LLMs and Words Preview

CS 485, Fall 2026
Applications of Natural Language Processing

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

[slides from SLP3]

- Today: overview basic components of LLMs
- Preview our pathway through other language models and language tasks

Large Language Models

OpenAI GPT-5

ANTHROP\C CLAUDE

R1

Google Gemini

Microsoft® PHI-4

Meta LLAMA

Large Language Models

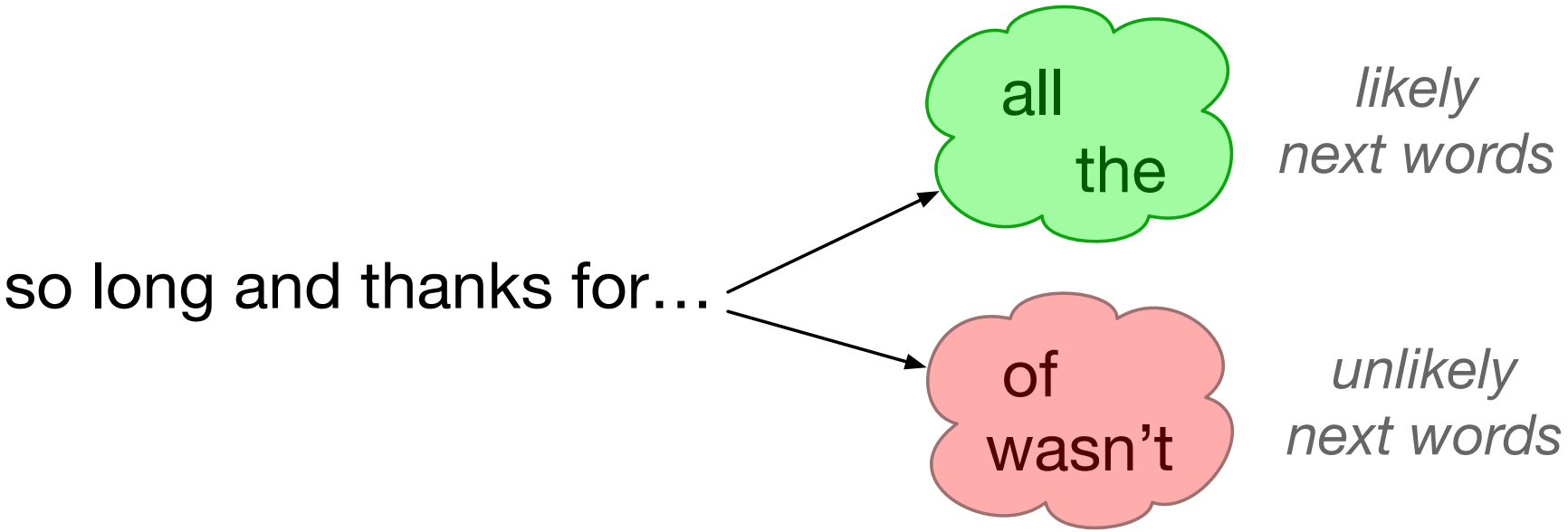
- The good?
- The bad?

What is a language model?

1. An agent that carries on conversations and acts in the world
2. A computational system that can predict the next word from previous words.

- For LMs, we'll usually focus on #2
- Note *analyzing* text with a computational model, may or may not be an "LM"

A language model predicts words

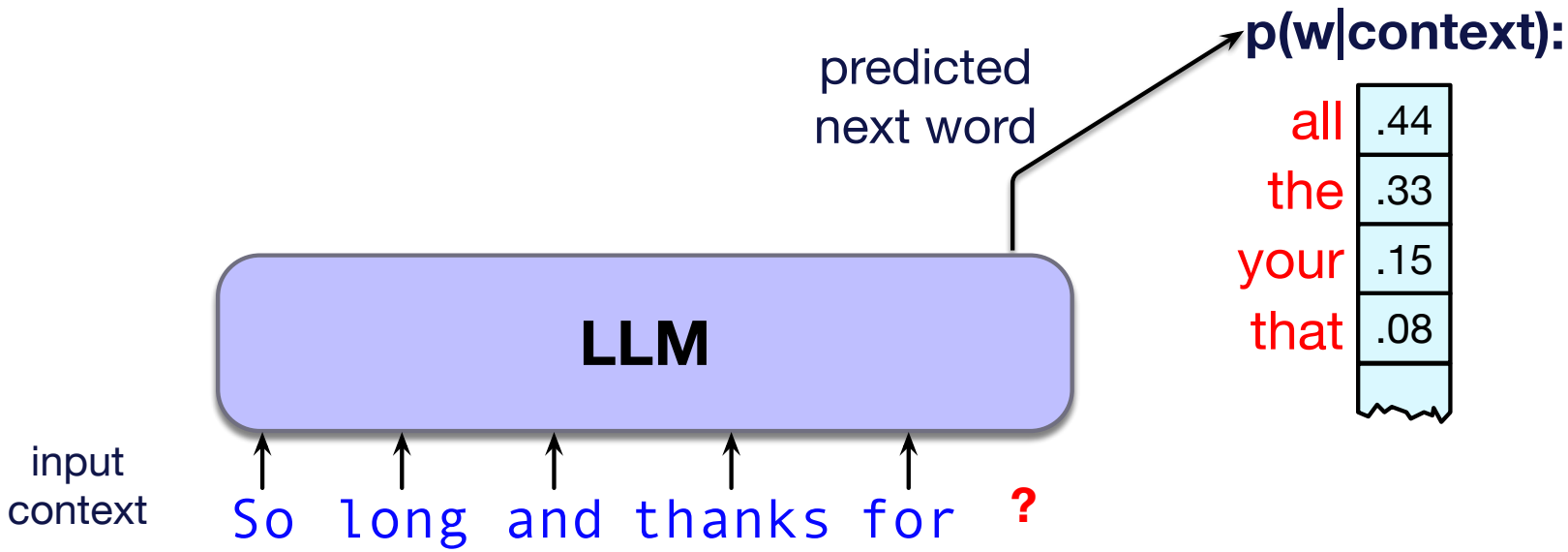


What is a large language model?

