Pre-Training Strategies

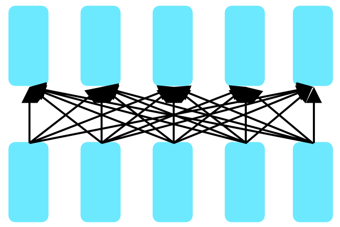
Encoder-Decoder and Decoder-only Models

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

Introduction to Large Language Models

Slide Acknowledgements:
Colin Raffel, Daniel Khashabi, Graham Neubig, Greg Durrett, Tatsunori Hashimoto

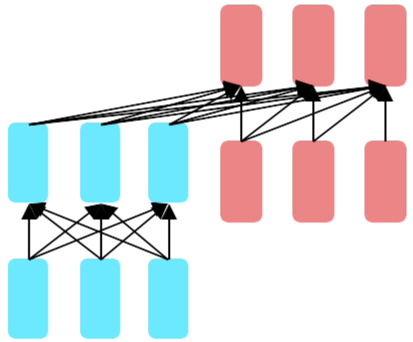
Pre-Training for Different Types of Architectures



Encoder-only

e.g. BERT (already discussed)

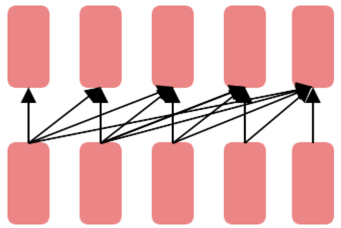
- Gets bi-directional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoder

e.g. BART, T5

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoder-only

e.g. GPT, Llama

- Language models!
- Nice to generate from; can't condition on future words

Pre-Training Encoder-Decoder Models

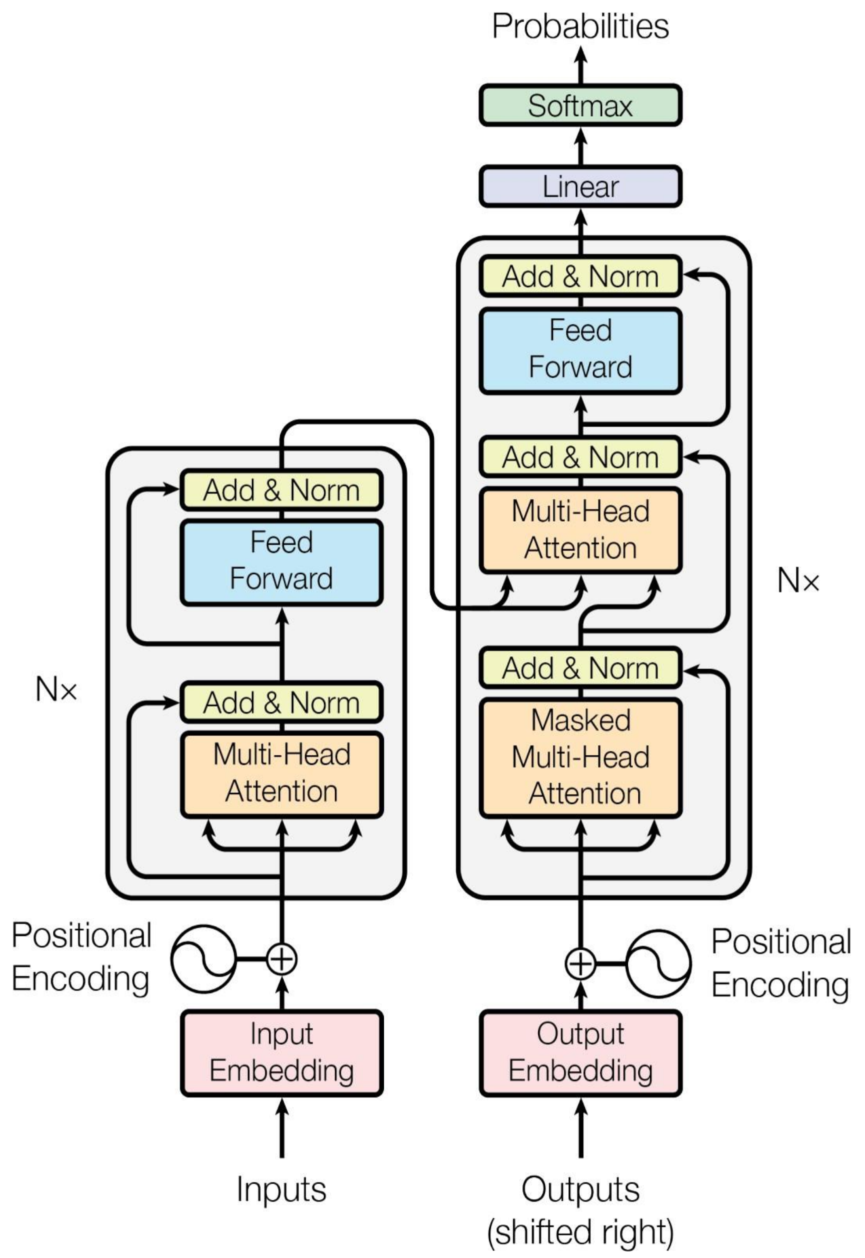
BART and T5

Pre-Training Encoder-Decoder Models

- Masked LMs: trained bidirectionally but with masking
- How can we pre-train a model for P(**y** | **x**)?
- Why was BERT effective?
  - Predicting a mask requires some kind of text "understanding".
- What would it take to do the same for sequence prediction?

Recall: Encoder-Decoder Architecture

- Standard Transformer Architecture
- Decoder attends back to the input. But the input doesn't change, so this just needs to be encoded once.



Pre-Training Encoder-Decoder Models

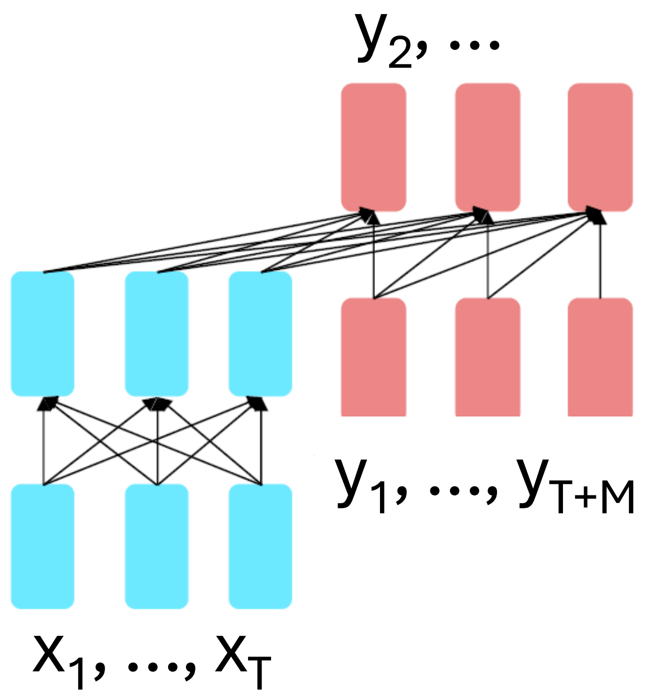
- For **encoder-decoders**, we could do something like **language modeling**, but where a prefix of every input is provided to the encoder and is not predicted.

