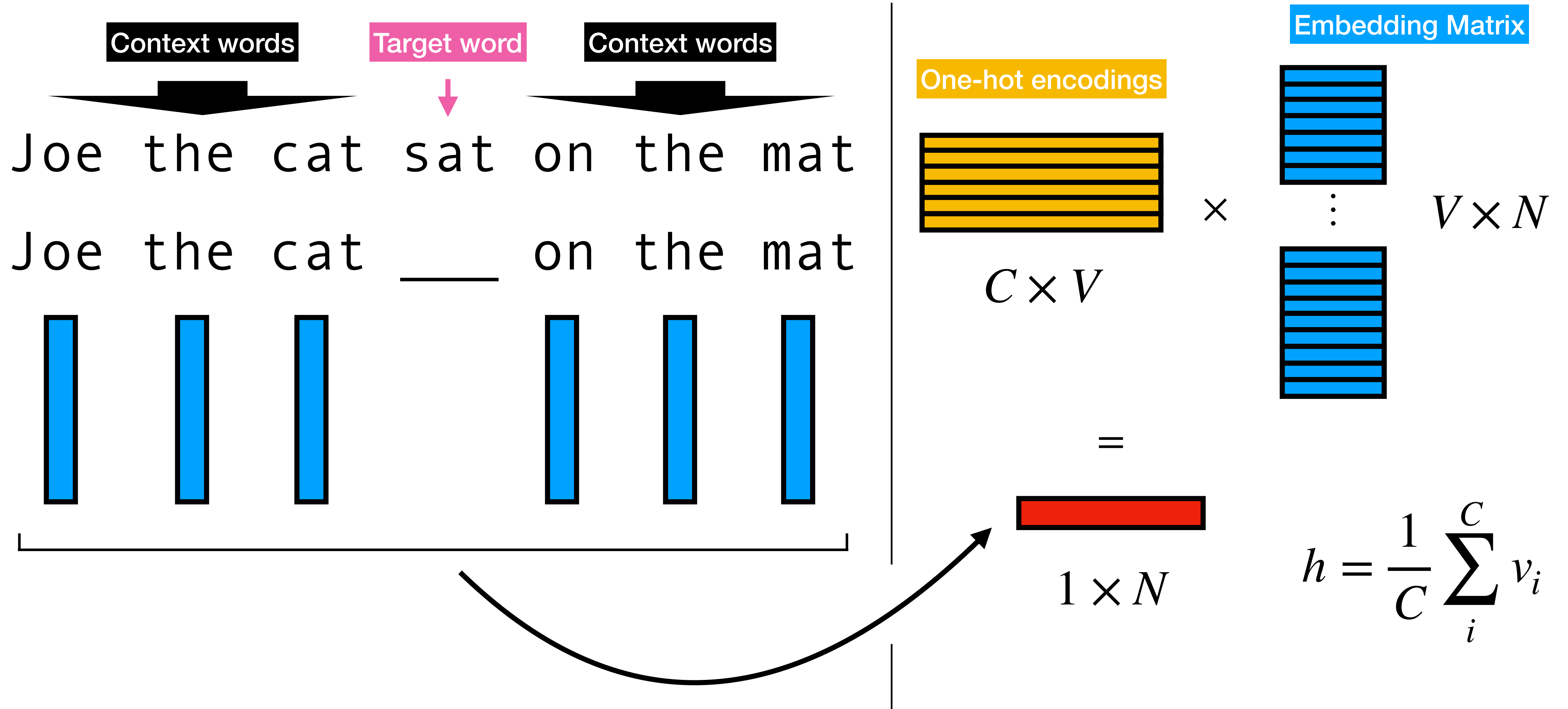
dog and **puppy** appear in similar contexts, therefore, they should have similar representations!