A neural network with:

Input: a context or prefix,

Output: a distribution over possible next words



Intuition of generation

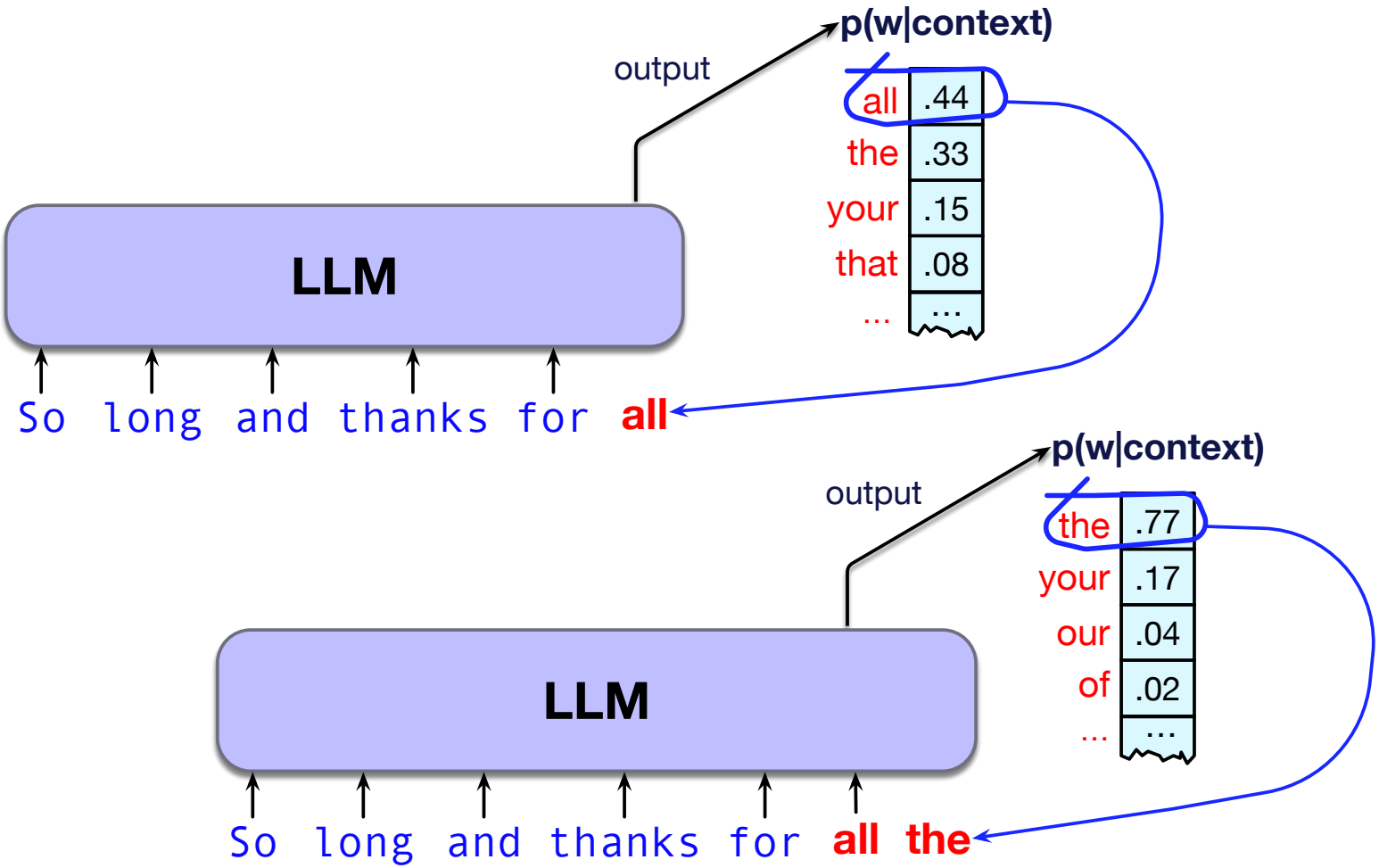
An LLM that can predict text

- assigning a probability distribution over upcoming words

Can generate words

- by **sampling** from the distribution

LLMs can generate!

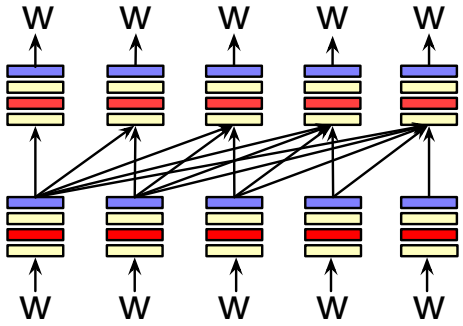


Causal or autoregressive model

Models that iteratively predict and generate words left-to-right from earlier words

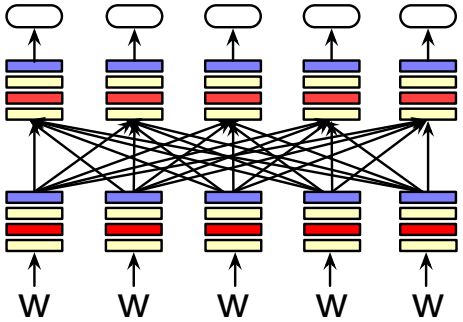
The most common LLMs

Three architectures for large language models



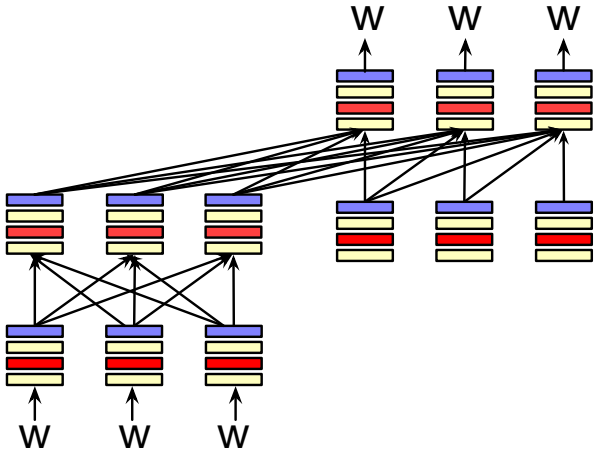
Decoders

GPT, Claude, Llama
Mixtral



Encoders

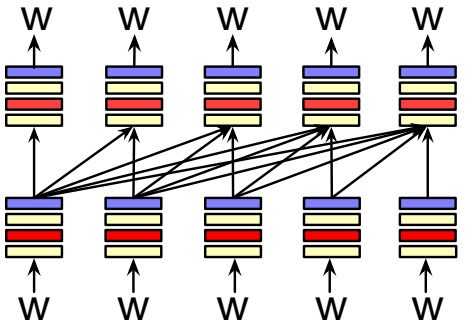
BERT family, HuBERT



Encoder-decoders

Flan-T5, Whisper

Decoders



What most people think of when we say LLM

- GPT, Claude, Llama, DeepSeek, Mistral
- A generative model
- It takes as input a series of tokens, and iteratively generates an output token one at a time.
- Left to right (causal, autoregressive)

Language models process **tokens**, not words

A **token** is a word or word part.