$$h_1, \ldots, h_T = \mathit{Encoder}(x_1, \ldots, x_T)$$

$$h_{T+1}, \ldots, h_{T+M} = \mathit{Decoder}(y_1, \ldots, y_{i-1}, h_1, \ldots, h_T)$$

$$P(y_i \mid y_{<i}, h_{1:T}) = \mathit{Softmax}(Wh_i + b)$$

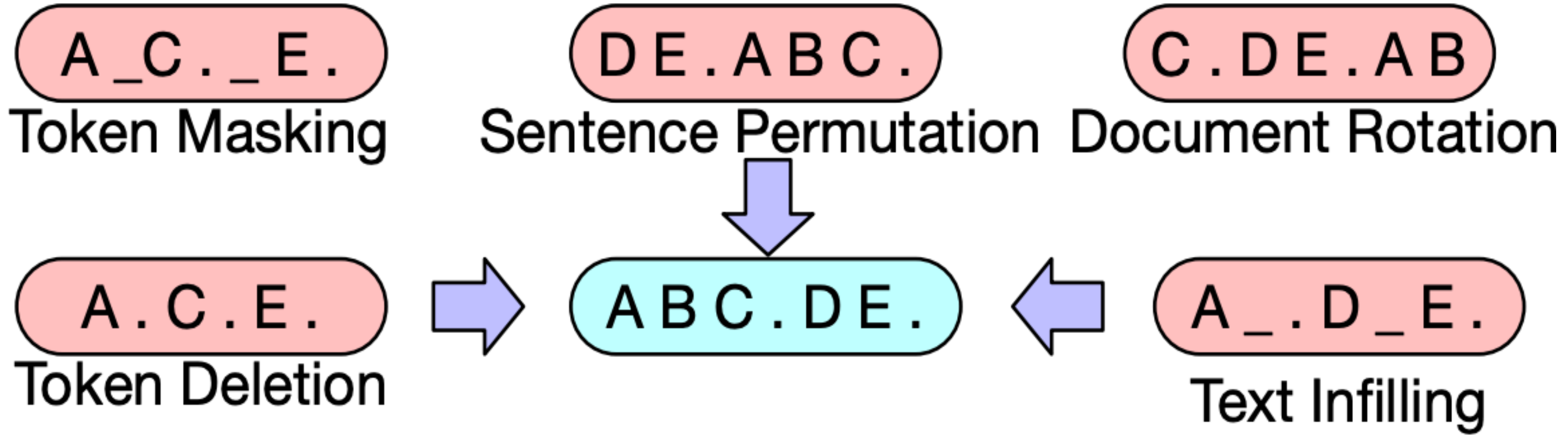
The **encoder** portion benefits from bidirectional context; the **decoder** portion is used to train the whole model through language modeling.



Pre-Training Encoder-Decoder Models

- How can we pre-train a model for P(**y** | **x**)?
- **Requirements:**
  1. should use unlabeled data
  2. should force a model to attend from **y** back to **x**

Pre-Training BART



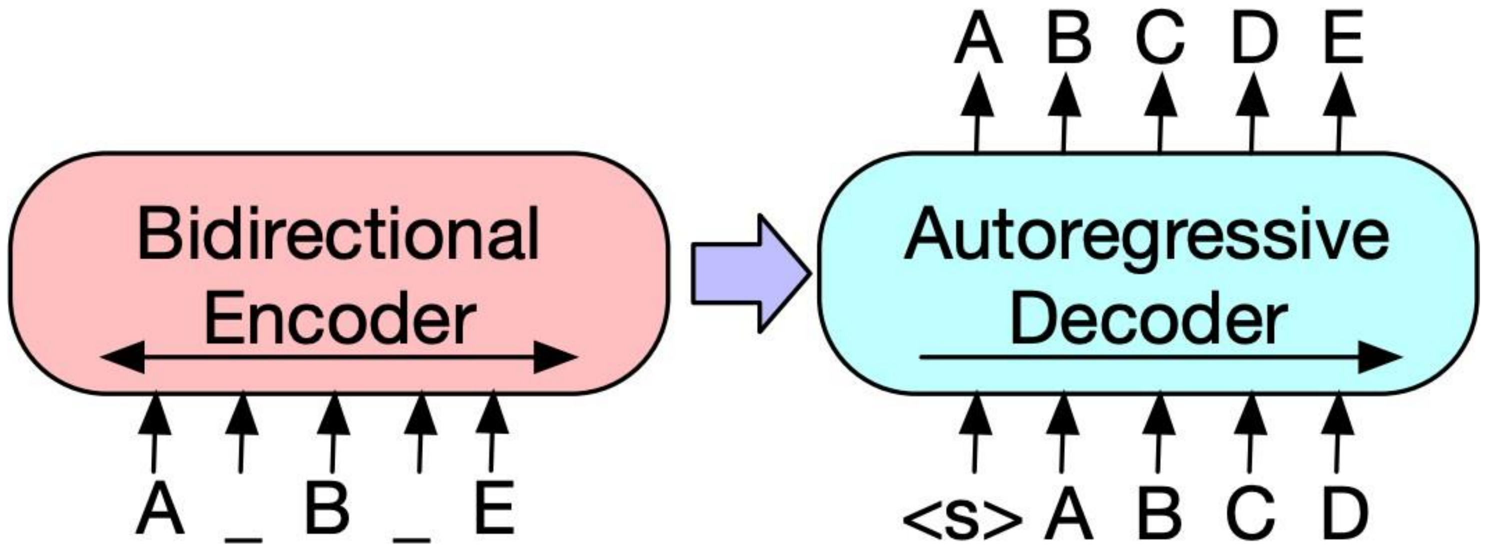
Infilling is longer spans than masking

- Several possible strategies for corrupting a sequence are explored in the BART paper.

Lewis et al. (2019), "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension"

Pre-Training BART

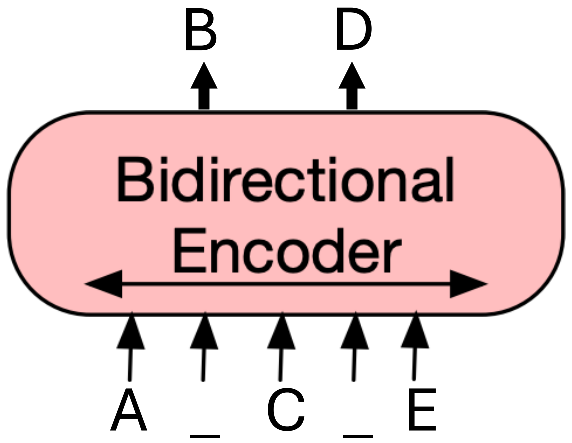
- Sequence-to-sequence Transformer trained on this data: permute/mask/delete tokens, then predict full sequence autoregressively.

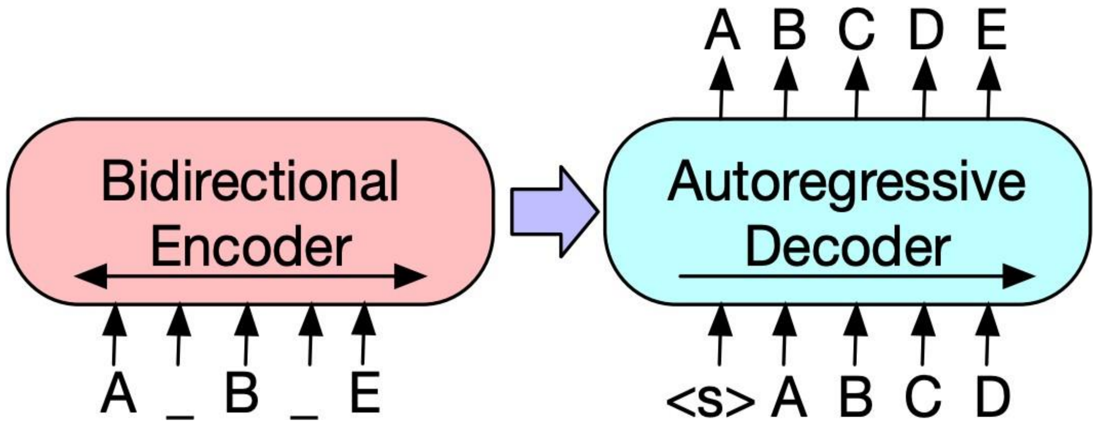


Lewis et al. (2019), “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”

BERT vs. BART

- **BERT:** only an encoder, trained with masked language modeling objective. Cannot generate text or do Seq2Seq tasks (in standard form).





- **BART:** consists of both an encoder and a decoder. Can also use just the encoder wherever we would use BERT.