Word2Vec

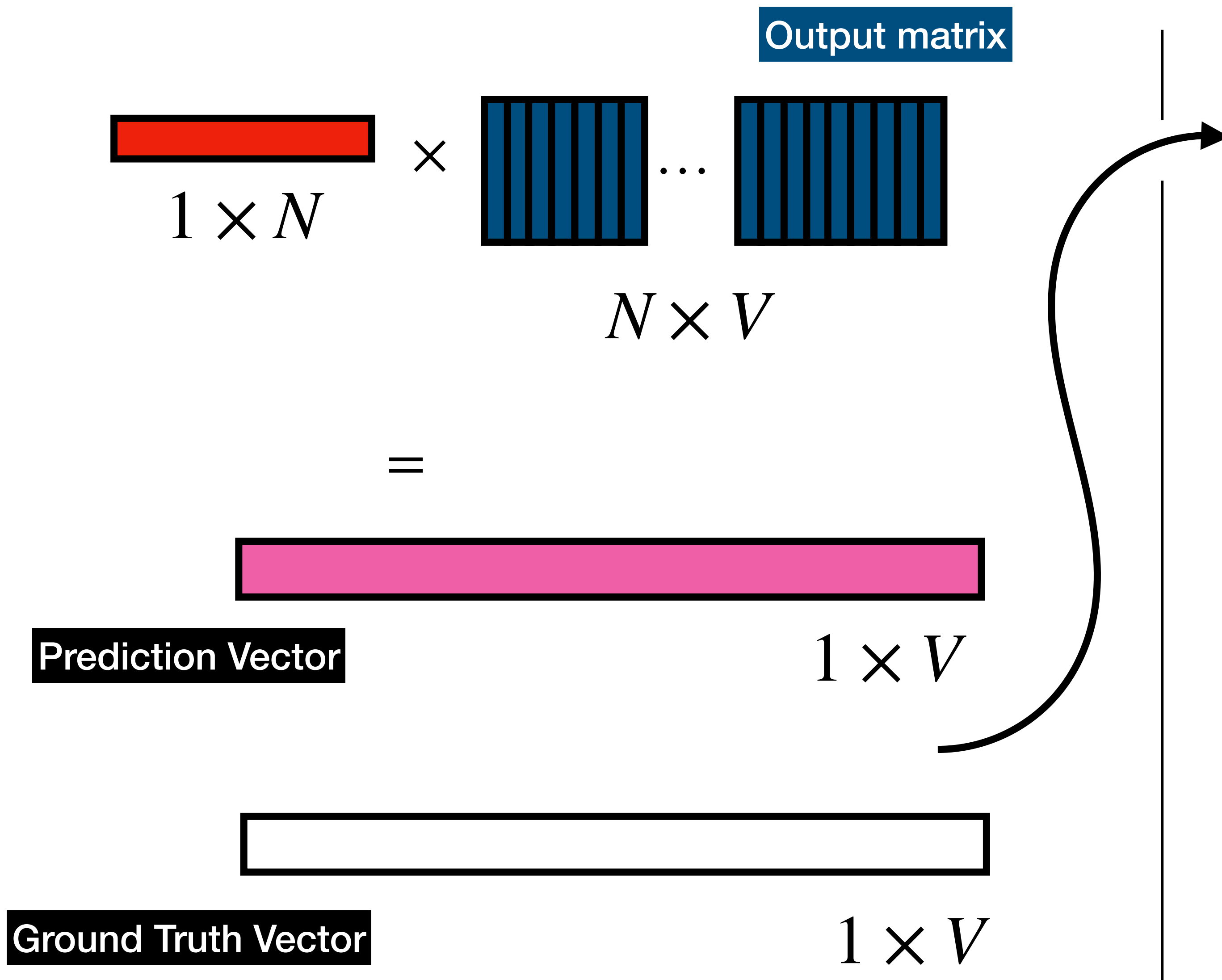
- **Idea:** Learn representation for words (an embedding) from a prediction task!
- **Popular Approach:** CBoW (continuous bag of words!) in Mikolov et al. 2013
- Generates dense embeddings for each word!
- Very simple neural network without non-linearities!



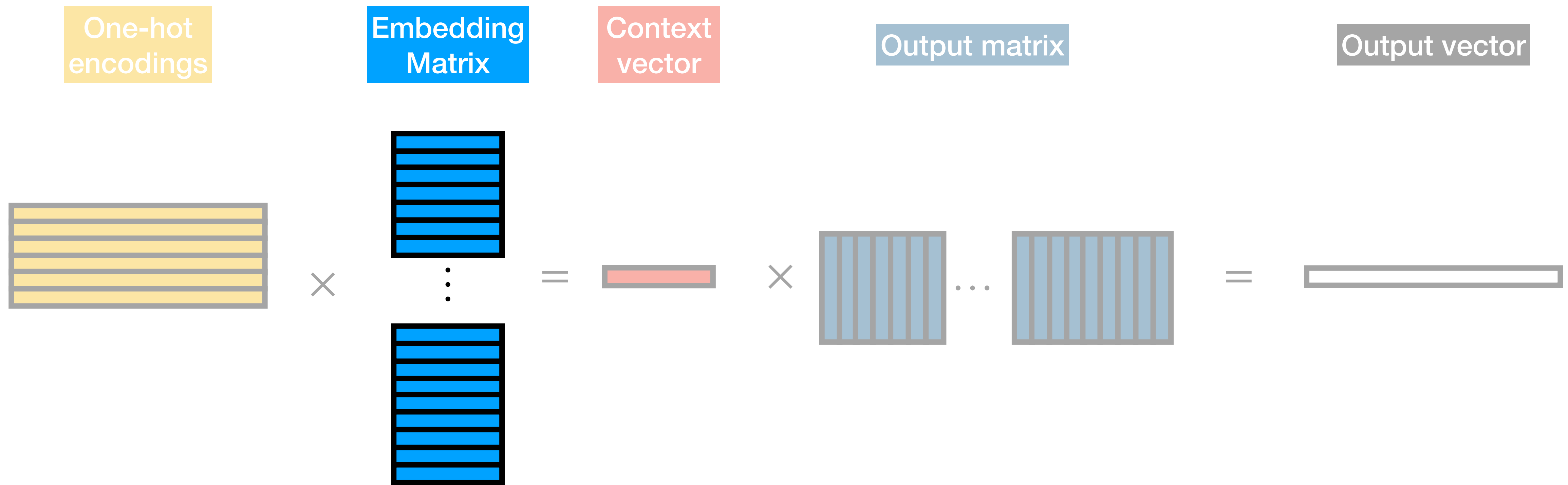
Getting the “context vector”



