LLMs and the future of NLP

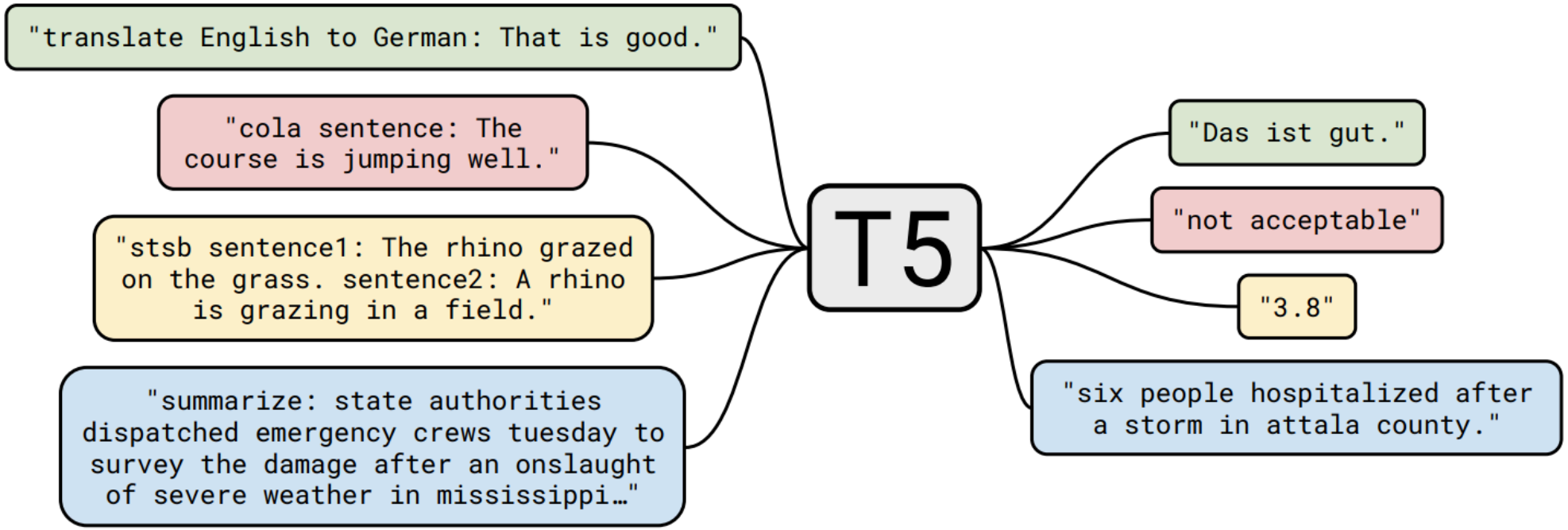
NLP Week 11

Thanks to Dan Jurafsky for most of the slides this week!

Plan for today

1. T5: everything is text-to-text
2. Large language models are few-shot learners
3. Sampling from language models
4. Prompting and in-context learning
5. Scaling up
6. Harms of large language models
7. *Group exercises*

T5: seq-to-seq everywhere



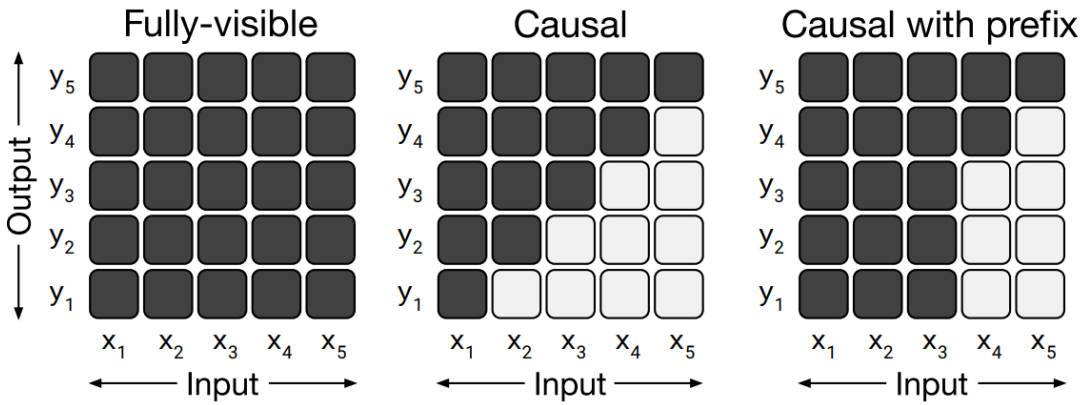
There are a few famous figures in our field, this is one of them :)

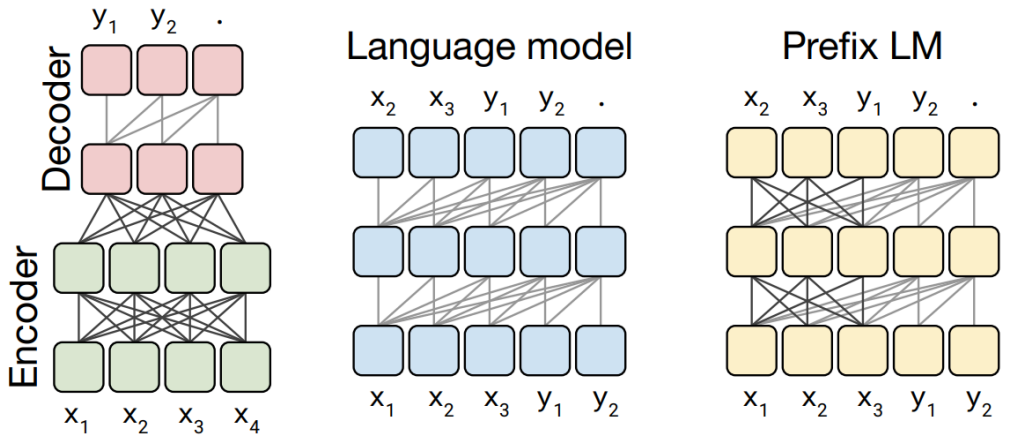
T5: seq-to-seq everywhere

- General **text-to-text** format
- Machine translation
  - Translate this text to French: Montréal is great. Montréal est génial
- Summarization
  - Summarize this news article: [Some article] [Summary of the article]
- What other tasks can you think of?
- Can you think of a NLP task that **can't** be formatted as text-to-text?

What's the best possible model for text-to-text leaning?

- **Model structure**
  - Which attention should we use?
    - Bi-directional
    - Causal
    - A mix of the two
- **Model architecture**
  - Encoder-decoder
  - Decoder-only
  - A mix of the two





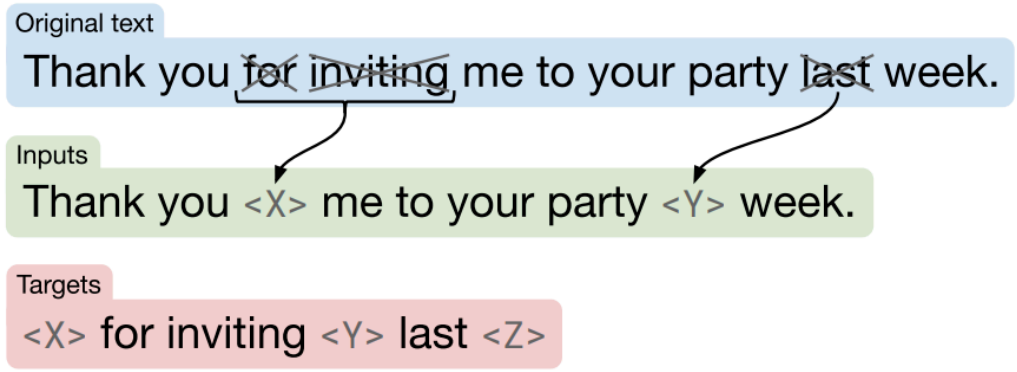
What's the best possible model for text-to-text leaning?

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

- Some combinations are better than others
- Denoising seems to be almost always better than LMing

What's the best possible model for text-to-text leaning?