BART for Summarization

- **Pre-train** on the BART task: take random chunks of text, noise them according to the schemes described, and try to “decode” the clean text
- **Fine-tune** on a summarization dataset: a news article is the input and a summary of that article is the output (usually 1-3 sentences depending on the dataset)
- Can achieve good results even with **few summaries to fine-tune on**, compared to basic seq2seq models which require 100k+ examples to do well

BART for Summarization: Output Example

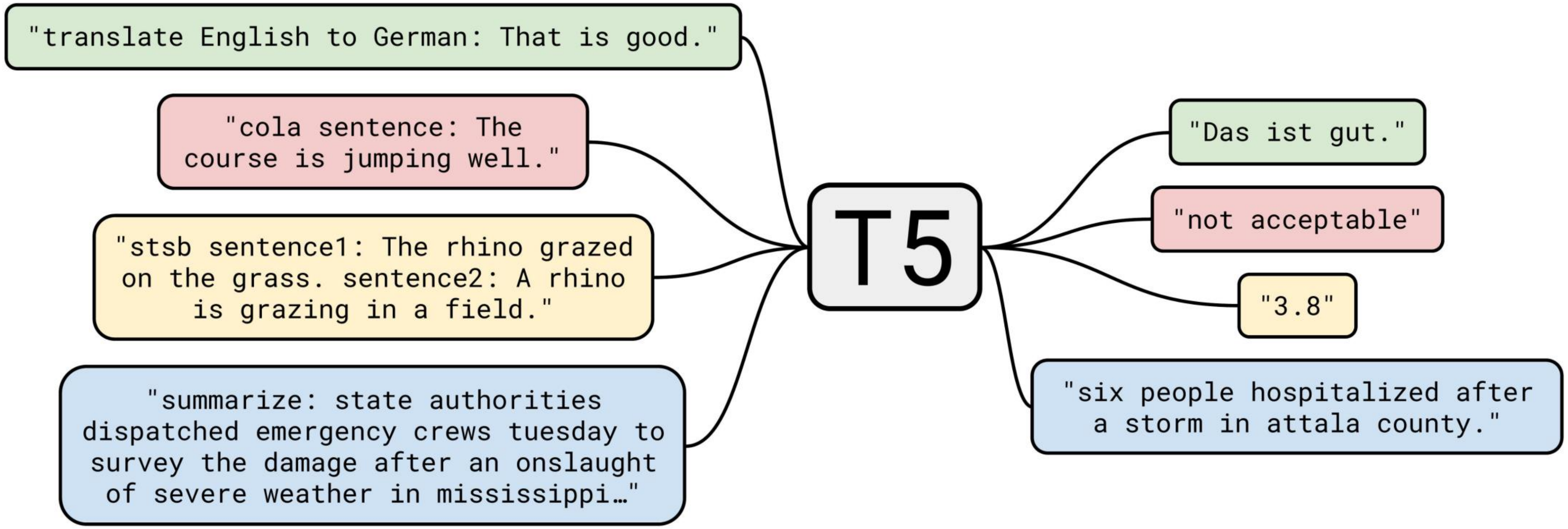
PG&E stated it scheduled the blackouts in response to forecasts for high winds amid dry conditions. The aim is to reduce the risk of wildfires. Nearly 800 thousand customers were scheduled to be affected by the shutoffs which were expected to last through at least midday tomorrow.

$\downarrow$

Power has been turned off to millions of customers in California as part of a power shutoff plan.

Lewis et al. (2019)

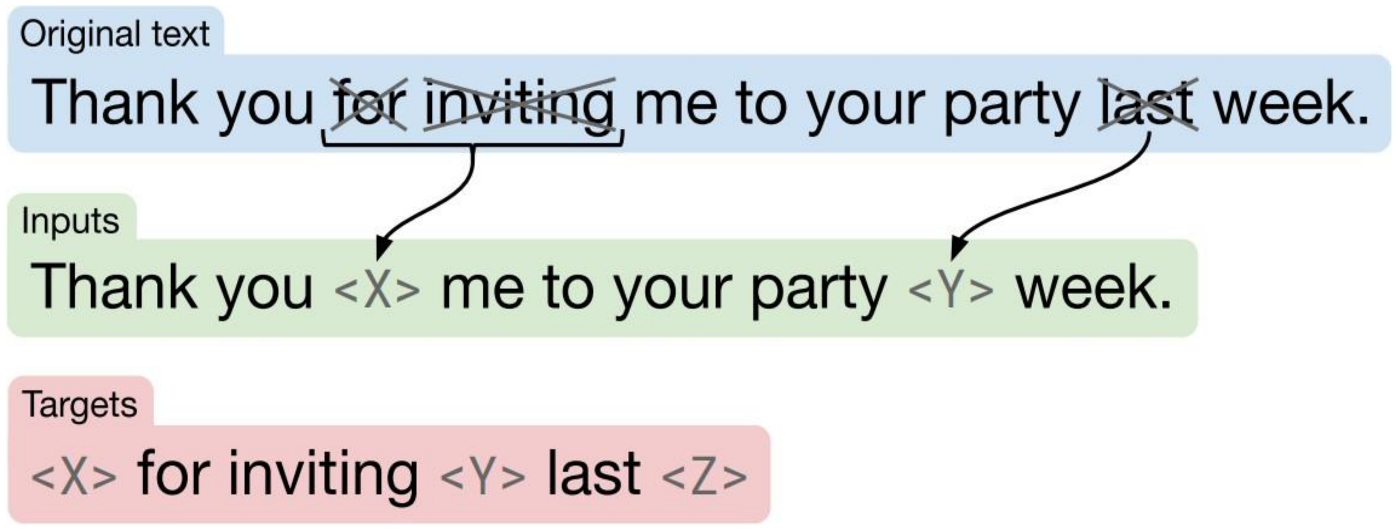
T5: Text-to-Text Transfer Transformer


"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Raffel et al. (2019), "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer"

Pre-Training T5

- **Pre-training:** similar denoising scheme to BART (they were released within a week of each other in fall 2019)
- **Input:** text with gaps ; **Output:** a series of phrases to fill those gaps.



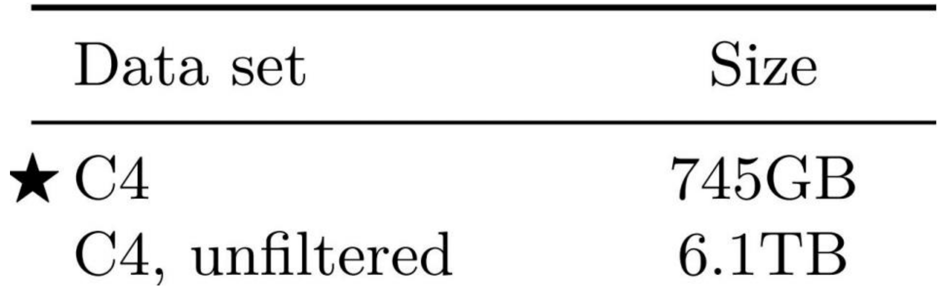
Original text
Thank you ~~for inviting~~ me to your party ~~last~~ week.
Inputs
Thank you <X> me to your party <Y> week.
Targets
<X> for inviting <Y> last <Z>

Replace different-length spans from the input with unique placeholders; decode out the spans that were removed!