Getting the prediction vector



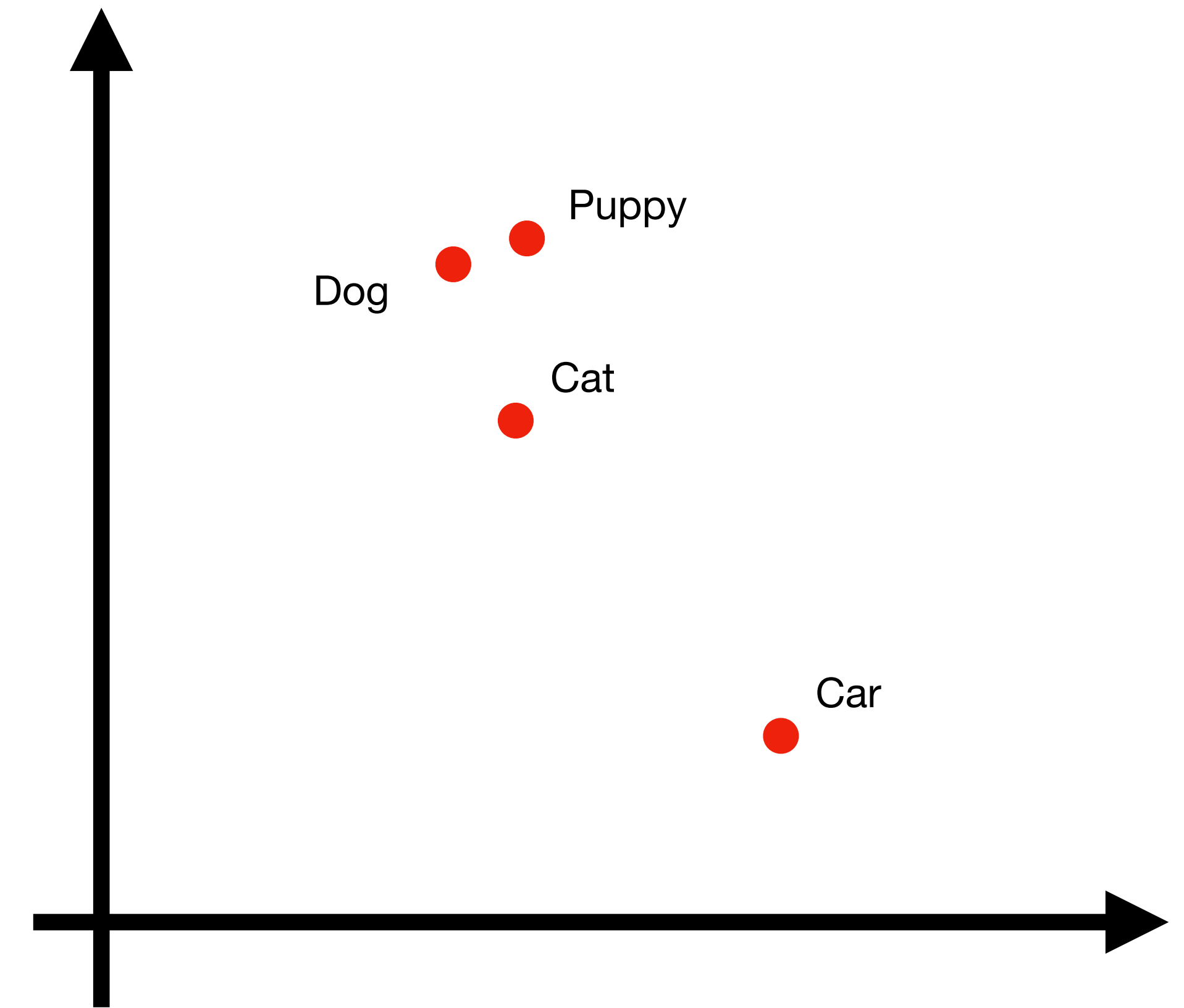
- Loss based on comparison between prediction & ground truth vector (1 only for the target word).
- Learns weights in the **embedding matrix** and the **output matrix** via backpropagation.