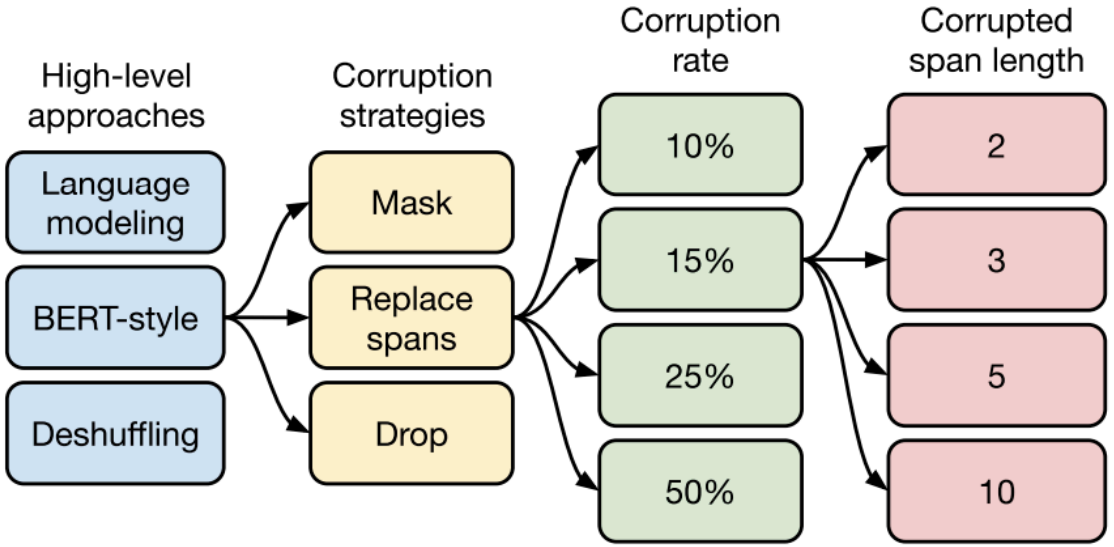
- Training objective
- What's the best way to encode useful information about and decode from a sequence?



Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you **<M>** **<M>** me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you **<M>** **<M>** me to your party **<M>** week .	*(original text)*
I.i.d. noise, replace spans	Thank you **<X>** me to your party **<Y>** week .	**<X>** for inviting **<Y>** last **<Z>**
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you **<X>** to **<Y>** week .	**<X>** for inviting me **<Y>** your party last **<Z>**

What's the best possible model for text-to-text leaning?

- Simply try all possible combinations



What's the best possible model for text-to-text leaning?

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

- Each of these matters to some extent

What’s the best possible model for text-to-text leaning?

- One important ingredient is missing.
- Can you guess which one?

What's the best possible model for text-to-text leaning?

- One important ingredient is missing
- Can you guess which one?

Data

What's the best possible model for text-to-text leaning?

- Data 
  - C4: Colossal Clean Crawled Corpus

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

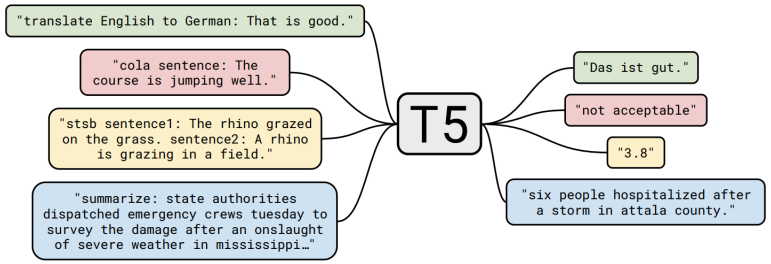
Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full data set	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

What's the best possible model for text-to-text leaning?

- **Fine-tuning**
- Recall from last week:
  - You can choose which weights to update
  - Updating all weights seems to works better for T5

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

Overall performance

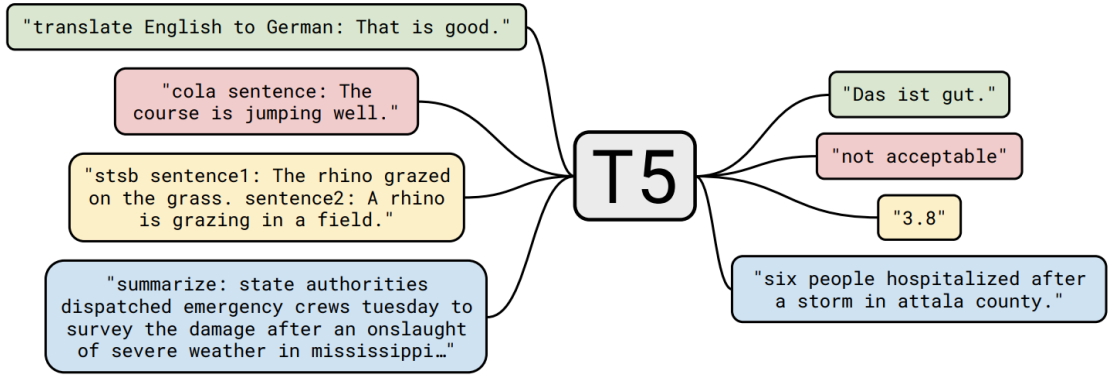


Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

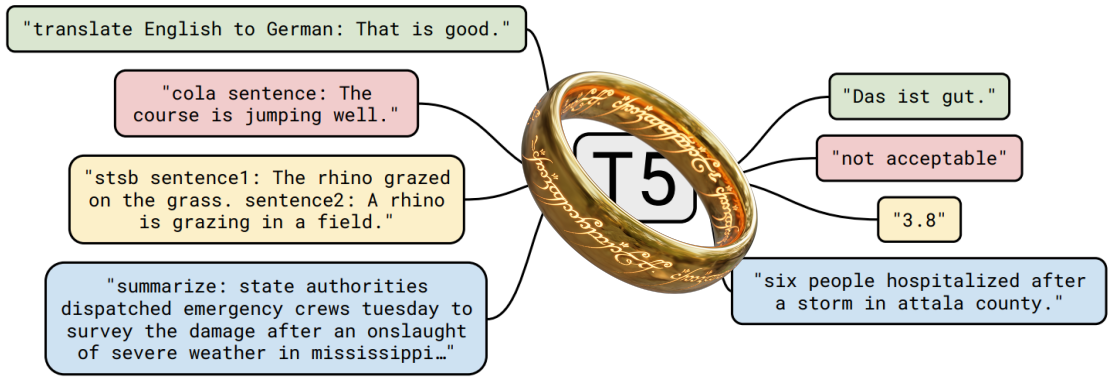
- Really good across the board. New SOTA on several tasks.
- More results in the paper

T5: seq-to-seq everywhere



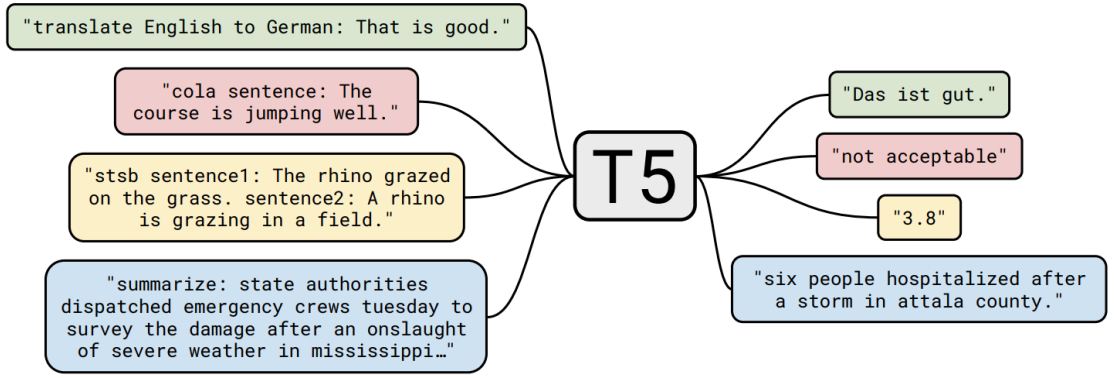
- Exhaustive search over the space of possible seq-to-seq models
- Very strong enc-dec model for NLP downstream tasks