Raffel et al. (2019)

C4: The Data

- C4: Colossal Clean Crawled Corpus
  - Web-extracted text
  - English language only
  - 750GB

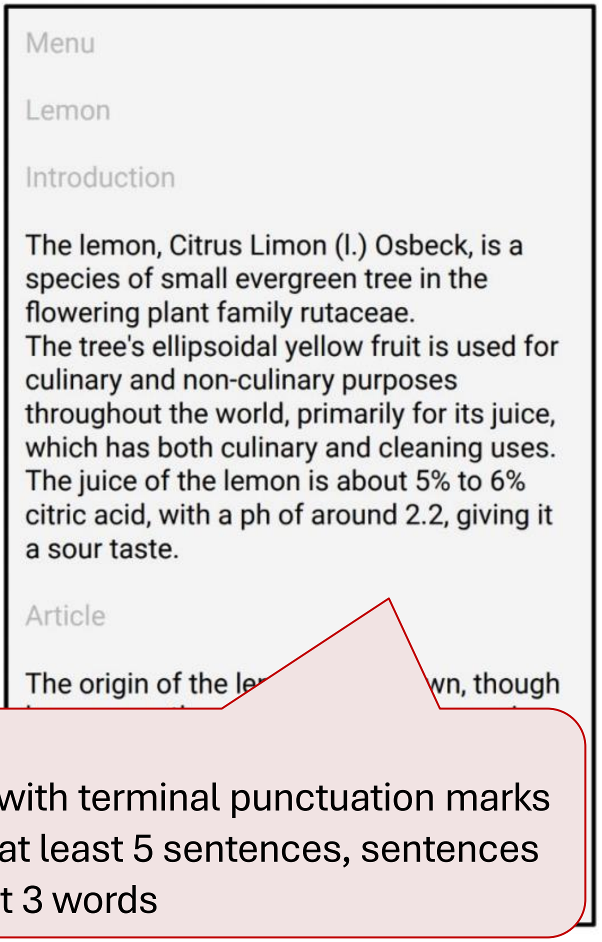


Data set	Size
★ C4	745GB
C4, unfiltered	6.1TB

C4: The Data

Remove any:

- References to JavaScript
- “Lorem ipsum” text — placeholder text commonly used to demonstrate the visual form of a document



Menu

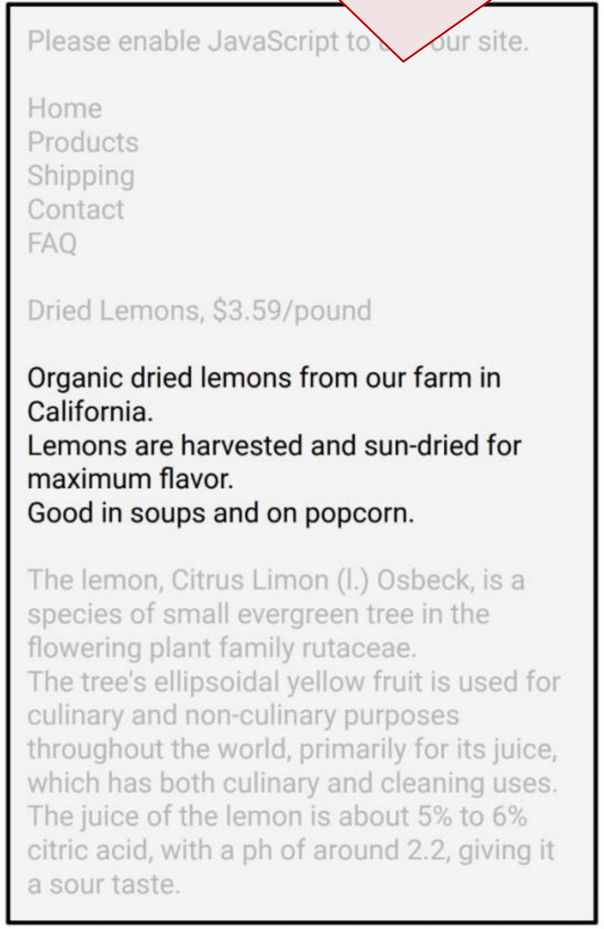
Lemon

Introduction

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.
The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the le wn, though



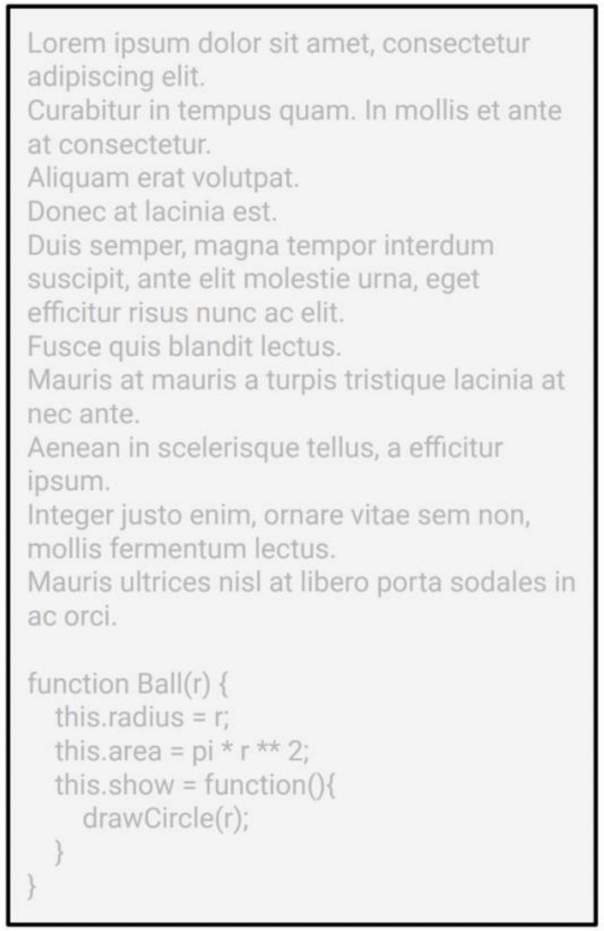
Please enable JavaScript to our site.

Home
Products
Shipping
Contact
FAQ

Dried Lemons, $3.59/pound

Organic dried lemons from our farm in California.
Lemons are harvested and sun-dried for maximum flavor.
Good in soups and on popcorn.

The lemon, Citrus Limon (l.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae.
The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses.
The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.



Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Curabitur in tempus quam. In mollis et ante at consectetur.
Aliquam erat volutpat.
Donec at lacinia est.
Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit.
Fusce quis blandit lectus.
Mauris at mauris a turpis tristique lacinia at nec ante.
Aenean in scelerisque tellus, a efficitur ipsum.
Integer justo enim, ornare vitae sem non, mollis fermentum lectus.
Mauris ultrices nisl at libero porta sodales in ac orci.

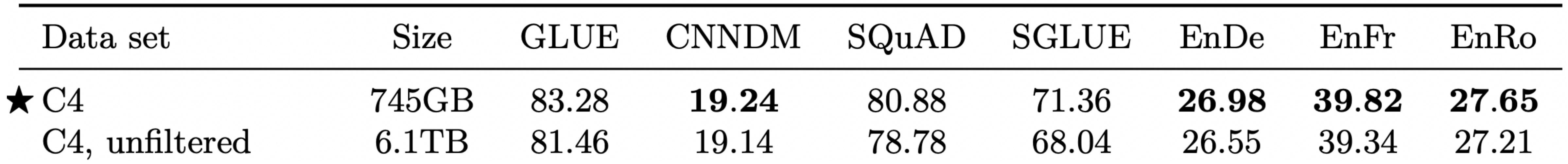
```
function Ball(r) {
  this.radius = r;
  this.area = pi * r ** 2;
  this.show = function(){
    drawCircle(r);
  }
}
```

Retain:

- Sentences with terminal punctuation marks
- Pages with at least 5 sentences, sentences with at least 3 words