Ultimately, we throw everything else away and just keep the embeddings! These capture similarities in how words are used!

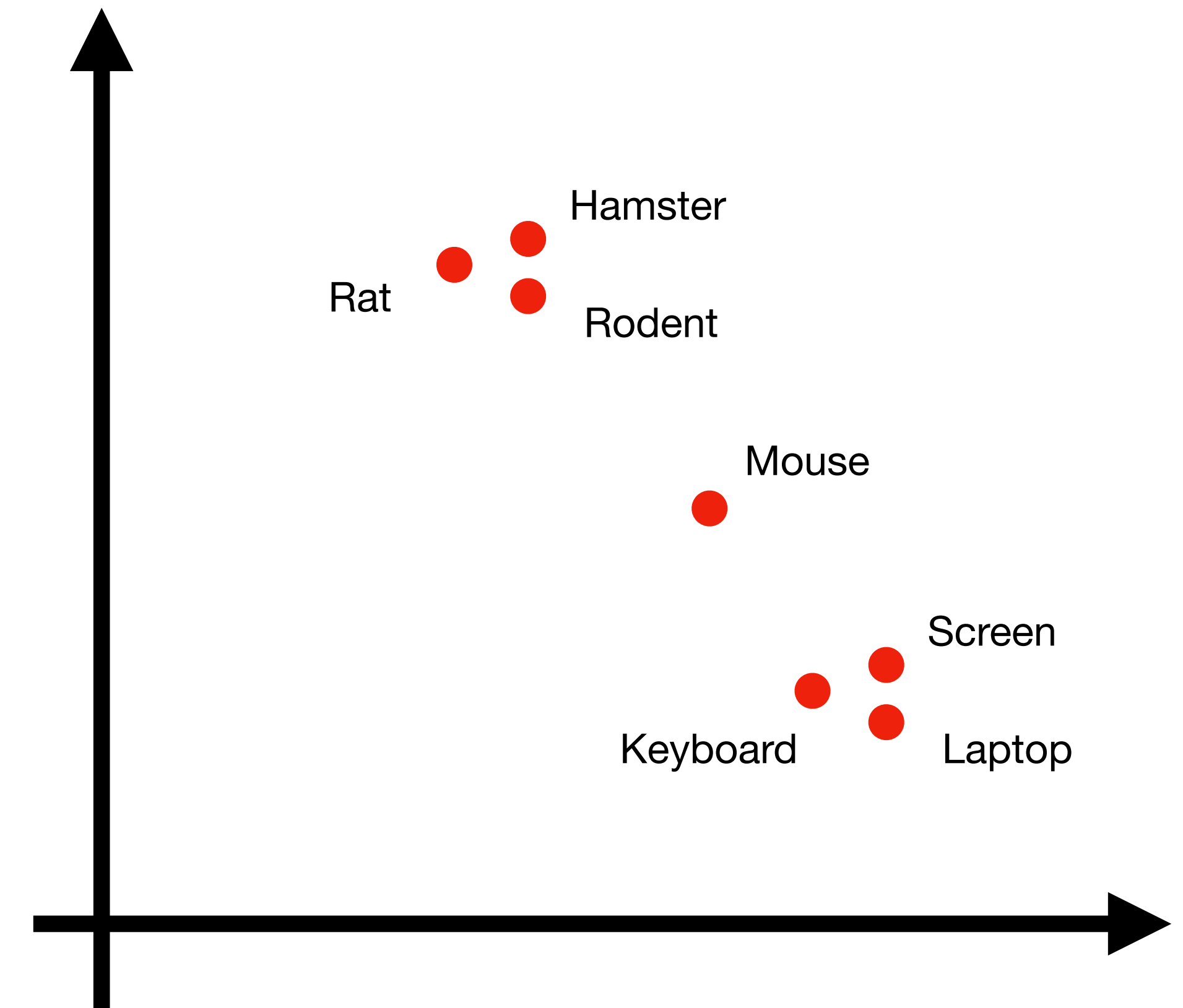
Surprising properties

- Similar words are close together!
- Embeddings also encapsulate meaning!
 - $paris - france + italy \simeq rome$
 - $king - man + woman \simeq queen$
- Using these representations instead of BoW typically leads to higher performance!



What is missing?