First LLM step is converting text into tokens

- Via tokenization algorithms like **BPE**

English tokens correspond roughly to words:

```
Anyhow,·she's·seen·Jane's·224123·flowers·anyhow!
```

German tokens tend to be smaller than words

```
Ist·die·Trägheit·eines·Körpers·von·seinem·
Energieinhalt·abhängig?·von·A.·Einstein.
```

- What can a model learn from a collection of texts??

Fundamental intuition of large language models

Text contains enormous amounts of knowledge

Pretraining on lots of text with all that knowledge is what gives language models their ability to do so much

Intuition from humans

Vocabulary size of young adult speakers of American English: 30,000 to 100,000 words

Children learn **7 to 10 words a day** to get to that level by 20 years old

How do they do it?

Reading! (Laundauer and Dumais 1997)

Reading isn't passive

It's rich contextual processing

At some points during learning, rate of vocabulary growth exceeds the rate of seeing new words

Every time we read a word, we are **strengthening our understanding of other associated words**.

The distributional hypothesis

Suppose you didn't know "difficult" in these sentences:

It was difficult to make it up

What a difficult decision!

But you saw "hard" in similar contexts (hills, decisions):

I found climbing the steep hill hard.

It was a very hard decision.

You could infer that "difficult" meant something like "hard"!

Pretraining

We teach language models to predict upcoming words, trillions of times

This is called pretraining

By doing so they induce knowledge

What does a model learn from pretraining?

- There were roses, dahlias, and peonies, I was simply surrounded by flowers
- The room wasn't just big it was enormous
- The square root of 4 is 2
- The author of "A Room of One's Own" is Virginia Woolf
- The doctor told me that he

What goes into an LLM?

Implementation

1. **neural networks** as basic machine-learned computational elements
2. **embeddings**, vectors that represent meanings inside the network

Parameters

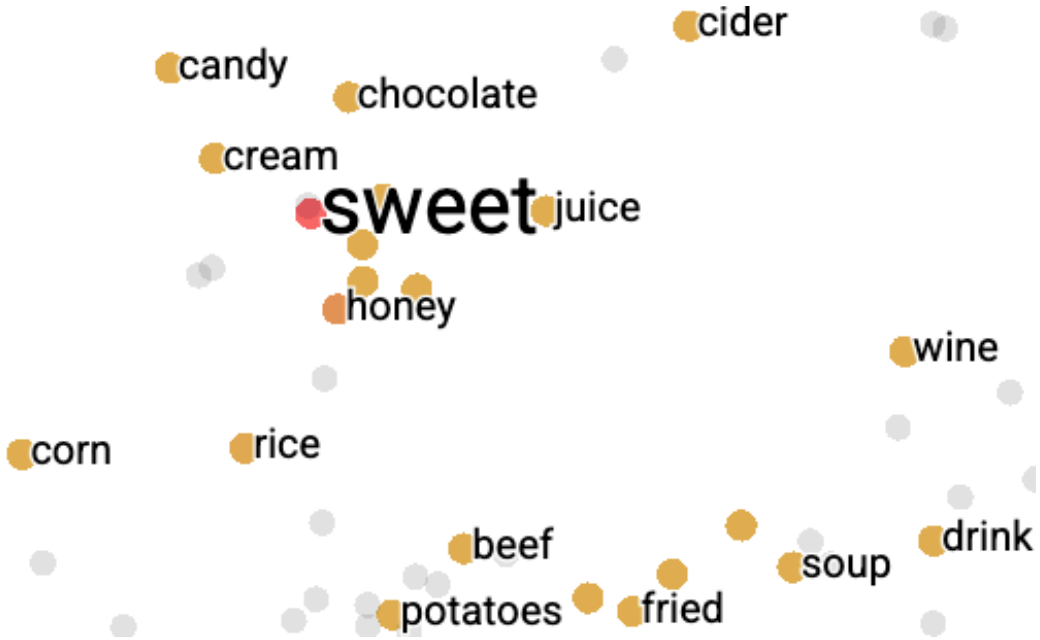
The weights and biases of the neural network

Open-weight models: the developer releases all the weights so others can inspect and run them

Closed-weight or **proprietary** models: the developer hosts the model and only allows app/API access