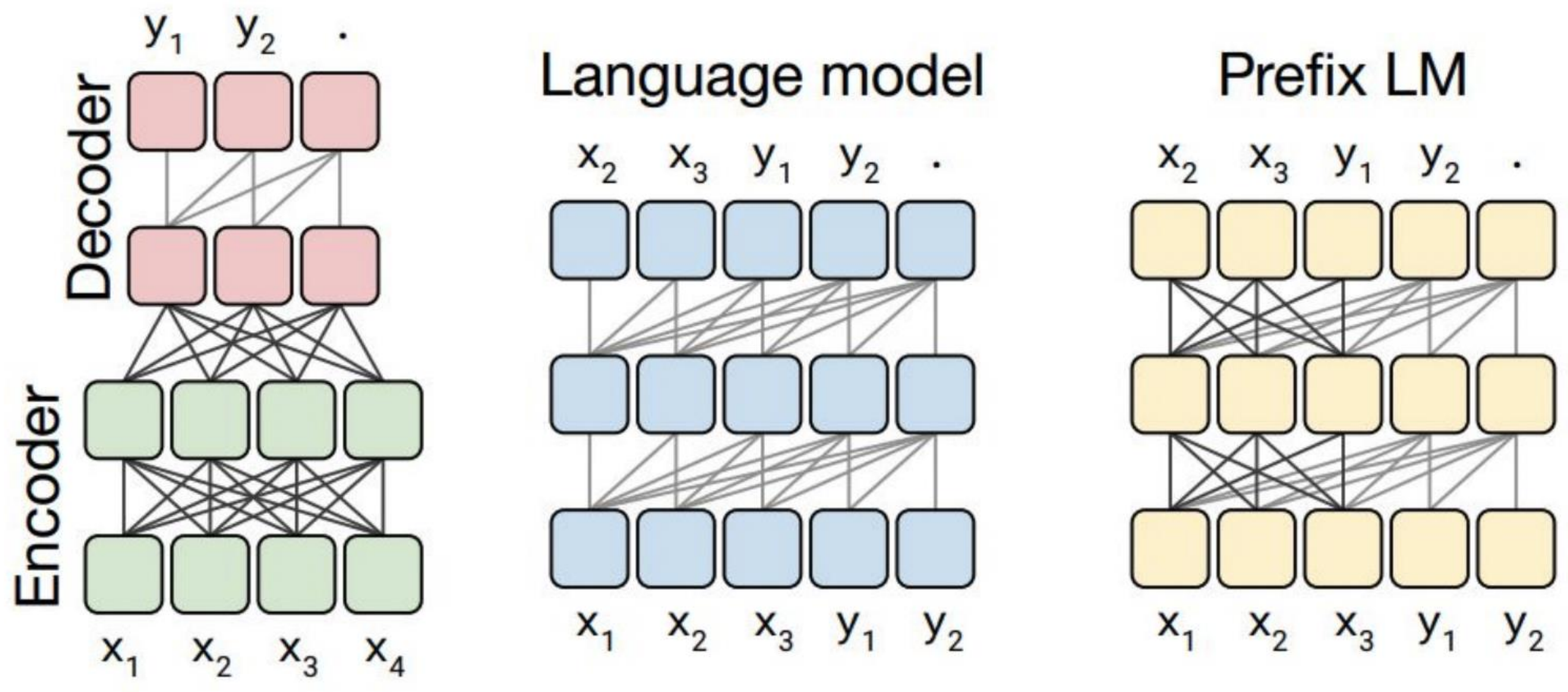
Pre-Training Data: Experiment

- Takeaway:
  - Clean and compact data is better than large, but noisy data.
  - Pre-training on in-domain data helps.



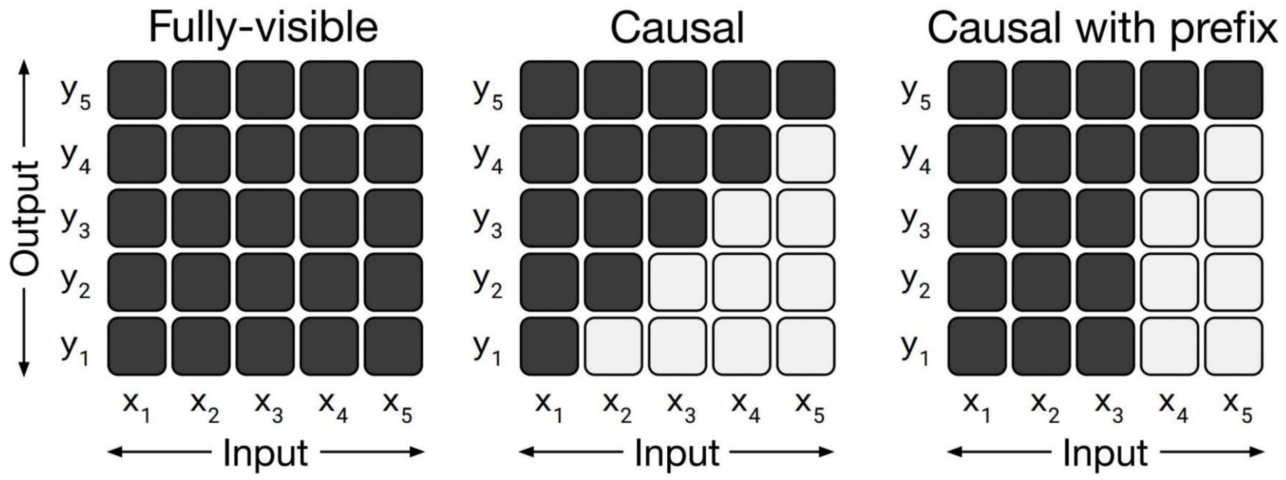
Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21

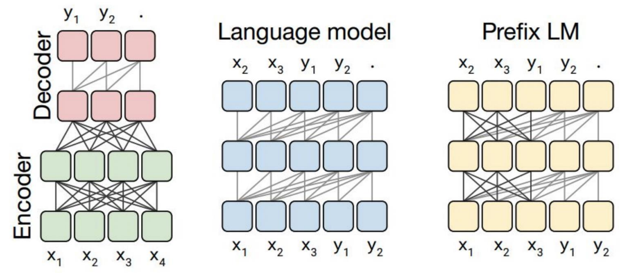
Architectures: Different Choices



Architectures: Different Attention Masks

- **Fully visible mask** allows the self attention mechanism to attend to the full input.
- A **causal mask** doesn't allow output elements to look into the future.
- **Causal mask with prefix** allows to fully-visible masking on a portion of input.

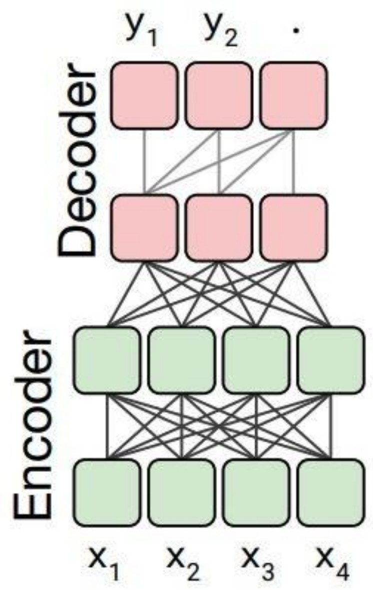




Architectural Variants: Experiments

Evaluated for classification tasks.

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**

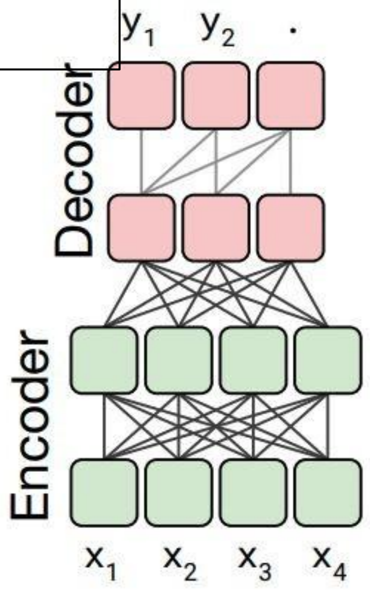


Architectural Variants: Experiments

Evaluated for classification tasks.

	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**

Input: Thank you for <X> me to your party <Y>. Target: <X> inviting <Y> last week.

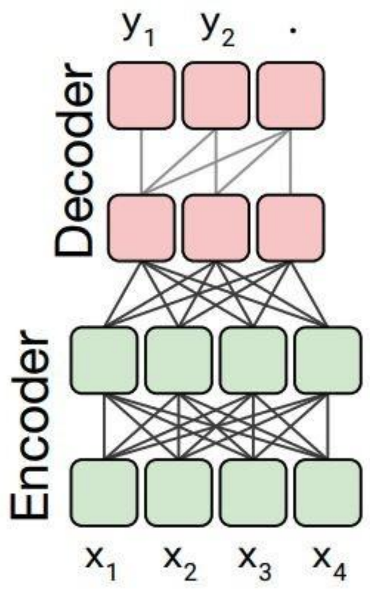


Architectural Variants: Experiments

Evaluated for classification tasks.