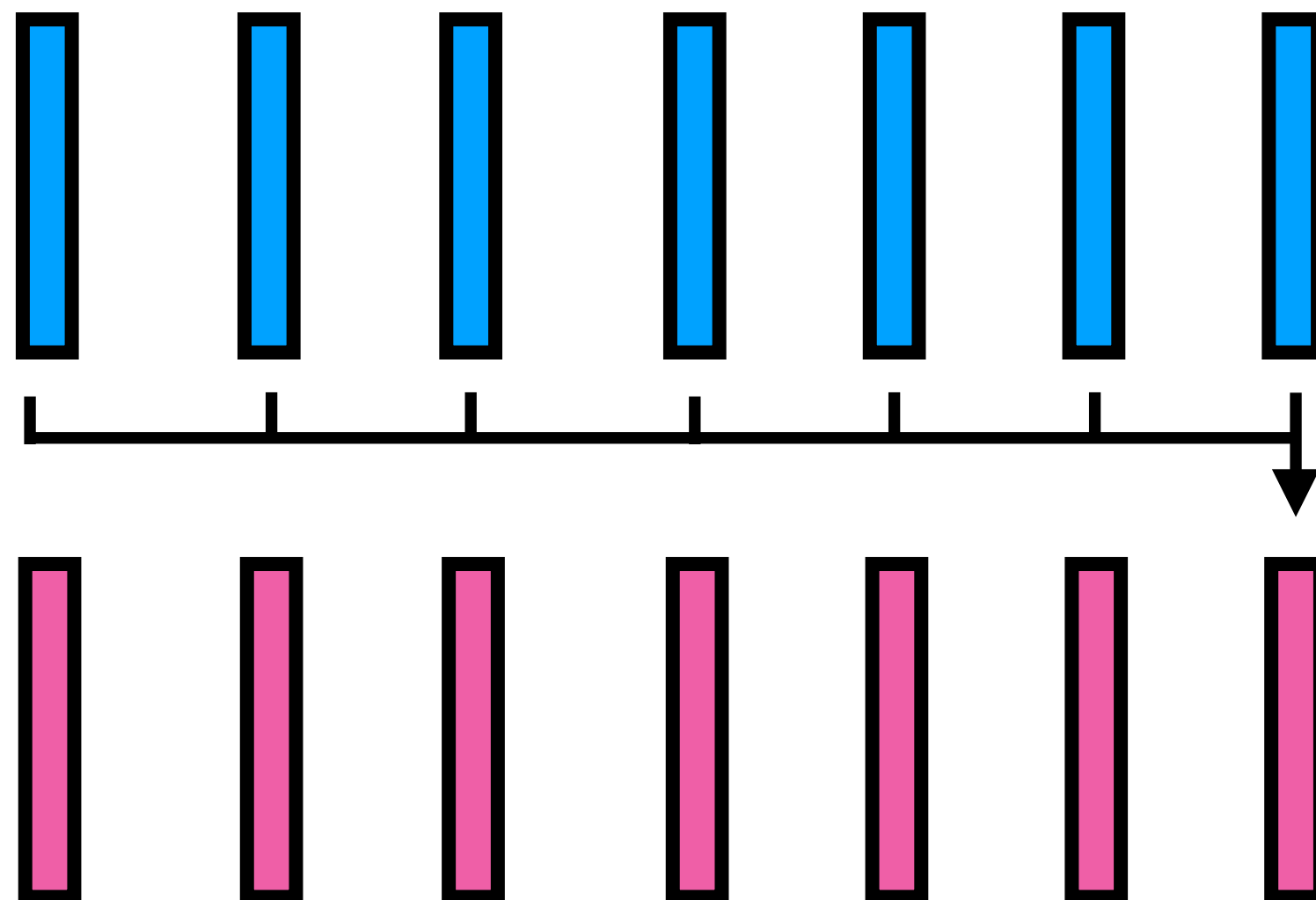
- Each word only gets one embedding! But words have different meanings!
- More broadly, meaning is compositional; a paragraph means more than the set of words it contains!
 - The teacher praised the student.
 - The student praised the teacher.