T5: seq-to-seq everywhere



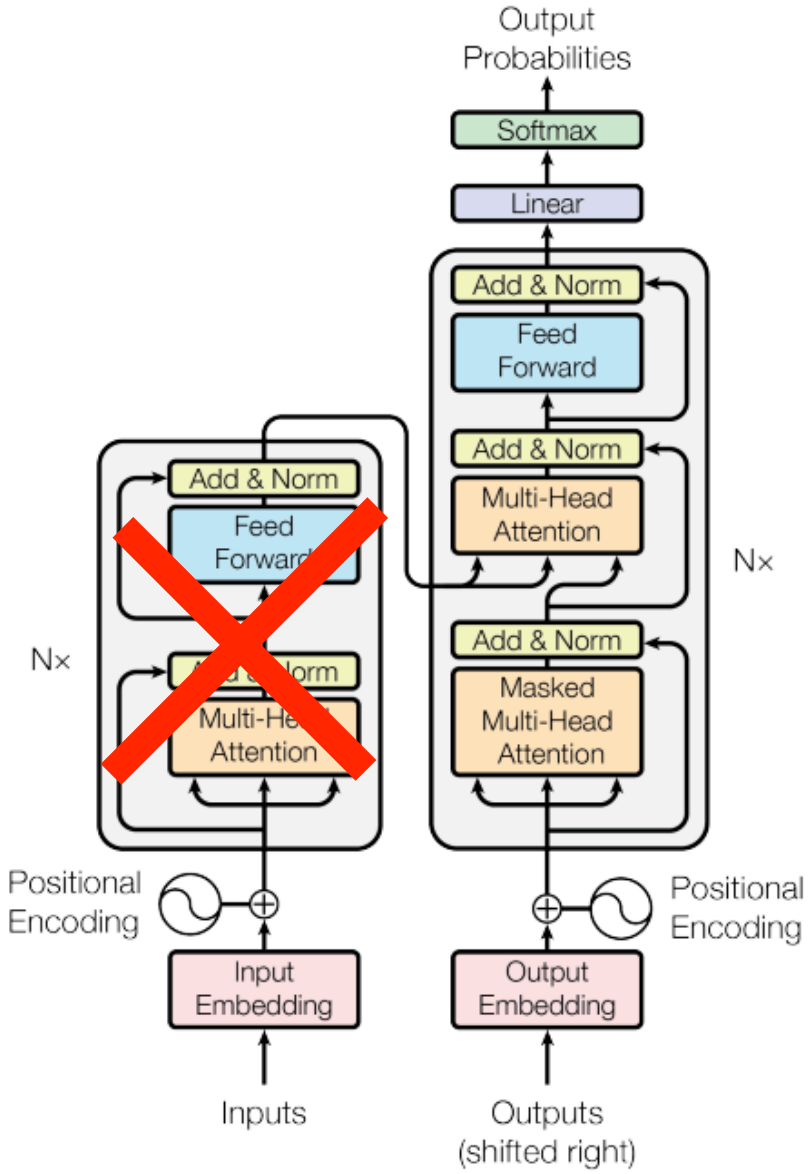
- Paradigm-shift: **One model to rule them all?**
- Preview for next week:
  - ~> This can be simplified and pushed even further …

Can we simplify this even more?

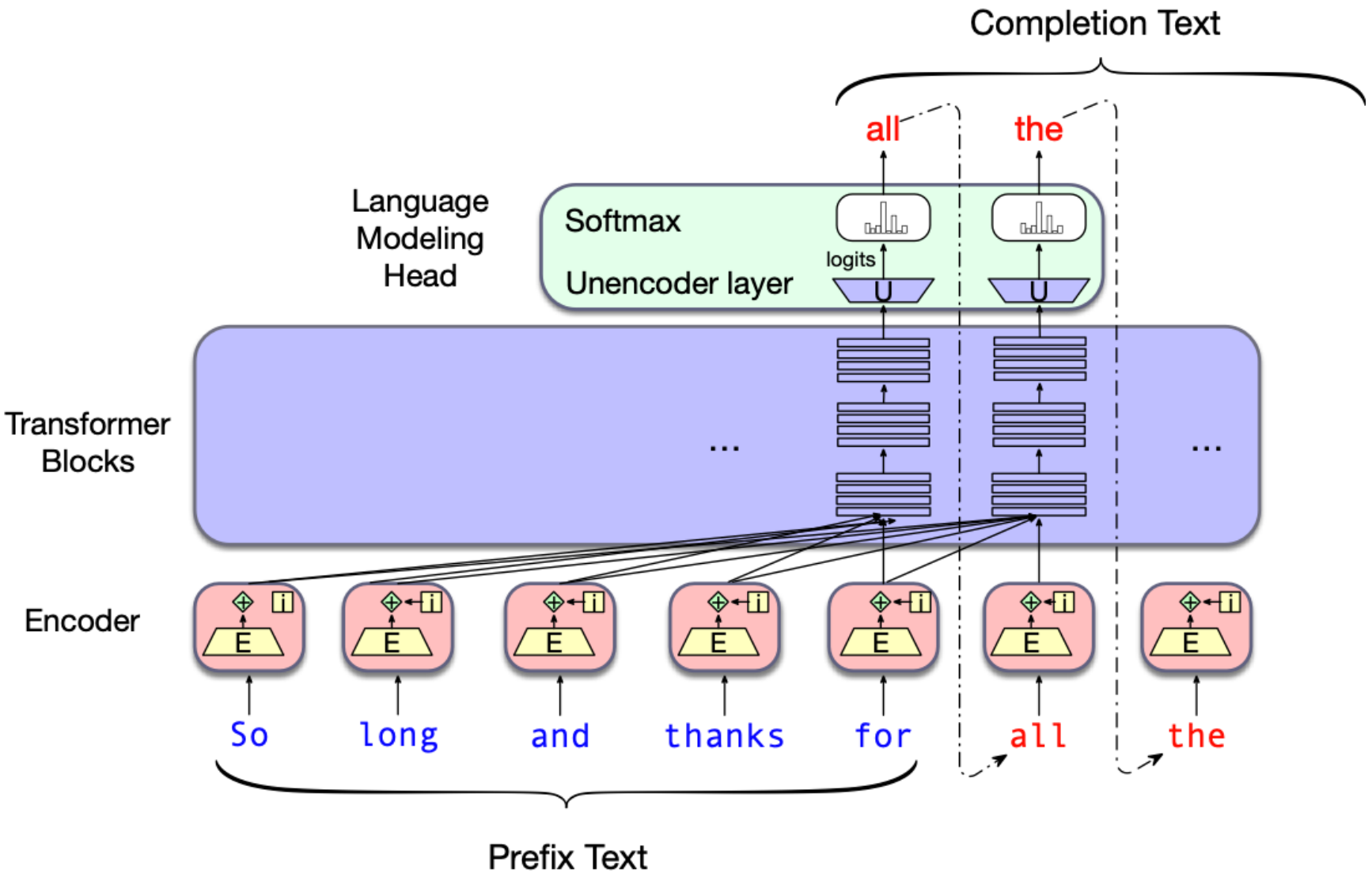


Big idea

- Let's keep only the **decoder**
- Many tasks are text-to-text
- Learn them by predicting the next word



Conditional Generation: Generating text conditioned on previous text!



Many practical NLP tasks can be cast as word prediction!

Sentiment analysis: “I like Jackie Chan”

1. We give the language model this string:
   `The sentiment of the sentence "I like Jackie Chan" is:`
2. And see what word it thinks comes next:

$$P(\text{positive}|\texttt{The sentiment of the sentence ‘‘I like Jackie Chan" is:})$$
$$P(\text{negative}|\texttt{The sentiment of the sentence ‘‘I like Jackie Chan" is:})$$

Framing lots of tasks as conditional generation

QA: “Who wrote The Origin of Species”

1. We give the language model this string:

```
Q: Who wrote the book ‘‘The Origin of Species"?  A:
```

2. And see what word it thinks comes next:

$$P(w|\texttt{Q: Who wrote the book ‘‘The Origin of Species"? A:})$$

3. And iterate:

$$P(w|\texttt{Q: Who wrote the book ‘‘The Origin of Species"? A: Charles})$$

Summarization

Original

The only thing crazier than a guy in snowbound Massachusetts boxing up the powdery white stuff and offering it for sale online? People are actually buying it. For $89, self-styled entrepreneur Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says.

But not if you live in New England or surrounding states. “We will not ship snow to any states in the northeast!” says Waring’s website, ShipSnowYo.com. “We’re in the business of expunging snow!”