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**

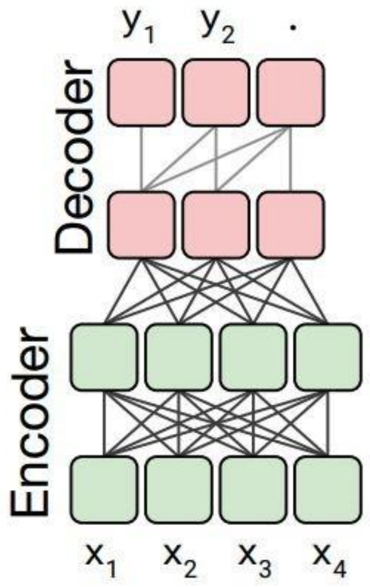
Number of parameters



Architectural Variants: Experiments

Evaluated for classification tasks.

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**

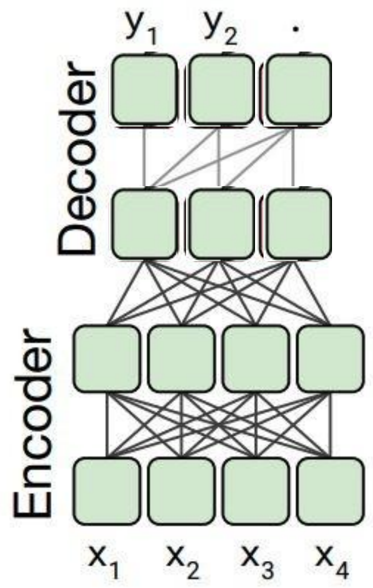


Number of FLOPS

Architectural Variants: Experiments

Evaluated for classification tasks.

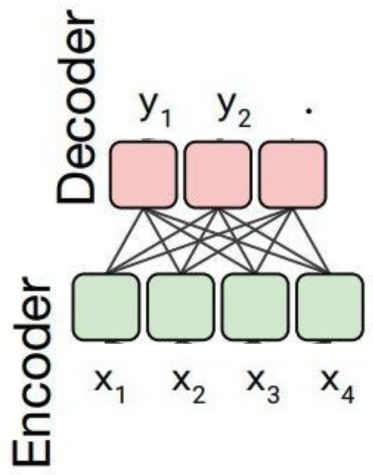
Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**



Architectural Variants: Experiments

Evaluated for classification tasks.

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

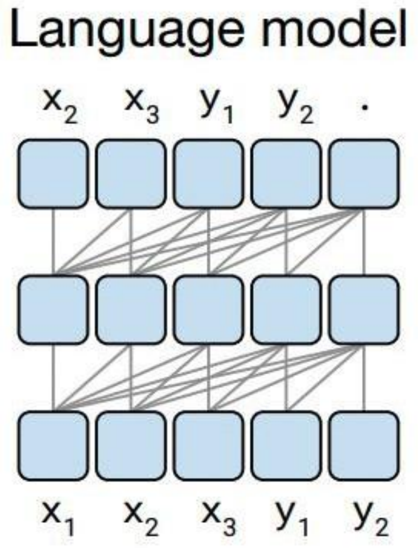


Architectural Variants: Experiments

Evaluated for classification tasks.

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

Language model is decoder-only

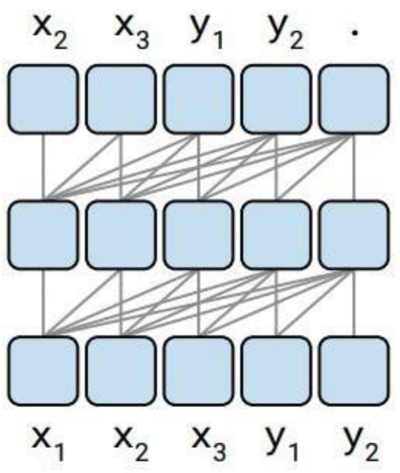


Architectural Variants: Experiments

Evaluated for classification tasks.

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

Language model



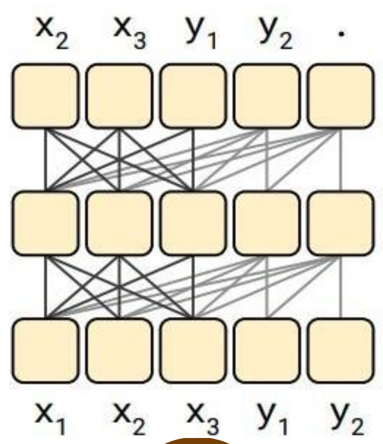
LM looks at both input and target, while encoder only looks at input sequence and decoder looks at output sequence.

Architectural Variants: Experiments

Evaluated for classification tasks.

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

Prefix LM



Architectural Variants: Experiments

Evaluated for classification tasks.

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

Takeaways:

1. Halving the number of layers in encoder and decoder hurts the performance.

Architectural Variants: Experiments

Evaluated for classification tasks.

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

Takeaways:

1. Halving the number of layers in encoder and decoder hurts the performance.
2. Performance of Encoder-Decoder with shared params is almost on-par with prefix LM.

T5: Pre-Training Objectives

- **Prefix language modeling**
  - **Input:** Thank you for inviting
  - **Output:** me to your party last week
- **BERT-style denoising**
  - **Input:** Thank you <M> <M> me to your party apple week
  - **Output:** Thank you for inviting me to your party last week
- **De-shuffling**
  - **Input:** party me for your to. last fun you inviting week Thanks.
  - **Output:** Thank you for inviting me to your party last week
- **Replace spans**
  - **Input:** Thank you <X> me to your party <X> week
  - **Output:** <X> for inviting <Y> last <Z>
- **Drop tokens**
  - **Input:** Thank you me to your party week .
  - **Output:** for inviting last

Pre-Training Objectives: Experiments

- All the variants perform similarly
- "Replace corrupted spans" and "Drop corrupted tokens" are more appealing because **target sequences are shorter, speeding up training**.

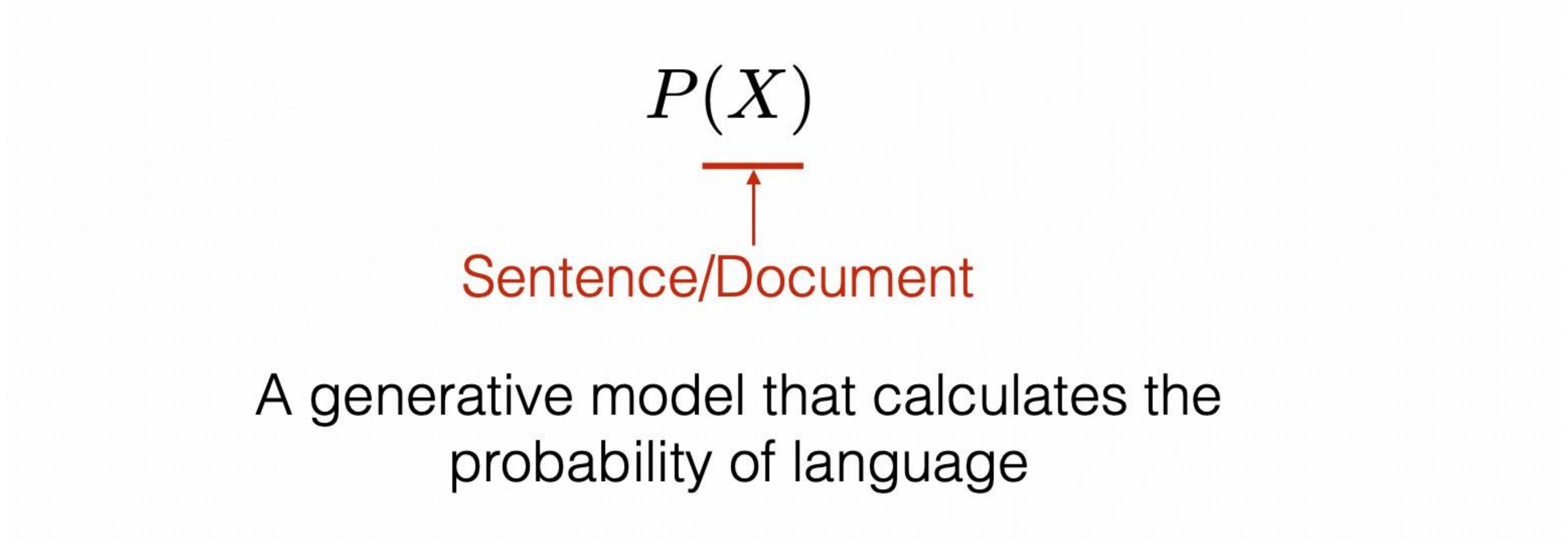
Assuming Enc-Dec architecture.
Evaluated for classification tasks.