Embeddings

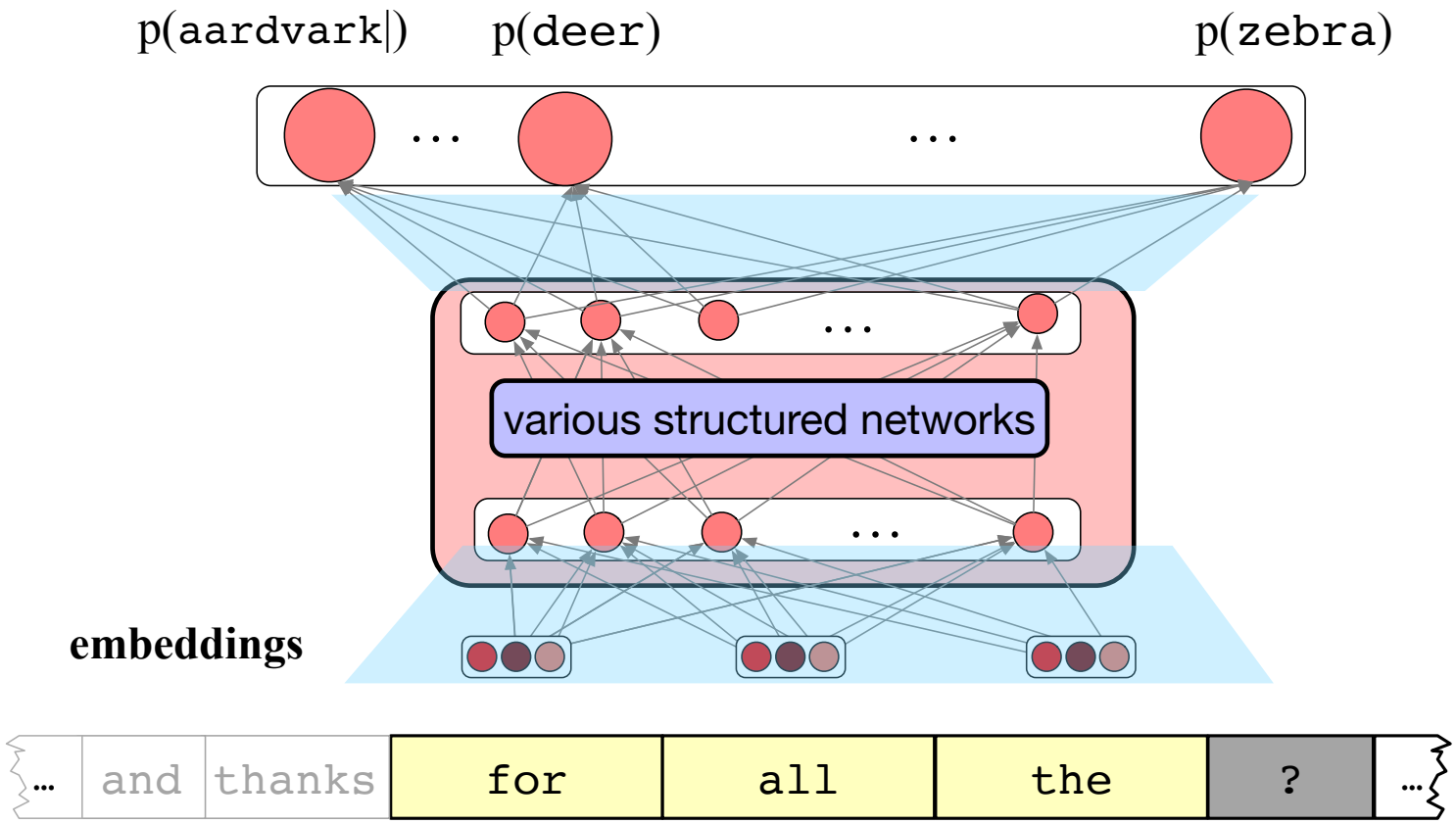
Invented by psychologist Charles Osgood in the 1950s



Embeddings represent meaning of a token as a point in high-dimensionality space.

We'll see how these can be used to do computations on meaning like word similarity