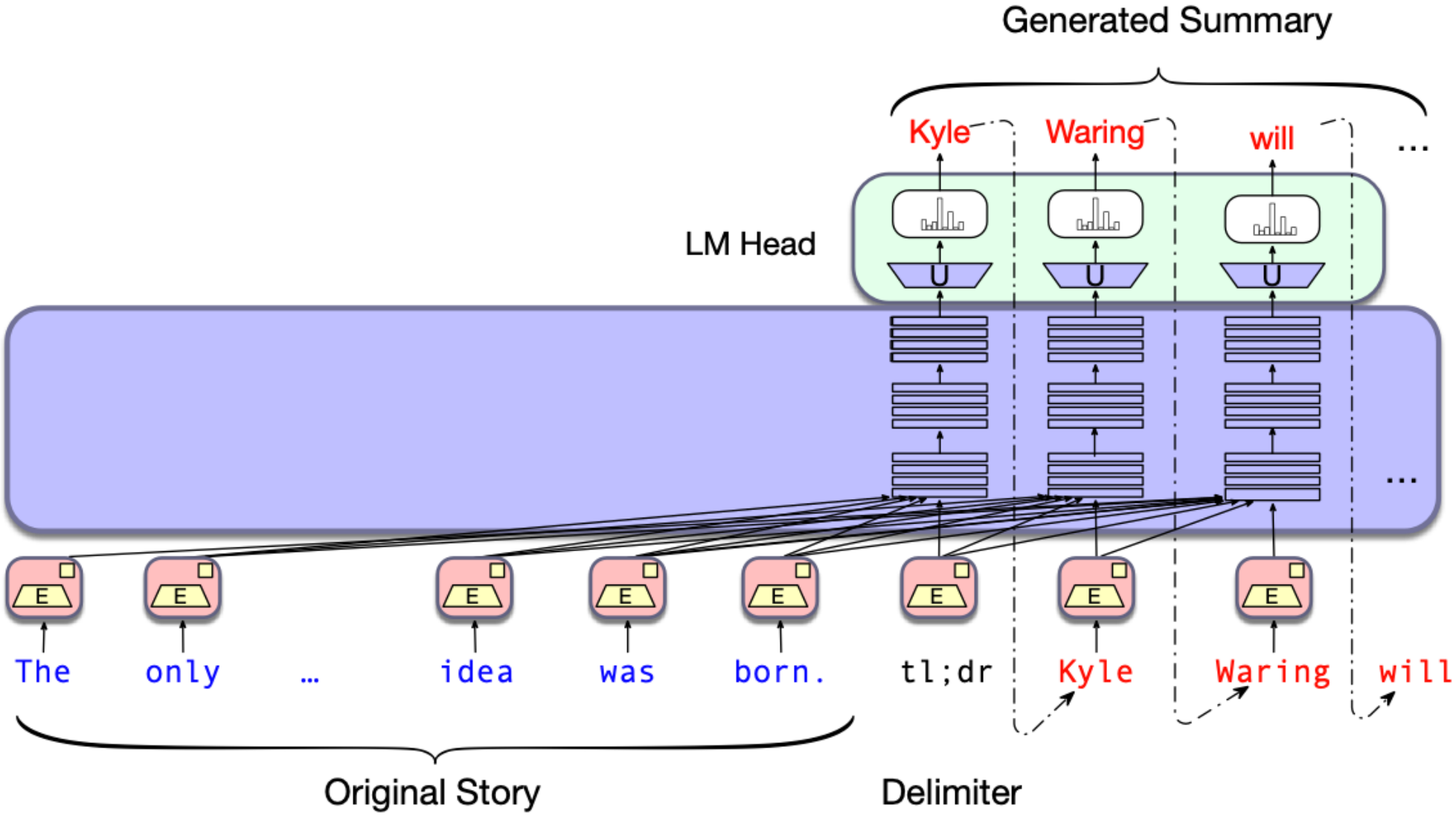
His website and social media accounts claim to have filled more than 133 orders for snow – more than 30 on Tuesday alone, his busiest day yet. With more than 45 total inches, Boston has set a record this winter for the snowiest month in its history. Most residents see the huge piles of snow choking their yards and sidewalks as a nuisance, but Waring saw an opportunity.

According to Boston.com, it all started a few weeks ago, when Waring and his wife were shoveling deep snow from their yard in Manchester-by-the-Sea, a coastal suburb north of Boston. He joked about shipping the stuff to friends and family in warmer states, and an idea was born. [...]

Summary

Kyle Waring will ship you 6 pounds of Boston-area snow in an insulated Styrofoam box – enough for 10 to 15 snowballs, he says. But not if you live in New England or surrounding states.

LLMs for summarization (using tl;dr)



Reminder: Self-supervised training algorithm

We just train them to predict the next word!

1. Take a corpus of text
2. At each time step *t*
   i. ask the model to predict the next word
   ii. train the model using gradient descent to minimize the error in this prediction

"**Self-supervised**" because it just uses the next word as the label!

Reminder: Cross-entropy loss for language modeling

CE loss: difference between the correct probability distribution and the predicted distribution

$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

The correct distribution $\mathbf{y}_t$ knows the next word, so is 1 for the actual next word and 0 for the others.

So in this sum, all terms get multiplied by zero except one: the logp the model assigns to the correct next word, so:

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

Reminder: Perplexity

Just as for n-gram grammars, we use perplexity to measure how well the LM predicts unseen text

The perplexity of a model θ on an unseen test set is the **inverse probability that θ assigns to the test set, normalized by the test set length.**

For a test set of n tokens $w_{1:n}$ the perplexity is :

$$\begin{aligned}\text{Perplexity}_\theta(w_{1:n}) &= P_\theta(w_{1:n})^{-\frac{1}{n}} \\ &= \sqrt[n]{\frac{1}{P_\theta(w_{1:n})}}\end{aligned}$$

Reminder: Perplexity

Just as for n-gram grammars, we use perplexity to measure how well the LM predicts unseen text

The perplexity of a model θ on an unseen test set is the **inverse probability that θ assigns to the test set, normalized by the test set length.**

For a test set of n tokens $w_{1:n}$ the perplexity is :

$$\begin{aligned}\text{Perplexity}_\theta(w_{1:n}) &= P_\theta(w_{1:n})^{-\frac{1}{n}} \\ &= \sqrt[n]{\frac{1}{P_\theta(w_{1:n})}}\end{aligned}$$

Why perplexity instead of raw probability of the test set?

- Probability depends on size of test set
  - Probability gets smaller the longer the text
  - Better: a metric that is **per-word**, normalized by length
- **Perplexity** is the inverse probability of the test set, normalized by the number of words
  - (The inverse comes from the original definition of perplexity from cross-entropy rate in information theory)
  - Probability range is [0,1], perplexity range is [1,∞]

Reminder: Perplexity

Alternative view on perplexity

- Intuitively, ppl can be seen as a measure of among how many tokens a models has to choose when predicting the next tokens
- ppl $= 2^{H(x)}$, where
- $H(x) = -\frac{1}{N} \log P(x_1, \ldots, x_n)$
- Hence, in practice we simply exponentiate the loss of our model to get perplexity
- Hyper-parameter: how do you choose your context?

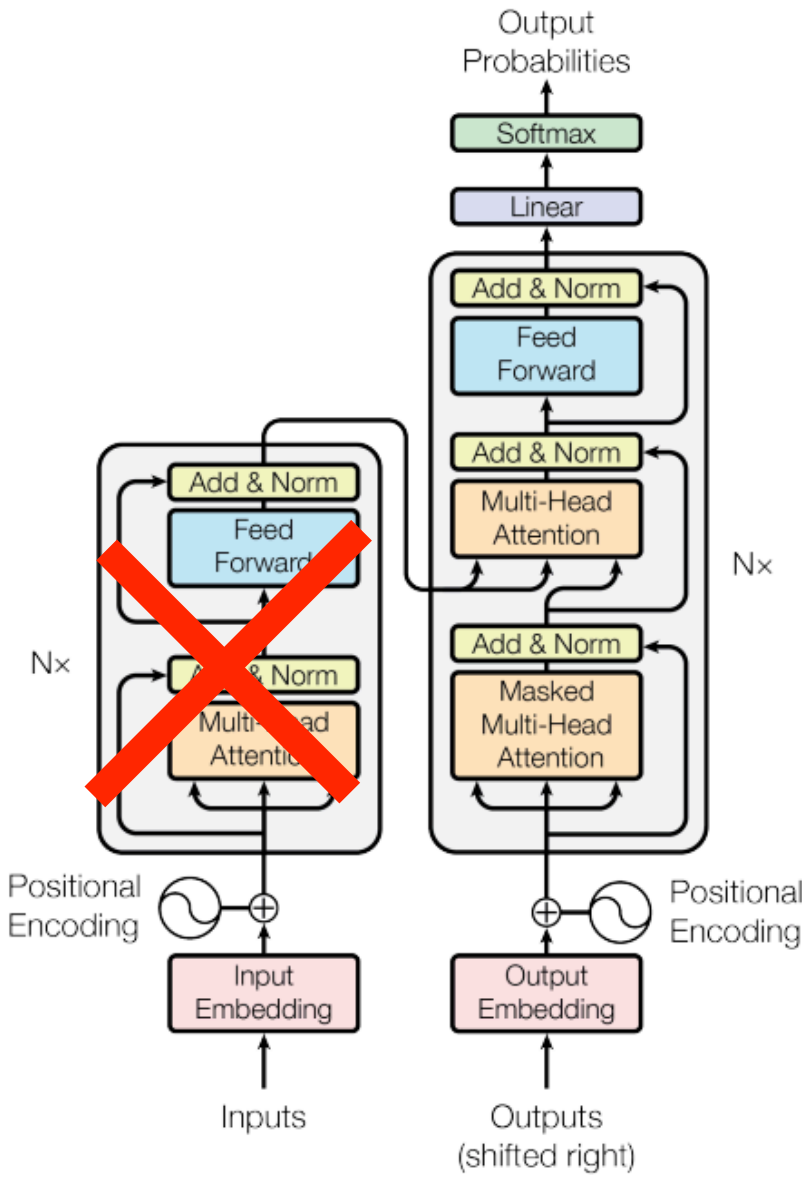
Cross-entropy is an upper-bound of the entropy:
$H(p) \leq H(p, m)$

Perplexity

- The higher the probability of the word sequence, the lower the perplexity.
- Thus the lower the perplexity of a model on the data, the better the model.
- **Minimizing perplexity is the same as maximizing probability**
- Also: perplexity is sensitive to length/tokenization so best used when comparing LMs that use the same tokenizer.