Modern LLM Embeddings

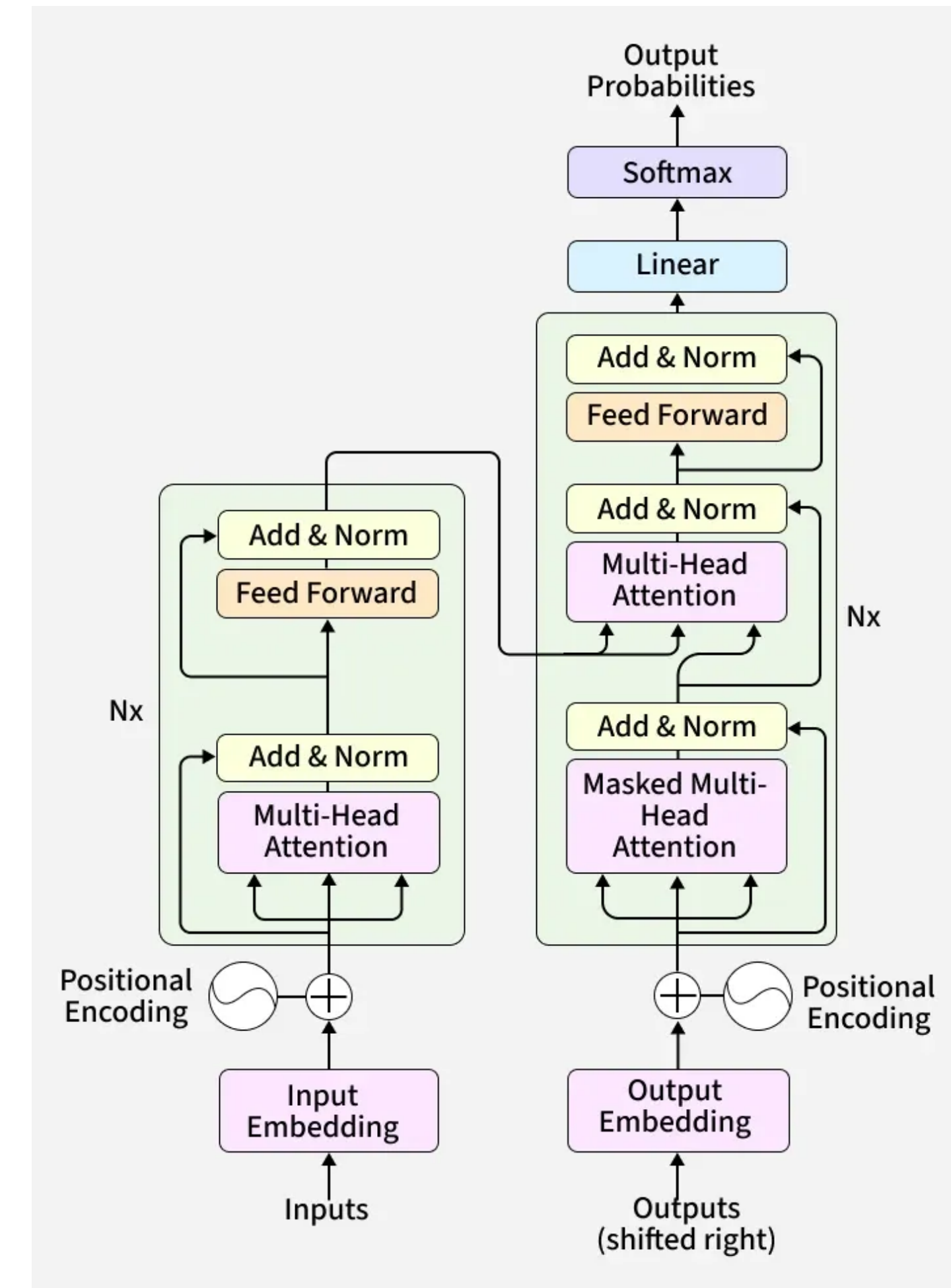
- **Idea:** embed entire pieces of text!
- Modern embedding models use transformers to do this.