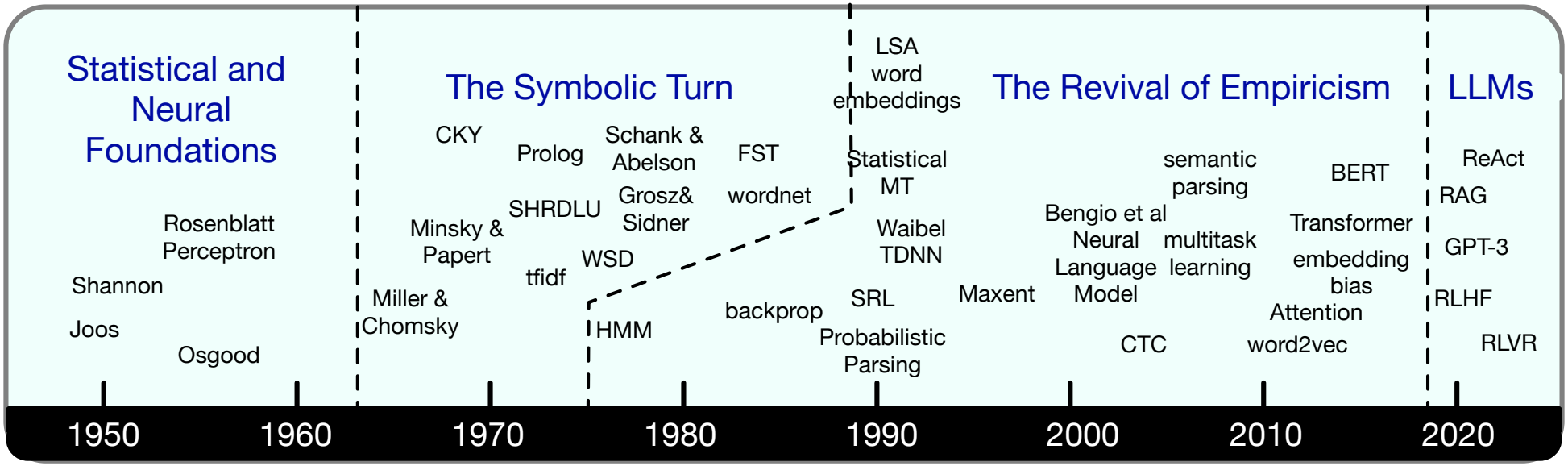
Putting together



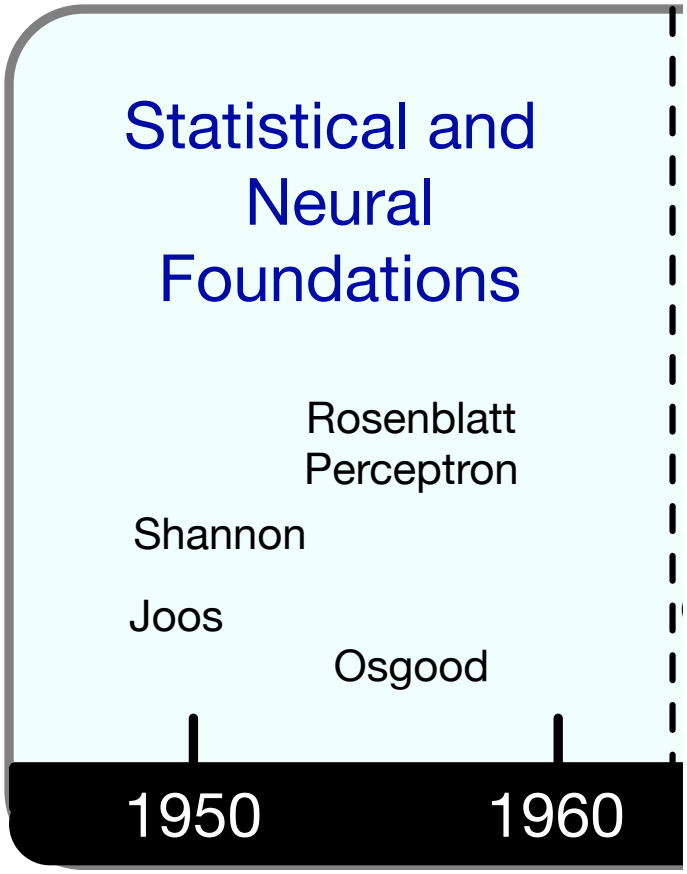
Large Language Models

A Brief History of Speech and Language Processing

Four stages in the history of the field

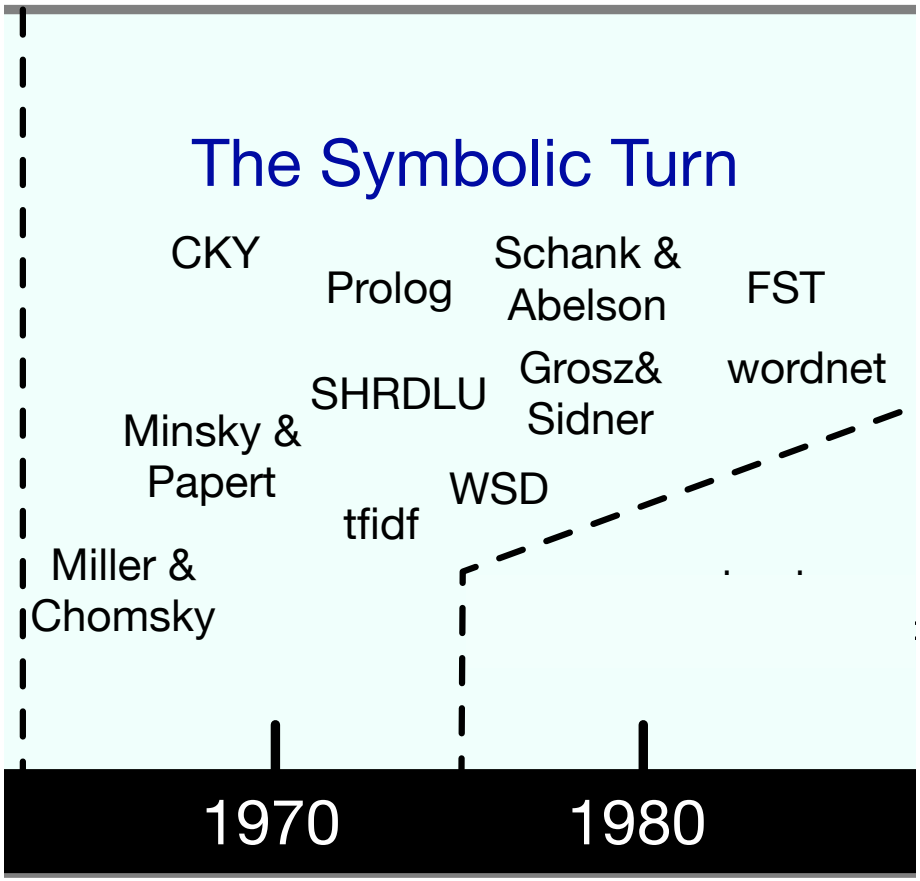


Statistical and Neural Foundations 1950s/early 1960s



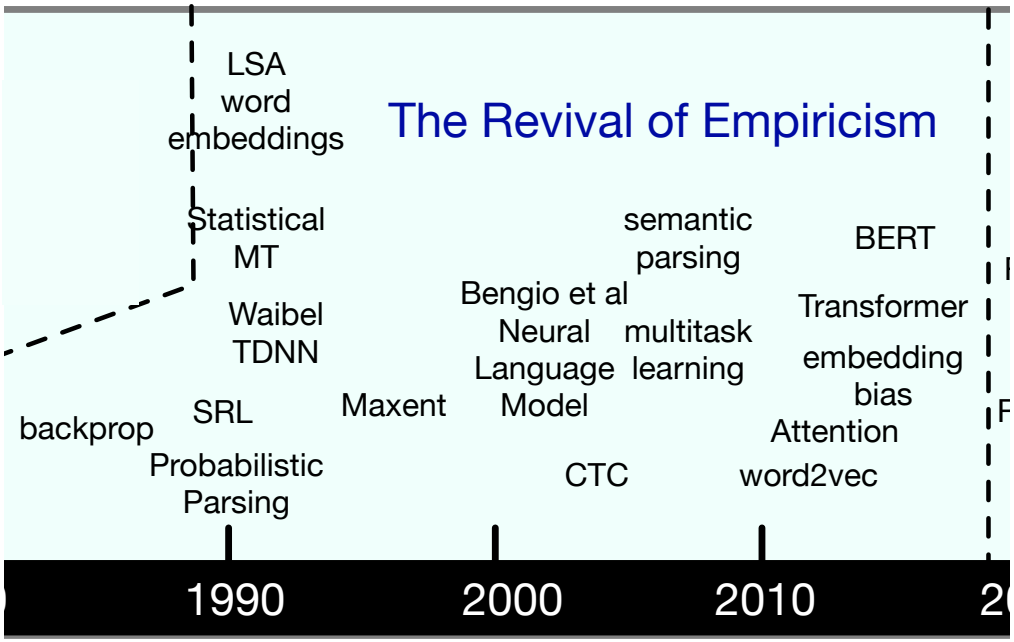
- Roots in 1950s and early 1960s
- Shannon (1951) on word prediction
- Rosenblatt at Cornell and Widrow at Stanford doing early neural nets
- Embedding intuition:
  - Osgood in psychology
  - Harris and Joos in linguistics
  - Early CS work on information retrieval

The Symbolic Turn, 1960s-1988



- Linguistic turn toward formal grammars, Chomsky 1957 and Miller and Chomsky 1963 arguments against statistical modeling
- Minsky and Papert (1969) XOR arguments against neural nets
- Parsing algorithms central to field
- Semantics (Prolog, logic, wordnet and Schank and Abelson)
- Discourse (Grosz and Sidner)

The Revival of Empiricism, 1988-2019



1975-Jelinek and IBM Speech team. Invented 'language model'