Let's go back to the big idea

- Let's keep only the **decoder**
- Many tasks are text-to-text
- Learn them by predicting the next word



Why would this work?

Pre-training

The technique that underlies (most of) the amazing performance of language models

First **pre-train** a transformer model on enormous amounts of text

Then **apply** it to new tasks.

LLMs are mainly trained on the web

Common crawl, snapshots of the entire web produced by the non- profit Common Crawl with billions of pages

Colossal Clean Crawled Corpus (C4; Raffel et al. 2020), 156 billion tokens of English, filtered

What's in it? Mostly patent text documents, Wikipedia, and news sites

The Pile: a pre-training corpus

academics web books

dialog

What does a model learn from pretraining?

- There are canines everywhere! One dog in the front room, and two dogs
- It wasn't just big it was enormous
- The author of "A Room of One's Own" is Virginia Woolf
- The doctor told me that he
- The square root of 4 is 2

Filtering for quality and safety

Quality is subjective

- Many LLMs attempt to match Wikipedia, books, particular websites
- Need to remove boilerplate, adult content
- Deduplication at many levels (URLs, documents, even lines)

Safety also subjective

- Toxicity detection is important, although that has mixed results
- Can mistakenly flag data written in dialects like African American English

But there are problems with scraping from the web

Copyright: much of the text in these datasets is copyrighted

- Not clear if fair use doctrine in US allows for this use
- This remains an open legal question

Data consent

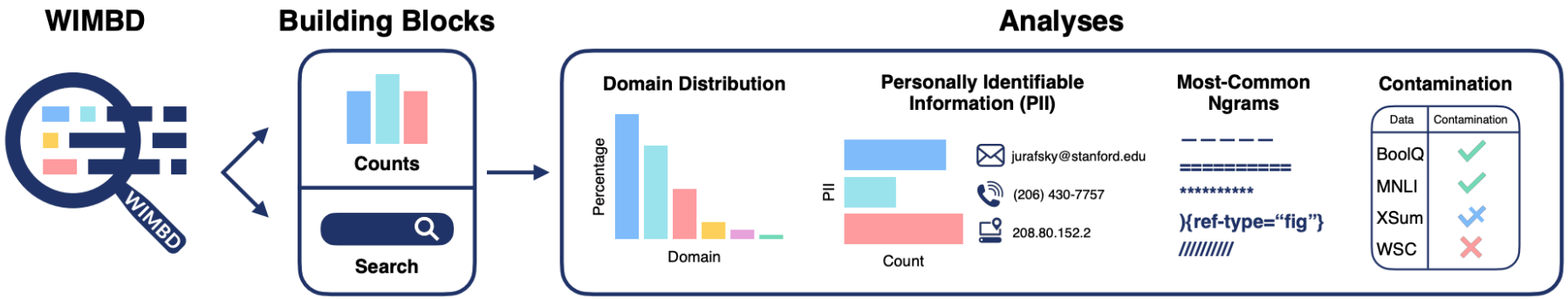
- Website owners can indicate they don't want their site crawled

Privacy:

- Websites can contain private IP addresses and phone numbers

What's in my big data?

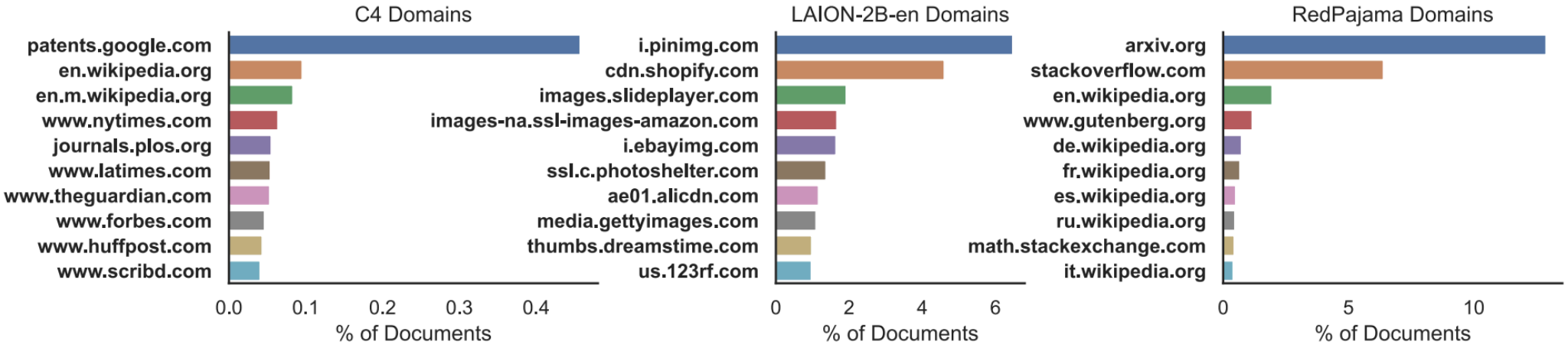
- A tool to count and search in pre-training corpora
- Allows you to investigate the pre-training data of a model
- Alternatives: infinigram (Liu et al. 2024; https://infini-gram.io/)



Elazar et al. (2024) - What's in my big data?

What's in my big data?

Elazar et al. (2024) - What's in my big data?



OpenWebText

n-gram	Count
- - - - - - - - - -	3.4M
. . . . .	1.05M
= = = = = = = = = =	830K
* * * * * * * * * *	595K
# # # # # # # # # #	302K
amp ; amp ; amp ; amp ; amp ;	278K
; amp ; amp ; amp ; amp ; amp	265K
— — — — — — — — — —	249K
… … … … … … … … … … …	88.1K
~ ~ ~ ~ ~ ~ ~ ~ ~ ~	83.3K

C4

n-gram	Count
? ? ? ? ? ? ? ? ? ?	9M
. . . . .	7.27M
- - - - - - - - - -	4.41M
* * * * * * * * * *	3.87M
! ! ! ! ! ! ! ! ! !	1.91M
. You can follow any responses to this entry through	784K
[illegible]	753K
You can follow any responses to this entry through the	752K
can follow any responses to this entry through the RSS	752K
follow any responses to this entry through the RSS 2.0	748K

mC4-en

n-gram	Count
. . . . .	1.76B
- - - - - - - - - -	823M
[illegible]	349M
* * * * * * * * * *	314M
\ / s \ / files \ / 1 \	183M
/ s \ / files \ / 1 \ /	183M
\ / \ / cdn.shopify.com \ / s \ /	182M
/ cdn.shopify.com \ / s \ / files \ /	182M
\ / cdn.shopify.com \ / s \ / files \	182M
/ \ / cdn.shopify.com \ / s \ / files	182M

OSCAR

n-gram	Count
	773M
\ \ \ \ \ \ \ \ \	395M
- - - - - - - - - -	175M
. . . . .	91.6M
* * * * * * * * * *	34.9M
= = = = = = = = = =	22.9M
(Opens in new window) Click to share on	15.7M
Log Out / Change) You are commenting using your	13.6M
(Log Out / Change) You are commenting using	13.6M
. (Log Out / Change) You are commenting	13.6M

The Pile

n-gram	Count
- - - - - - - - - -	3.64B
= = = = = = = = = =	602M
* * * * * * * * * *	188M
) { ref - type = " fig " }	59.1M
/ / / / / / / / / /	56.2M
. . . . .	54.9M
# # # # # # # # # #	38.3M
} - - - - - - - - -	30.1M
{ ref - type = " fig " })	28.9M
} = = = = = = = = =	21.8M

RedPajama

n-gram	Count
. . . . .	670M
- - - - - - - - - -	507M
\ \ \ \ \ \ \ \ \ \	213M
* * * * * * * * * *	195M
= = = = = = = = = =	145M
/ / / / / / / / / /	79.3M
. . / . . / . . / .	35.3M
. / . . / . . / . .	35.3M
/ . . / . . / . . /	35.2M
# # # # # # # # # #	33M

S2ORC

n-gram	Count
q q q q q q q q q q	30.2M
. . . . .	5.49M
+ + + + + + + + + +	3.03M
* * * * * * * * * *	1.93M
° ° ° ° ° ° ° ° ° °	1.73M
. . . . .	1.56M
- - - - - - - - - -	1.11M
[5] [6] [7] [	646K
[1] [2] [3] [	645K
[6] [7] [8] [	644K

peS2o

n-gram	Count
. . . . .	1.42M
[1] [2] [3] [	457K
] [2] [3] [4]	453K
1] [2] [3] [4	453K
[5] [6] [7] [	450K
[6] [7] [8] [	448K
] [6] [7] [8]	448K
5] [6] [7] [8	446K
] [7] [8] [9]	446K
6] [7] [8] [9	444K