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
Prefix language modeling	80.69	18.94	77.99	65.27	**26.86**	39.73	**27.49**
Deshuffling	73.17	18.59	67.61	58.47	26.11	39.30	25.62
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
★ 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**

Pre-Training Decoder-only Models

GPT and Llama

Recall: Probabilistic Language Models



$P(X)$

$\uparrow$

Sentence/Document

A generative model that calculates the probability of language

Auto-regressive Language Models

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Token}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

Next Token Prediction

- This is essentially **classification**!
  - We can think of neural language models as neural classifiers. They classify prefix of a text into |V| classes, where the classes are vocabulary tokens.

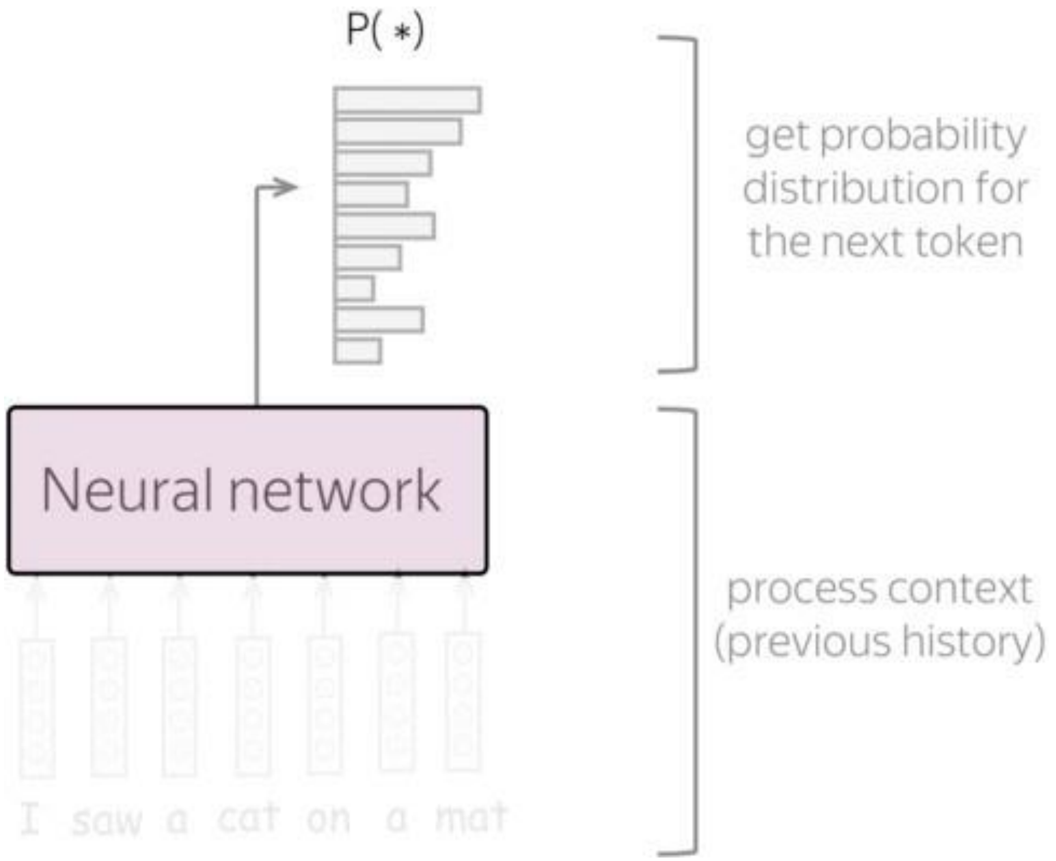
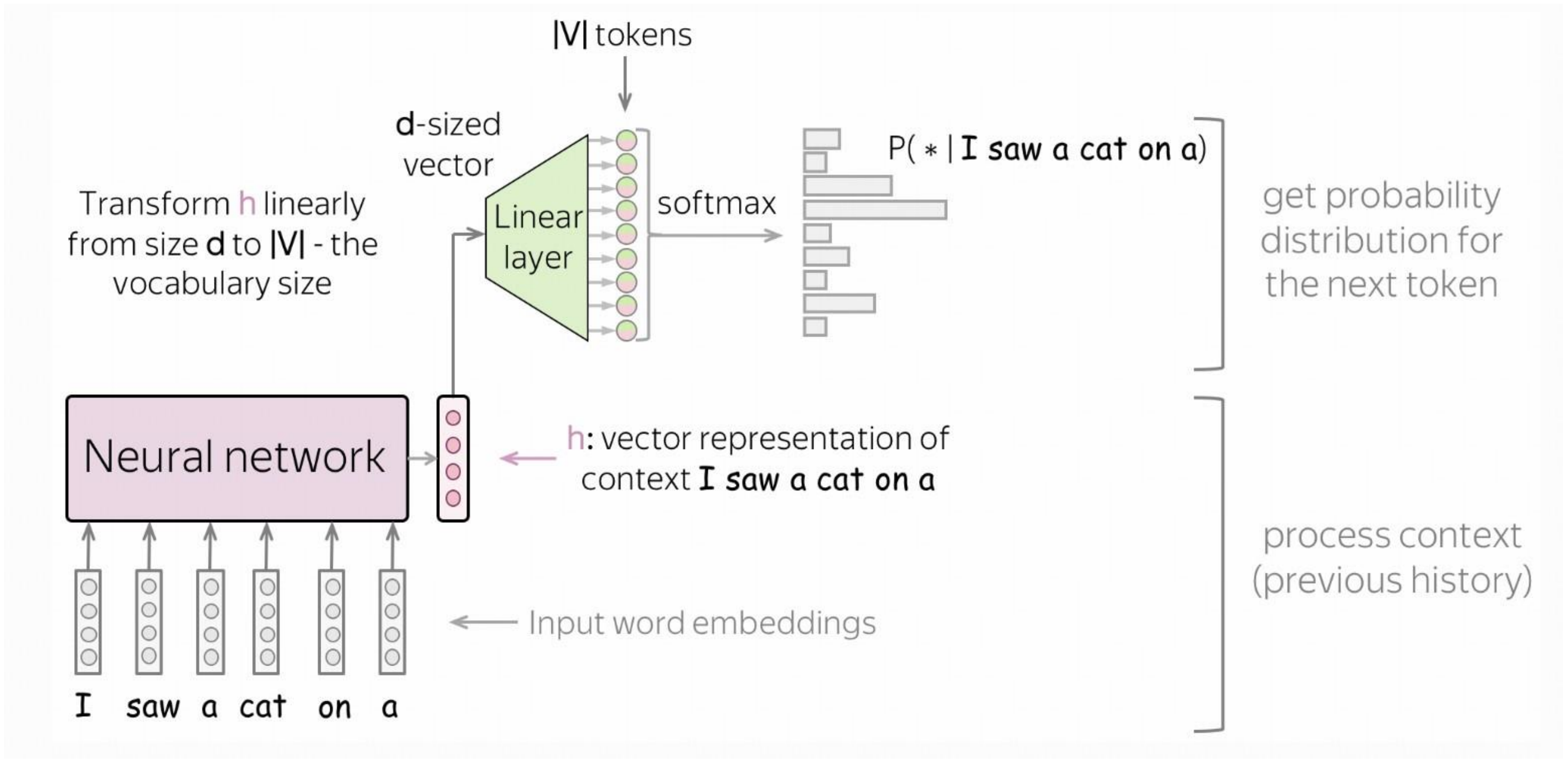

P(*)
get probability distribution for the next token
Neural network
process context (previous history)
I saw a cat on a mat

Image Credit: https://lena-voita.github.io/nlp_course/language_modeling.html

Next Token Prediction


|V| tokens
d-sized vector
Transform h linearly from size d to |V| - the vocabulary size
Linear layer
softmax
P(* | I saw a cat on a)
get probability distribution for the next token
Neural network
h: vector representation of context I saw a cat on a
process context (previous history)
Input word embeddings
I saw a cat on a

- Feed word embedding for previous (context) words into a network.
- Get vector representation of context from the network.
- From this vector representation, predict a probability distribution for the next token.