1986 Backprop, return of neural nets and connectionism

1988 Statistical models: speech to NLP (like IBM Statistical MT)

1990 LSA Embeddings

2003 Bengio Neural LM

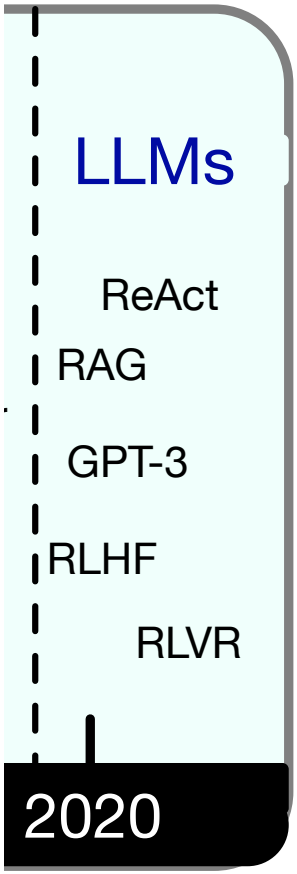
2008 neural multitask learning C&W

2015 Attention

2017 Transformer

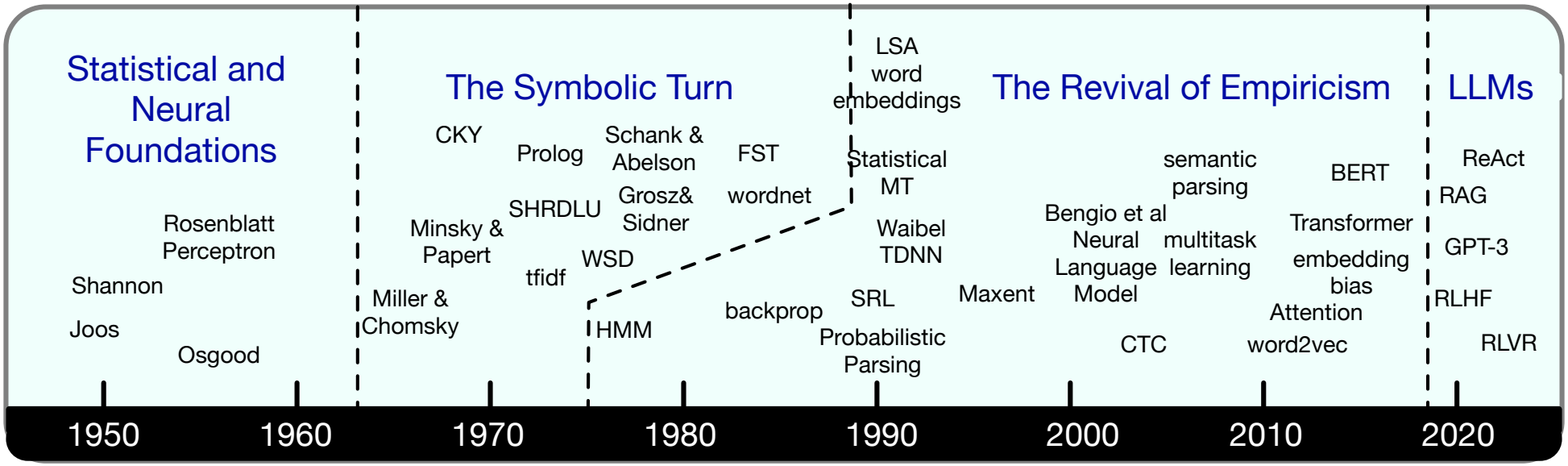
2019 BERT

Statistical and Neural Foundations LLM 2019-



GPT-2 (Radford et al 2019) prompting
RAG (Lewis et al 2020)
FLAN (Wei et al 2021) Instruction tuning
InstructGPT (Ouyang et al 2022) RLHF
ReAct (Yao et al 2022)
RLVR (Lambert et al 2024)

Four stages in the history of the field



Model architecture questions

- Token unit: Words, Phrases, Subwords, ...
- Token representations: Word counts, Linguistic structure, Embeddings, ...
- Model task: Predict next word, Classify document, Generate Translation, ...

- Coming up:
  - Surprising power of *word counts*: machine learning from text in a very simple statistical model

- Introduction: words and word frequencies

- First: How to extract words tokens from running text?

Raw text

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

Preprocessing: Text cleaning

```
DISCHARGE CONDITION:  The patient was able to oxygenate
on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.

DISCHARGE STATUS:  The patient will be discharged to
[**Hospital1 **]
for both pulmonary and physical rehabilitation.

DISCHARGE MEDICATIONS:
1. Levothyroxine 75 mcg p.o. q.d.
2. Citalopram 10 mg p.o. q.d.
3. Aspirin 81 mg p.o. q.d.
4. Fluticasone 110 mcg two puffs inhaled b.i.d.
5. Salmeterol Diskus one inhalation b.i.d.
6. Acetaminophen 325-650 mg p.o. q.4-6h. prn.
```

```
The patient was able to oxygenate on
room air at 93% at the time of discharge.  She was profoundly
weak, but was no longer tachycardic and had a normal blood
pressure.  Her respirations were much improved albeit with
transmitted upper airway sounds.
```

- Remove unwanted structure to just the sentences/ paragraphs you want to analyze
- Get text out of weird formats like HTML, PDFs, or idiosyncratic formatting (e.g. in these EHRs)

This step is usually specific to your dataset

Preprocessing: Tokenization

```
The patient was able to oxygenate on
room air at 93% at the time of discharge.  She was
profoundly
weak, but was no longer tachycardic and had a normal
blood
pressure.  Her respirations were much improved albeit
with
transmitted upper airway sounds.
```

```
['The', 'patient', 'was', 'able', 'to', 'oxygenate',
'on', 'room', 'air', 'at', '93', '%', 'at', 'the',
'time', 'of', 'discharge', '.', 'She', 'was',
'profoundly', 'weak', ',', 'but', 'was', 'no',
'longer', 'tachycardic', 'and', 'had', 'a',
'normal', 'blood', 'pressure', '.', 'Her',
'respirations', 'were', 'much', 'improved',
'albeit', 'with', 'transmitted', 'upper', 'airway',
'sounds', '.']
```

- Words are (usually) the basic units of analysis in NLP.
- In English, words are delineated as tokens via space and punctuation conventions, recognizable via moderately simple rules
- Tokenization: from text string to sequence of word strings
- Sentence splitting: harder but sometimes done too

There are good off-the-shelf word tokenizers (NLTK, SpaCy, Stanza, Huggingface...)

Preprocessing: Normalization

- Often:
  - Lowercase words (“She” -> “she”)
- Sometimes:
  - Remove numbers (“93” -> “NUMBER_NN”)
  - Correct misspellings / alternate spellings (“color” -> “colour”)
- Problem specific:
  - Resolve synonyms / aliases (if you know them already)
  - Remove “stopwords”
    - Punctuation and grammatical function words (“if”, “the”, “by”), and
    - Very common words in your domain that don’t add much meaning

How many words?