LAION-2B-en

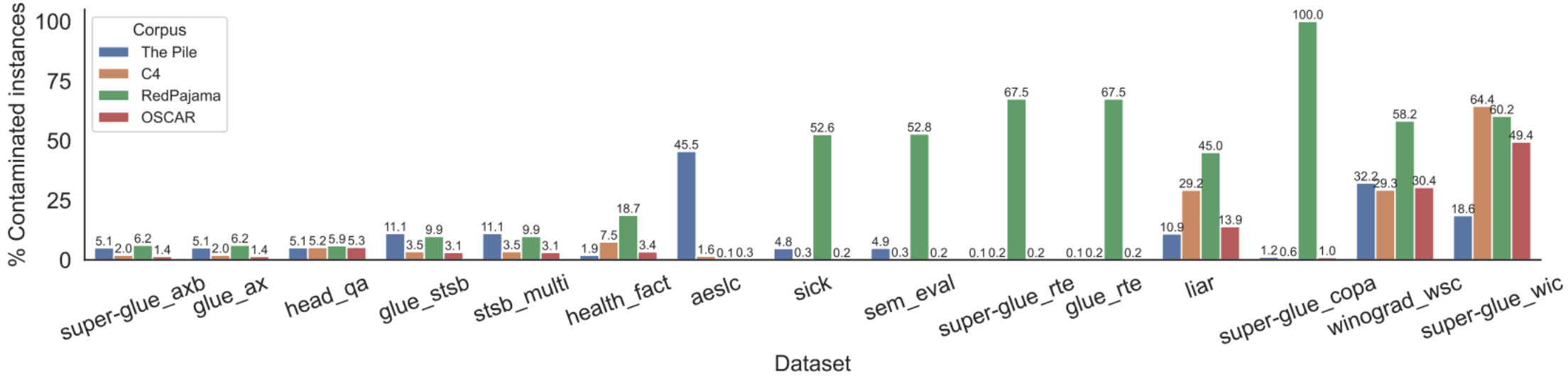
n-gram	Count
- - - - - - - - - -	1.65M
[illegible]	1.43M
. . . . .	1.15M
\ \ \ \ \ \ \ \ \ \	809K
< br / > < br / > < br	797K
/ > < br / > < br / >	796K
br / > < br / > < br /	796K
> < br / > < br / > <	576K
| Price : 1 Credit (USD $ 1)	437K
vector | Price : 1 Credit (USD $ 1	437K

The Stack

n-gram	Count
- - - - - - - - - -	4.29B
* * * * * * * * * *	3.87B
0 0 0 0 0 0 0 0 0 0	2.75B
= = = = = = = = = =	2.62B
, " resolved " : " https : / /	1.46B
" , " resolved " : " https : /	1.46B
" resolved " : " https : / / registry.npmjs.org	1.42B
resolved " : " https : / / registry.npmjs.org /	1.42B
, , , , , , , , , ,	1B
. tgz " , " integrity " : " sha512	938M

What's in my big data?

Elazar et al. (2024) - What's in my big data?



Once a model is trained, how can we generate text from it?

Decoding and Sampling

This task of choosing a word to generate based on the model's probabilities is called **decoding.**

The most common method for decoding in LLMs: **sampling.**

Sampling from a model's distribution over words:

- choose random words according to their probability assigned by the model.

After each token we'll sample words to generate according to their probability *conditioned on our previous choices*,

- A transformer language model will give the probability

Random sampling

$$
\begin{aligned}
&\mathrm{i} \leftarrow 1 \\
&w_i \sim \mathrm{p}(w) \\
&\textbf{while } w_i \text{ != EOS} \\
&\quad \mathrm{i} \leftarrow \mathrm{i} + 1 \\
&\quad w_i \sim \mathrm{p}(w_i \mid w_{<i})
\end{aligned}
$$

Random sampling doesn't work very well

Even though random sampling mostly generate sensible, high-probable words,

There are many odd, low- probability words in the tail of the distribution

Each one is low- probability but added up they constitute a large portion of the distribution

So they get picked enough to generate weird sentences

Factors in word sampling: quality and diversity

Emphasize **high-probability** words

\+ **quality**: more accurate, coherent, and factual,

- diversity: boring, repetitive.

Emphasize **middle-probability** words

\+ **diversity**: more creative, diverse,

\- **quality**: less factual, incoherent

Top-k sampling:

1. Choose # of words k
2. For each word in the vocabulary V , use the language model to compute the likelihood of this word given the context $p(w_t | w_{<t})$
3. Sort the words by likelihood, keep only the top k most probable words.
4. **Renormalize** the scores of the k words to be a legitimate probability distribution.
5. Randomly sample a word from within these remaining k most-probable words according to its probability.

Top-p sampling (= nucleus sampling)

Holtzman et al., 2020

Problem with top-*k:* *k* is fixed so may cover very different amounts of probability mass in different situations

Idea: Instead, keep the top p percent of the probability mass

Given a distribution $P(w_t | \mathbf{w}_{<t})$, the top-*p* vocabulary *V* (*p*) is the smallest set of words such that

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p.$$

Temperature sampling

Reshape the distribution instead of truncating it

Intuition from thermodynamics,

- a system at high temperature is flexible and can explore many possible states,
- a system at lower temperature is likely to explore a subset of lower energy (better) states.

In **low-temperature sampling**, $(\tau \leq 1)$ we smoothly

- increase the probability of the most probable words
- decrease the probability of the rare words.

Temperature sampling

Divide the logit by a temperature parameter τ before passing it through the softmax.

Instead of

$$\mathbf{y} = \text{softmax}(u)$$

We do

$$\mathbf{y} = \text{softmax}(u/\tau)$$

Temperature sampling

$0 \leq \tau \leq 1$

$$\mathbf{y} = \text{softmax}(u/\tau)$$

Why does this work?

- When τ is close to 1 the distribution doesn't change much.
- The lower τ is, the larger the scores being passed to the softmax
- Softmax pushes high values toward 1 and low values toward 0.
- Large inputs pushes high-probability words higher and low probability word lower, making the distribution more greedy.
- As τ approaches 0, the probability of most likely word approaches 1

Show this in VSCode

How can we guide the generations of a model?

Prompting and in-context learning (ICL)

- A prompt refers to the context provided to the language model
- Various components of a prompt, e.g., instruction, description, etc.