Joe the cat sat on the mat



Input
embeddings

Output
embeddings



Modern LLM Embeddings

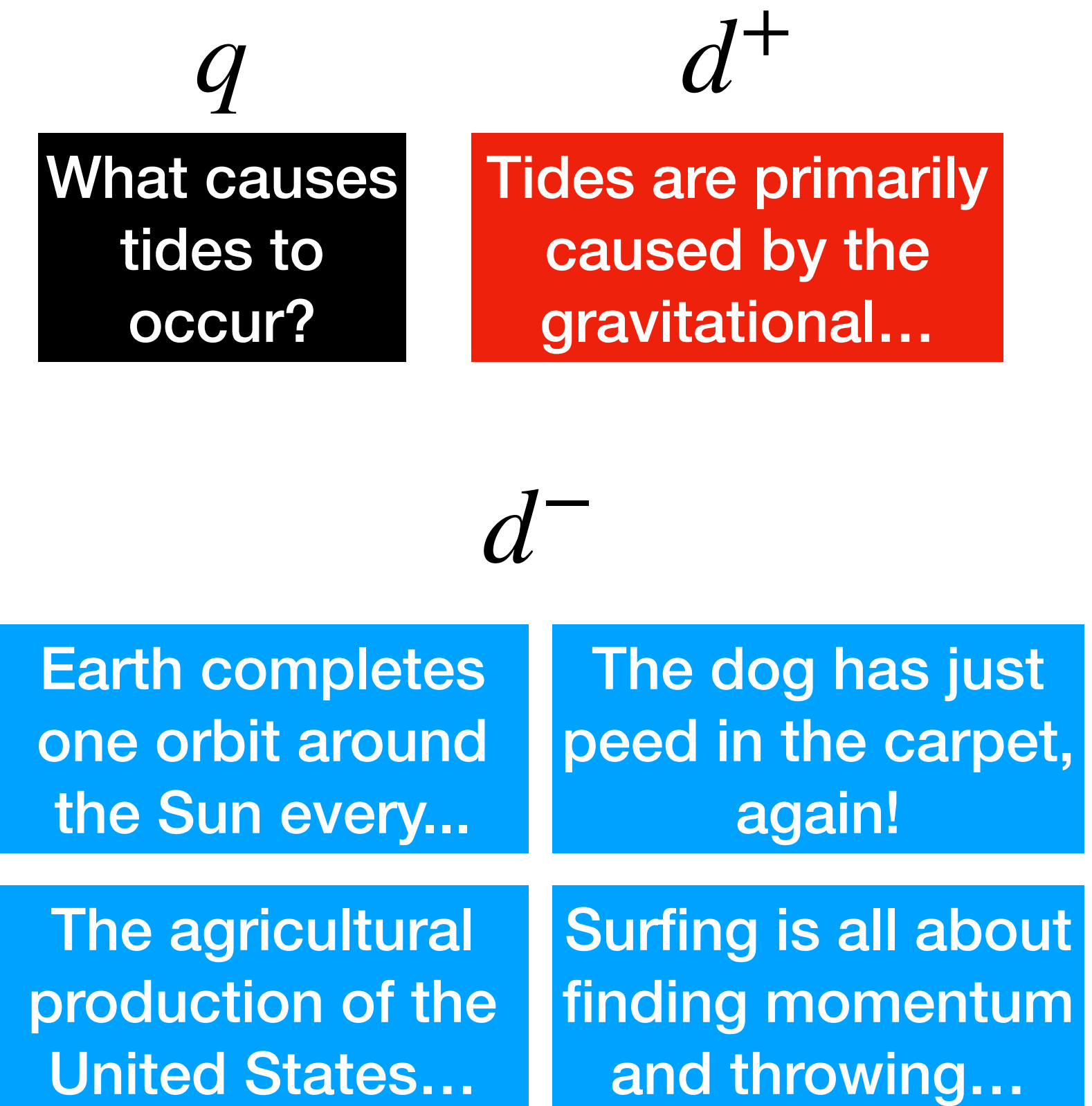
- Typically, foundation models do not yield very good embeddings!

- Qwen3-Embeddings:

- Start from the foundation model
- Create embeddings using various relevant tasks, e.g., retrieval, semantic similarity!

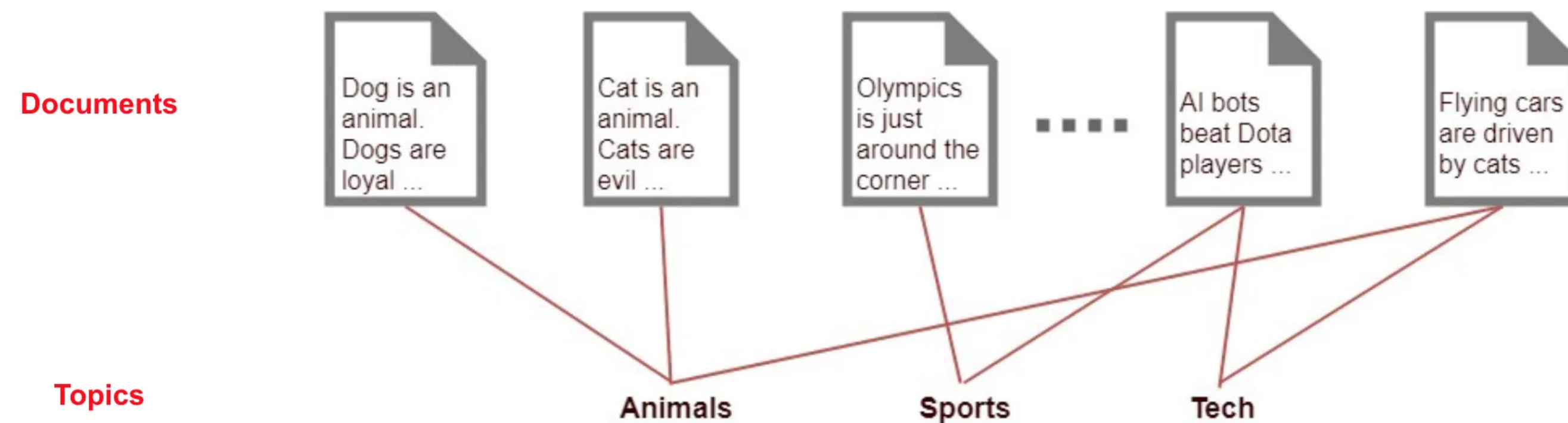
- **Contrastive loss:**
$$-\log \frac{e^{\text{sim}(q, d^+)/\tau}}{e^{\text{sim}(q, d^+)/\tau} + \sum_k e^{\text{sim}(q, d_k^-)/\tau}}$$