N = number of tokens

V = vocabulary = set of types

$|V|$ is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{½})$

	Tokens = N	Types = |V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Word frequencies

Word	Frequency (f)
the	1629
and	844
to	721
a	627
she	537
it	526
of	508
said	462
i	400
alice	385

Alice's Adventures in Wonderland, by Lewis Carroll

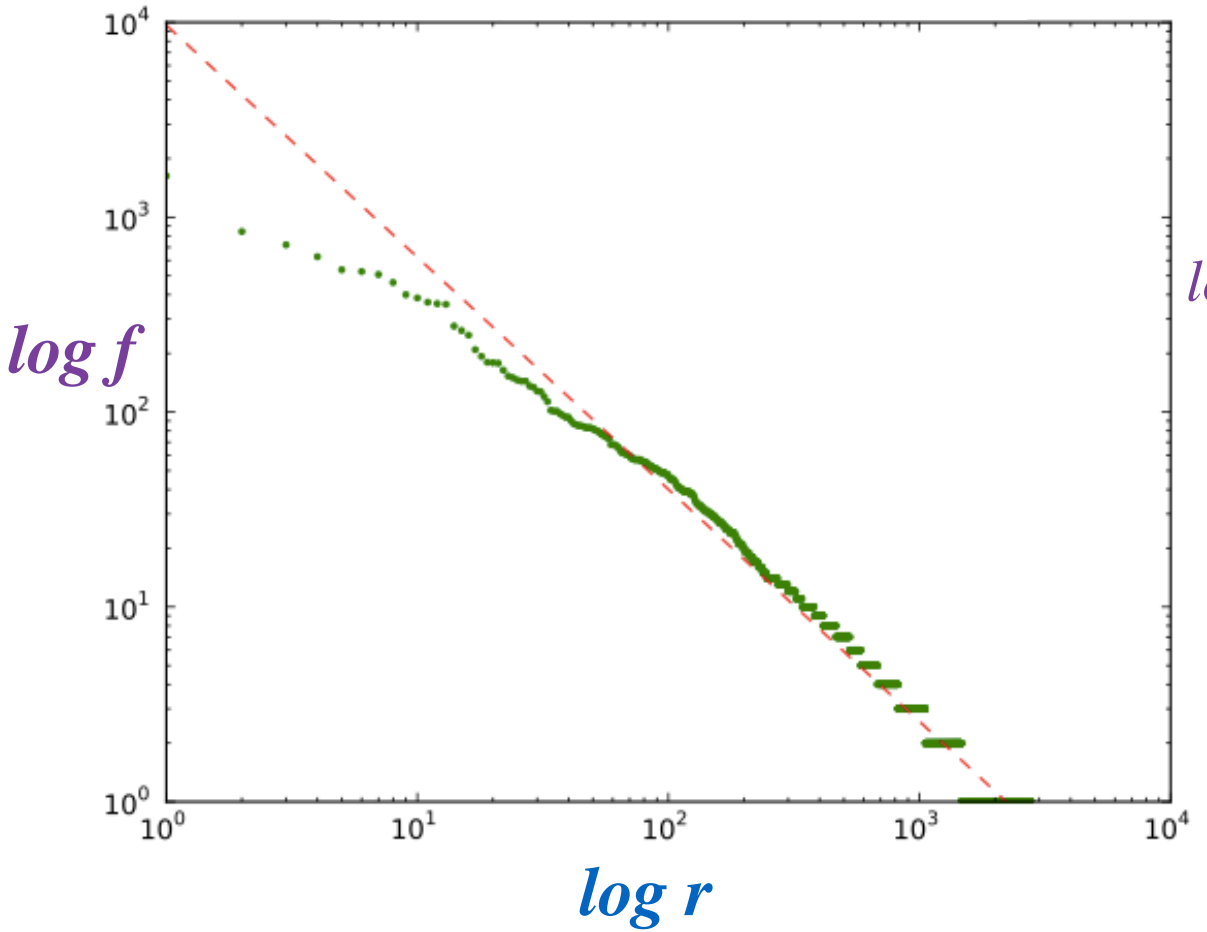
Zipf's Law

- When word types are ranked by frequency, then frequency (f) * rank (r) is roughly equal to some constant (k)

$$f \times r = k$$

Rank (r)	Word	Frequency (f)	$r \cdot f$
1	the	1629	1629
2	and	844	1688
3	to	721	2163
4	a	627	2508
5	she	537	2685
6	it	526	3156
7	of	508	3556
8	said	462	3696
9	i	400	3600
10	alice	385	3850
20	all	179	3580
30	little	128	3840
40	about	94	3760
50	again	82	4100
60	queen	68	4080
70	don't	60	4200
80	quite	55	4400
90	just	51	4590
100	voice	47	4700
200	hand	20	4000
300	turning	12	3600
400	hall	9	3600
500	kind	7	3500