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

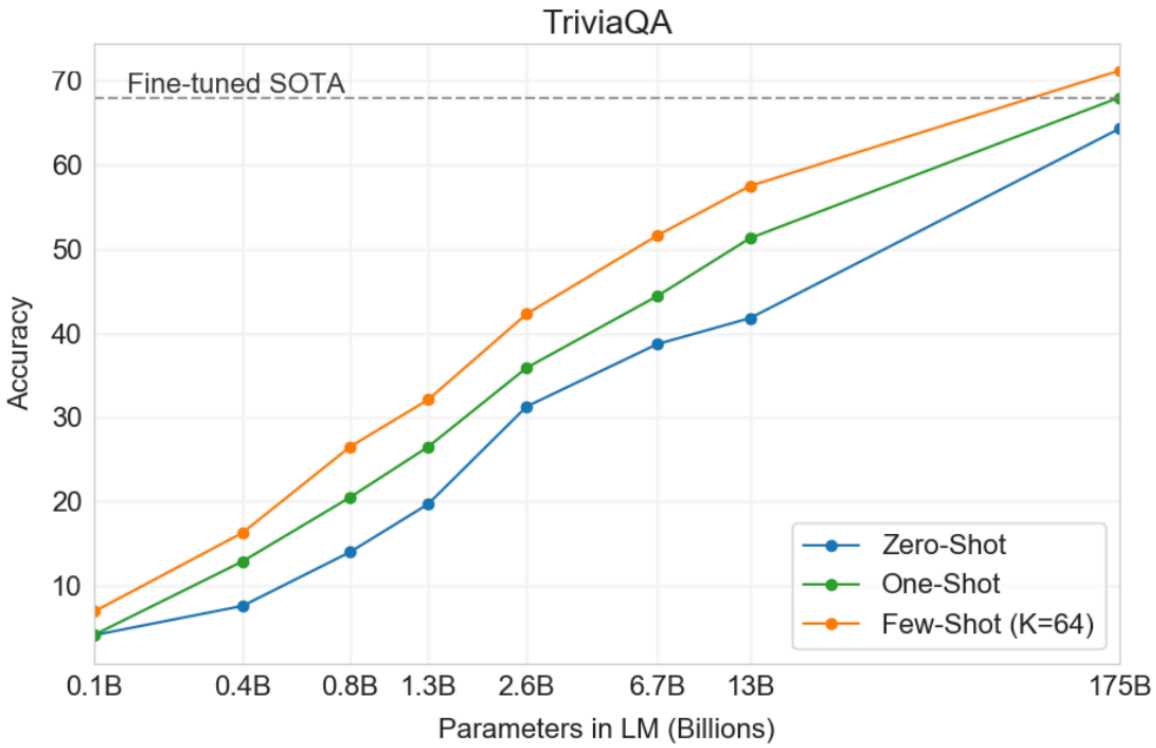
Few-shot

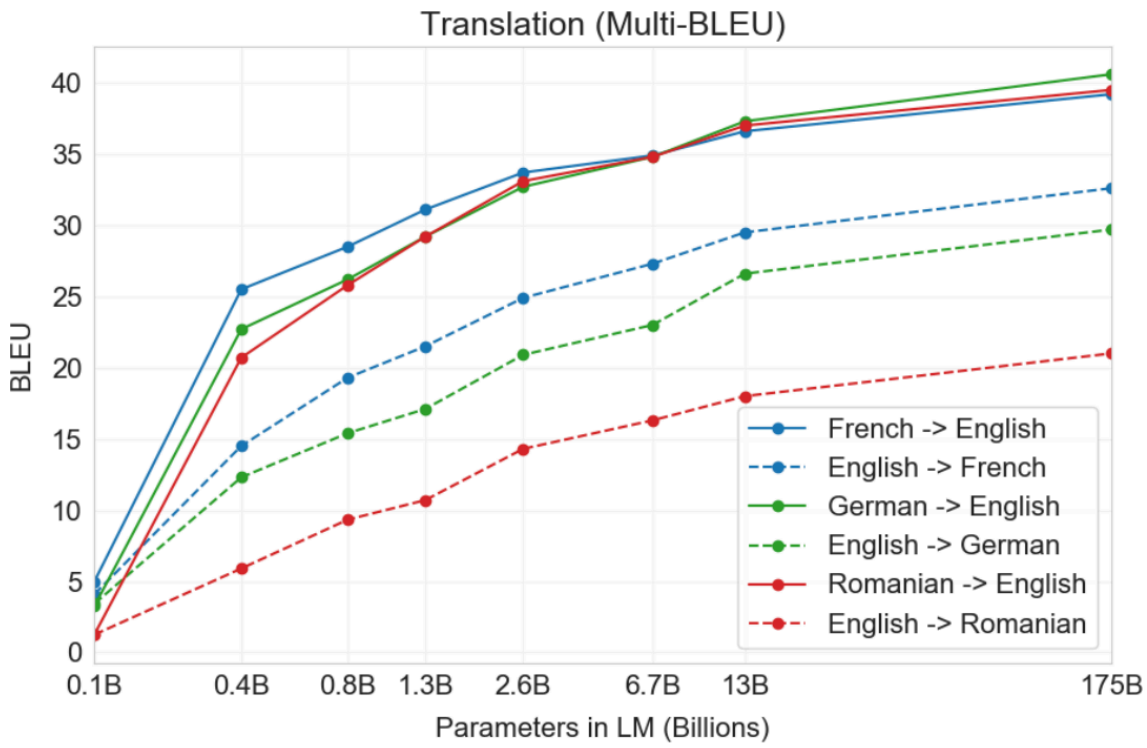
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Brown et al. 2020 - Language Models are Few-Shot Learners

How good are LLMs?