- Rewards similarity between q and d^+
- Penalizes similarity between q and d^-



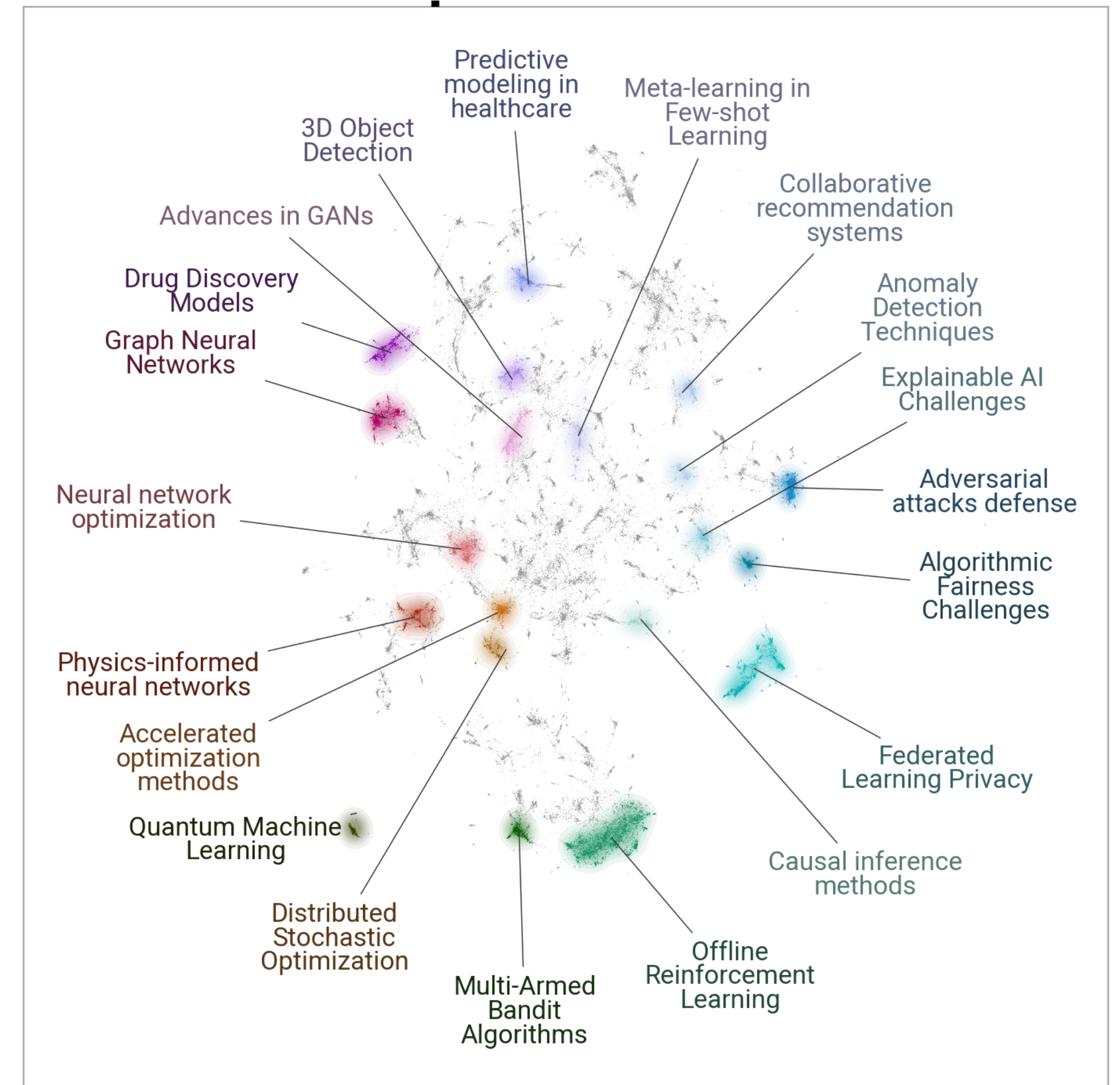
Application #3: Topic Modeling

- **Goal:** discover recurring themes in a large collection of documents
- Suppose we have 100,000 news articles. We want to discover topics like Elections, Ukraine, AI, and movies.
- But we don't have topic labels!



Application #3: Topic Modeling

- **Old approach:** LDA
- **Modern approach:** embeddings!
- **BERTopic:**
 - Embed each document (**BERT**)
 - Do dimensionality reduction (**UMAP**)
 - Do clustering on results (**HDBScan**)
 - Find words that are most distinctive of each cluster (**c-TF-IDF**)





Give us some feedback!



[https://tinyurl.com/
reasoningwithdata](https://tinyurl.com/reasoningwithdata)