Plot: log frequencies



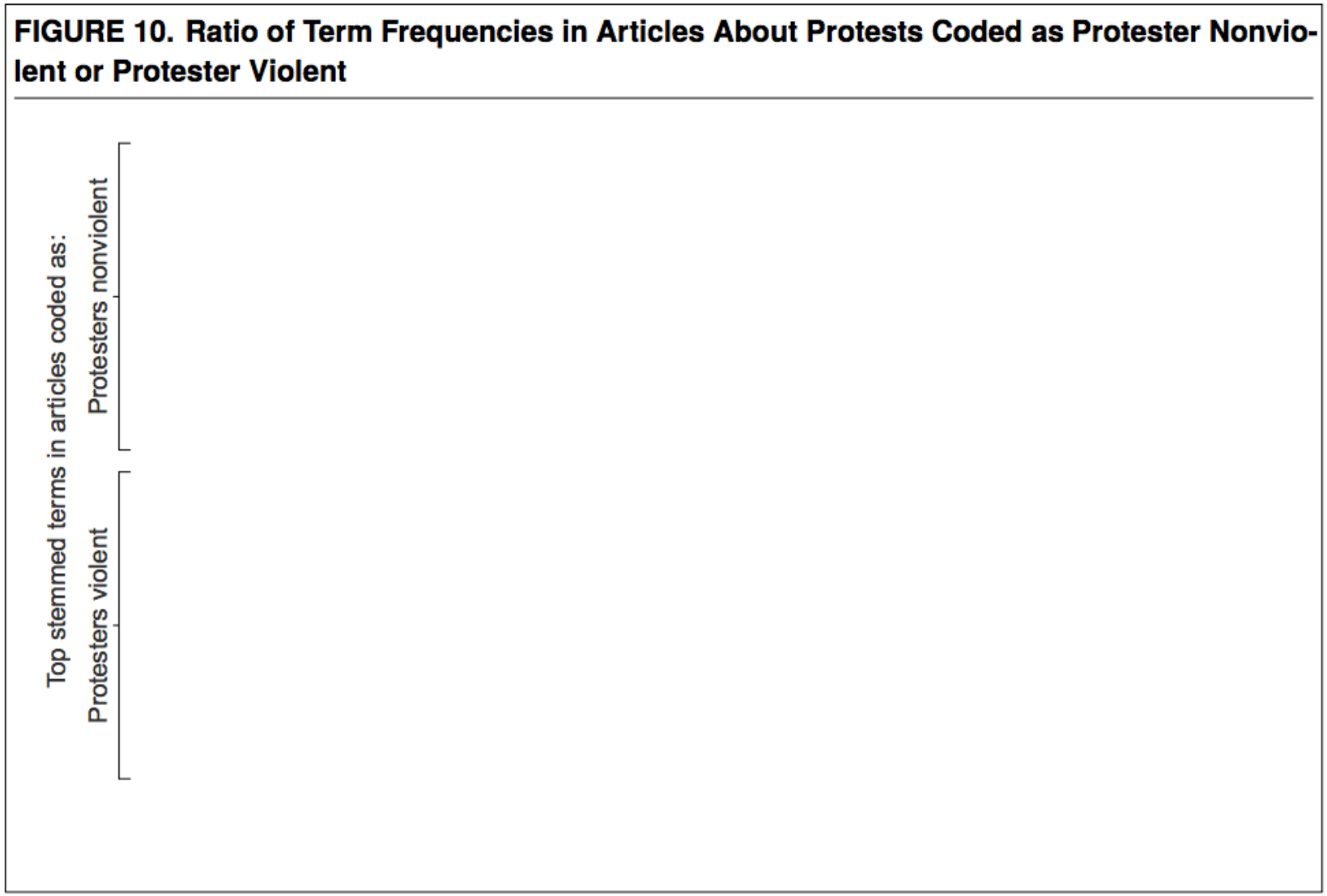
recall:

$f * r = k$

$\log f + \log r = \log k$

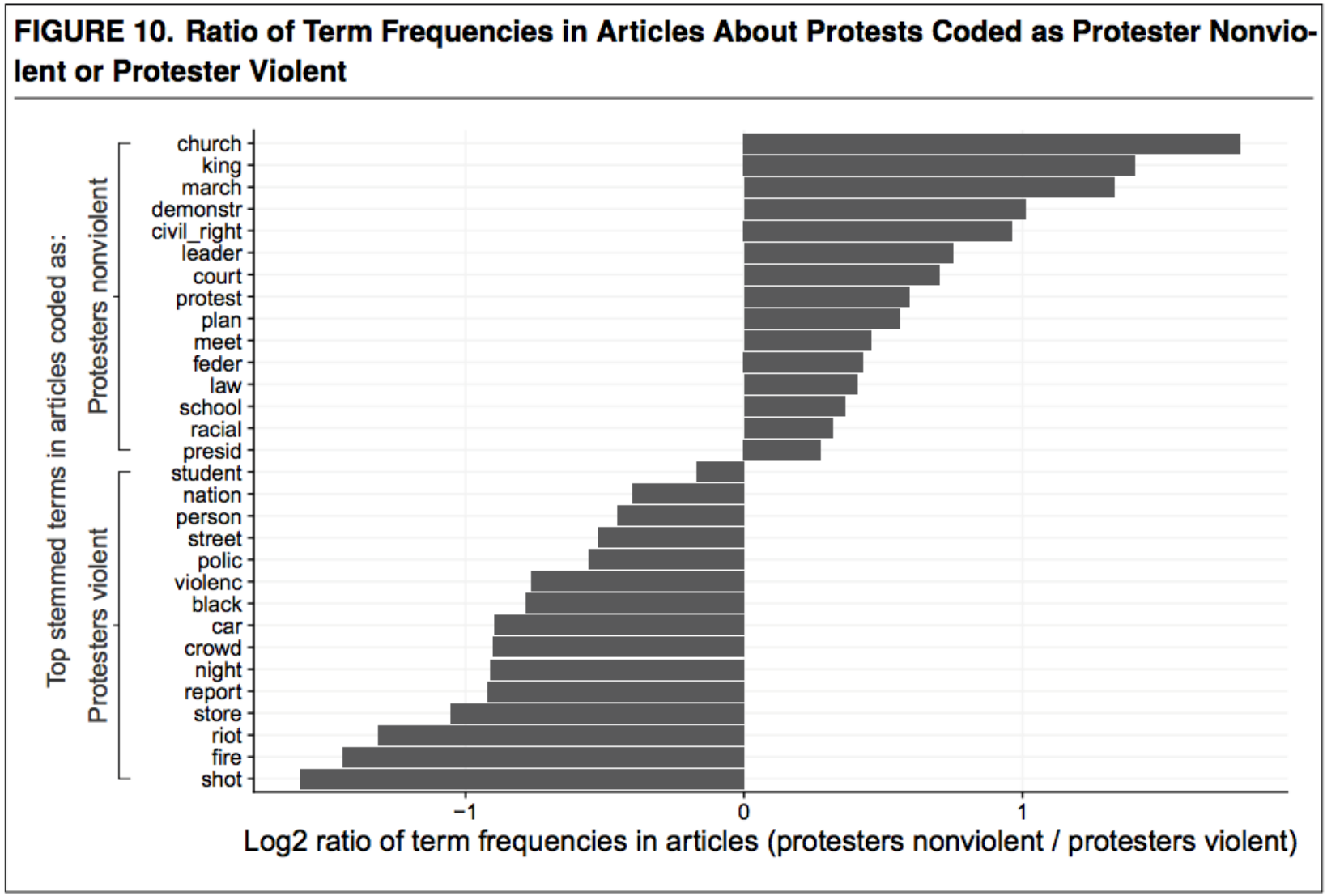
- stopped here 9/14

Contrast word counts from two subcorpora
Corpus: news articles from late 1960s



Wasow 2020

Contrast word counts from two subcorpora
Corpus: news articles from late 1960s



Wasow 2020