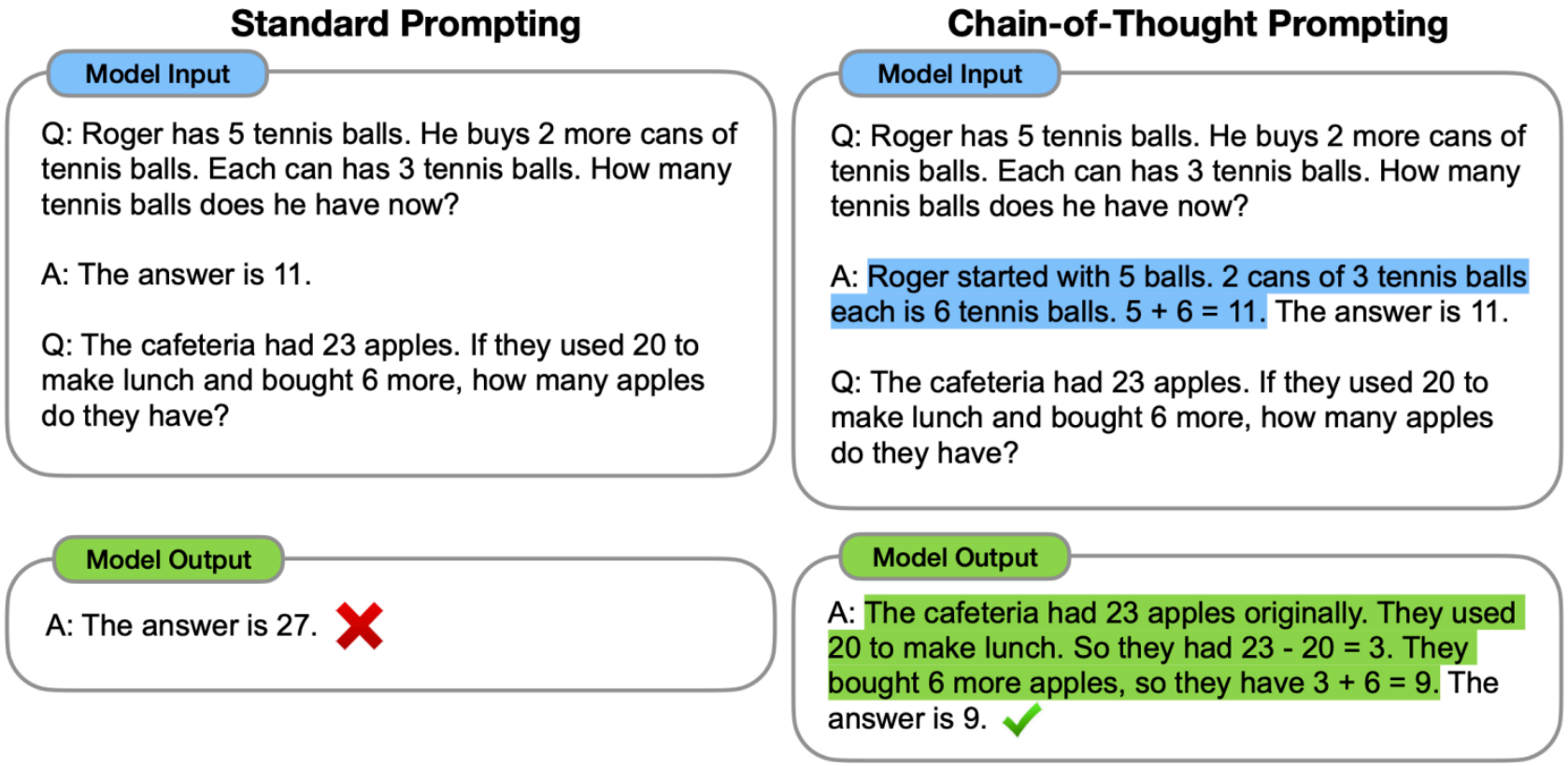




- As model increase in size, zero-shot and few-shot prompting performs better

Brown et al. 2020 - Language Models are Few-Shot Learners

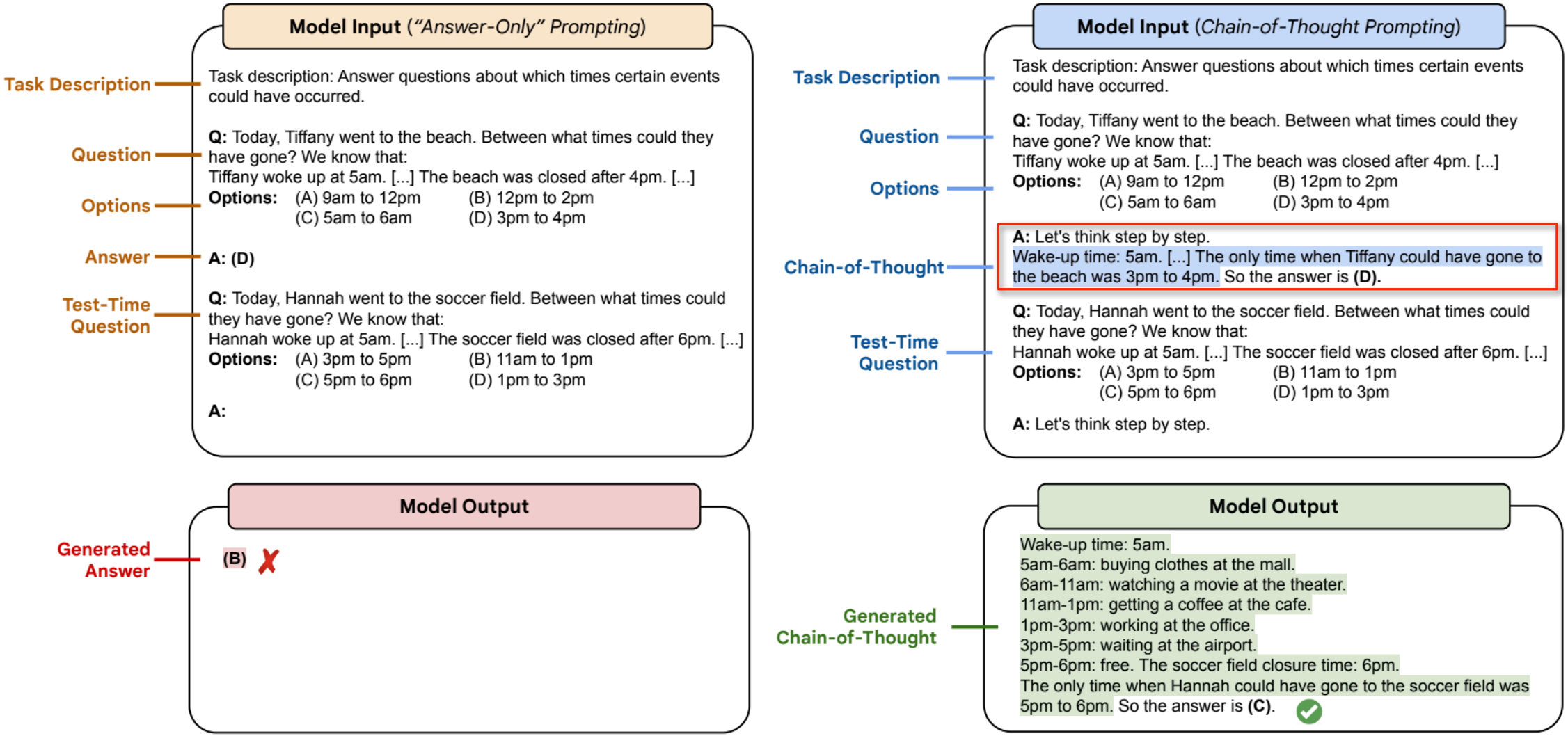
Chain-of-thought prompting (CoT)



Main idea: show a model how to decompose a problem into simpler sub-problems

Wei et al. 2022 - Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Chain-of-thought prompting (CoT)



Wei et al. 2022 - Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Many other factors that we evaluate, like:

Size

Big models take lots of GPUs and time to train, memory to store

Energy usage

Can measure kWh or kilograms of CO_2 emitted

Fairness

Benchmarks measure gendered and racial stereotypes, or decreased performance for language from or about some groups.

Scaling Laws

LLM performance depends on

- Model size: the number of parameters not counting embeddings
- Dataset size: the amount of training data
- Compute: Amount of compute (in FLOPS or etc

Can improve a model by adding parameters (more layers, wider contexts), more data, or training for more iterations

The performance of a large language model (the loss) scales as a power-law with each of these three

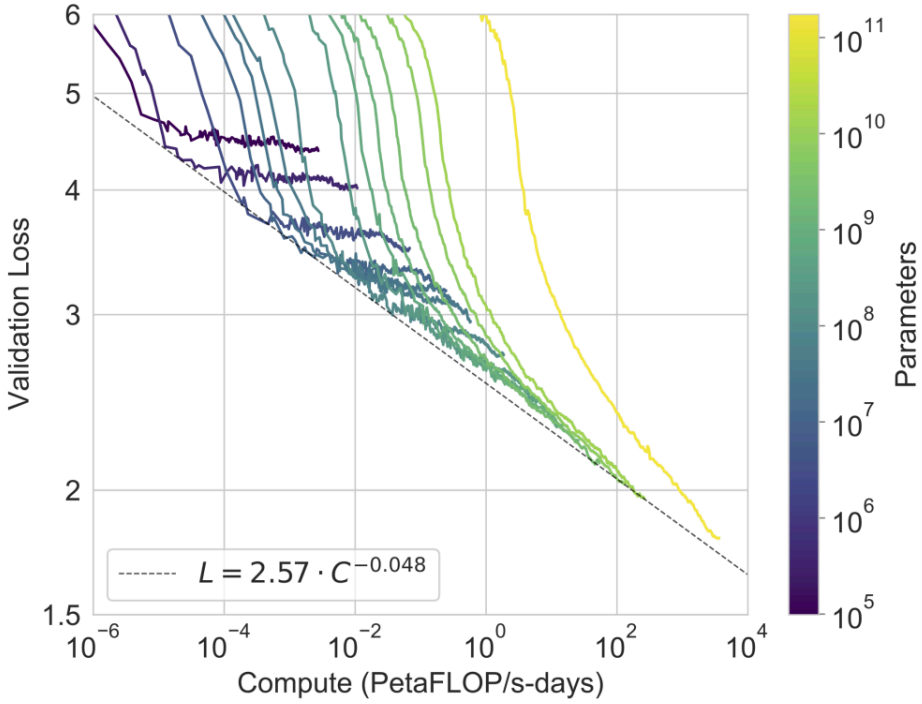
Scaling Laws

Loss *L* as a function of # parameters *N*, dataset size *D*, compute budget *C* *(*if other two are held constant)

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$$

$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$



Scaling laws can be used early in training to **predict** what the loss would be if we were to add more data or increase model size.

Number of non-embedding parameters N

$$
\begin{aligned}
N &\approx 2\, d\, n_{\text{layer}}(2\, d_{\text{attn}} + d_{\text{ff}}) \\
&\approx 12\, n_{\text{layer}}\, d^2 \\
&\quad (\text{assuming } \; d_{\text{attn}} = d_{\text{ff}}/4 = d)
\end{aligned}
$$

Thus GPT-3, with n = 96 layers and dimensionality d = 12288, has 12 × 96 × 122882 ≈ 175 billion parameters.

Hallucination

Chatbots May 'Hallucinate' More Often Than Many Realize

What Can You Do When A.I. Lies About You?

People have little protection or recourse when the technology creates and spreads falsehoods about them.

Air Canada loses court case after its chatbot hallucinated fake policies to a customer

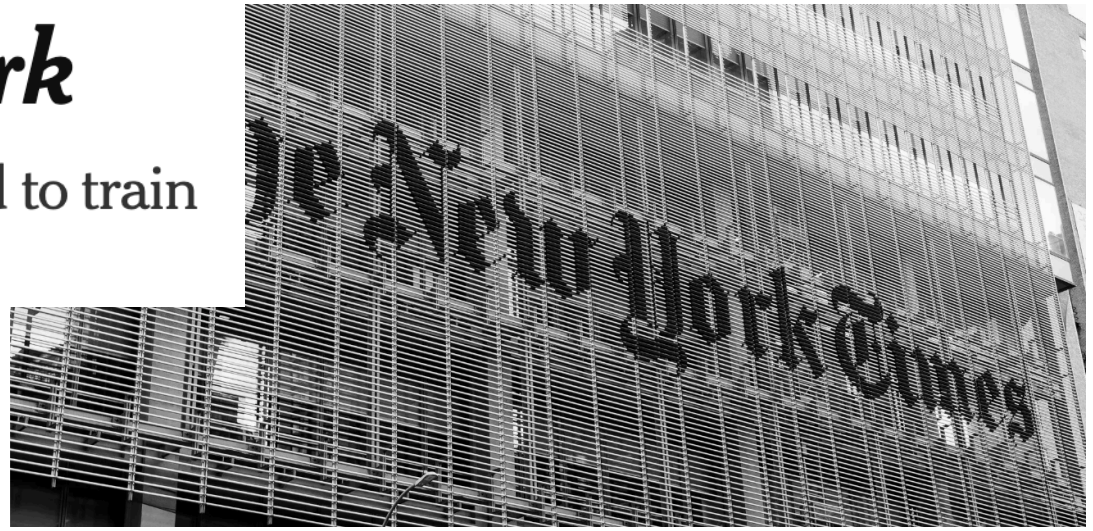
The airline argued that the chatbot itself was liable. The court disagreed.

Copyright

Authors Sue OpenAI Claiming Mass Copyright Infringement of Hundreds of Thousands of Novels

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.



Privacy

Toxicity and Abuse

The New AI-Powered Bing Is Threatening Users.

Cleaning Up ChatGPT Takes Heavy Toll on Human Workers

Contractors in Kenya say they were traumatized by effort to screen out descriptions of violence and sexual abuse during run-up to OpenAI's hit chatbot

Misinformation

Chatbots are generating false and misleading information about U.S. elections

[15 minute break]