Image Credit: https://lena-voita.github.io/nlp_course/language_modeling.html

Encoders vs. Decoders

- BERT is a Transformer **Encoder**: bidirectional attention, trained with masked language modelling.

$$P\ (x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

- GPT and many other Transformer language models (e.g., LLaMA) are **Decoders**: unidirectional attention, **trained to predict the next token**.

$$P\ (x_i \mid x_1, \ldots, x_{i-1})$$

Encoders vs. Decoders

- **Encoder: $P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$**
  - **To use in practice:** Ignore this probability distribution. Fine-tune the model for some other task P(y | x)
- **Decoder: $P(x_i \mid x_1, \ldots, x_{i-1})$**
  - You can treat this like an encoder: ignore this probability distribution and train a model for P(y | x). But encoders are better for this due to bidirectional attention
  - **To use in practice:** we use this model to actually ***generate*** text

Training Loss – Recap: Maximum Likelihood Estimation

- **Given**
  - A parametric family of probability distributions p_θ
  - n independent & identically distributed observations $s_1, \dots, s_n$
- **Task**
  - Find the parameter θ^* that most likely resulted in the observed data, that is

$$\begin{aligned} &\arg\max_\theta p_\theta(s_1, \dots, s_n) \\ &= \arg\max_\theta \log p_\theta(s_1, \dots, s_n) \\ &= \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(s_i) \end{aligned}$$

MLE from Sentence Data

- Given a sentence $s = (t_1, \dots, t_m)$, the probability of the sentence can be written as

$$p_\theta(s) = p_1(t_1) \prod_{j=1}^{m} p_{j+1}(t_{j+1} | t_1, \dots, t_j)$$

$$\log p_\theta(s) = \log p_1(t_1) + \sum_{j=1}^{m} \log p_{j+1}(t_{j+1} | t_1, \dots, t_j)$$

- Maximum Likelihood Estimation

Sampling a Sentence from an Auto-regressive LM

- Sample the first token t_1 from p_1
- For each subsequent step
  - Sample token t_{j+1} conditioned on the previous tokens $t_1, \ldots, t_j$

$$t_{j+1} \sim p_{j+1}(. | t_1, \ldots, t_j)$$

Since p_j are distributions over the vocabulary, they are easy to sample from !!

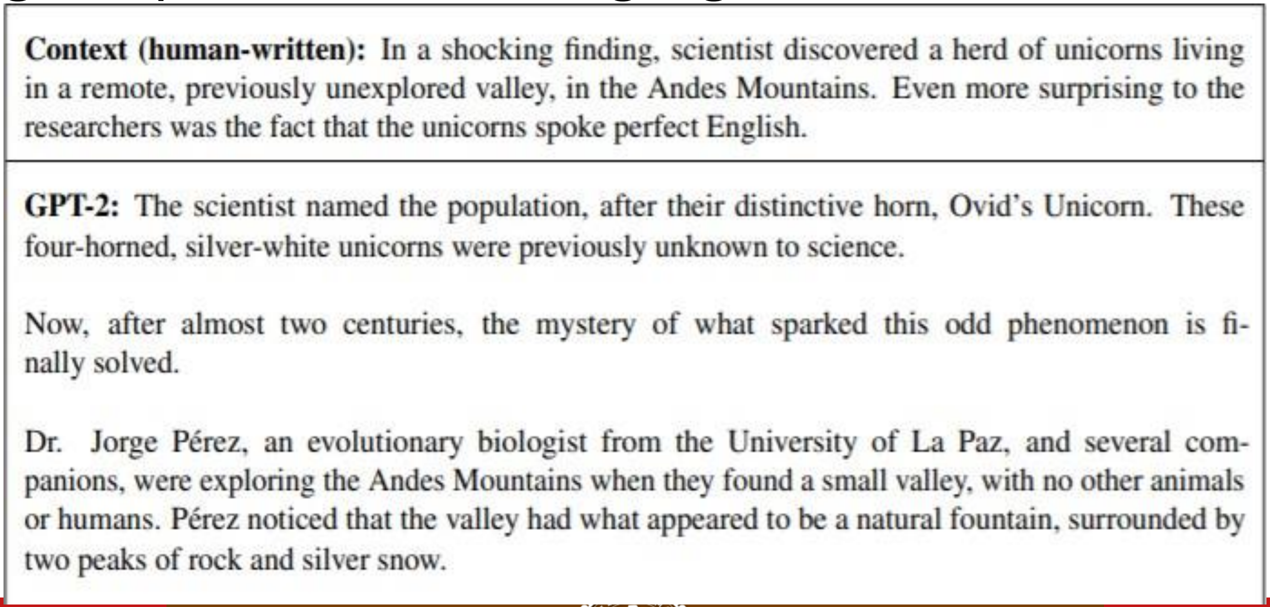
Generative Pre-trained Transformer (GPT)

- 2018's GPT was a big success in pretraining a decoder!
- **Transformer decoder with 12 layers, 117M parameters**.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
  - Trained on BooksCorpus: over 7000 unique books.
- Contains long spans of contiguous text, for learning long-distance dependencies.

Radford et al. (2018), "Improving Language Understanding by Generative Pre-Training"

Increasingly Convincing Generations by GPT-2

- We discussed how we can sample sentences from auto-regressive LMs for text generation.
  - This is how pre-trained decoders are used **in their capacities as language models.**
- **GPT-2,** a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.



Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

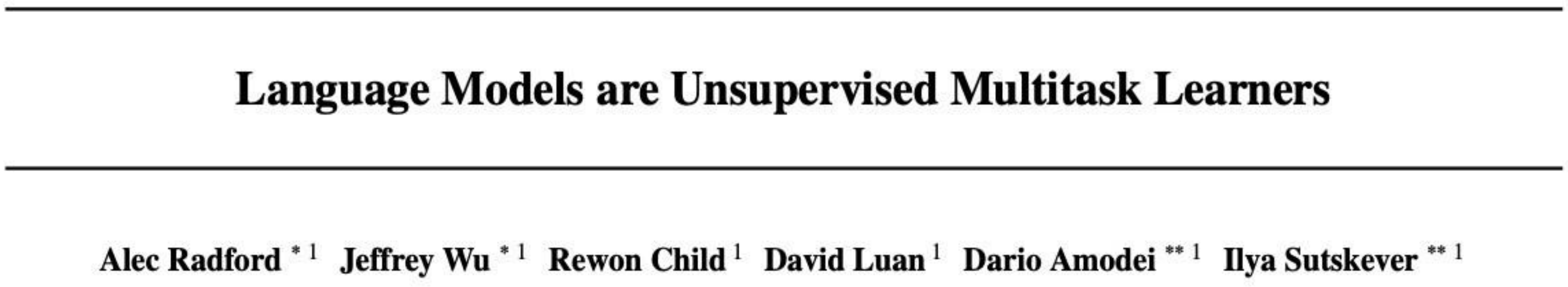
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

GPT-2

GPT-2 is identical to GPT-1, but:

- Has Layer normalization in between each sub-block
- Vocab extended to 50,257 tokens and context size increased from 512 to 1024
- Data: 8 million docs from the web (Common Crawl), minus Wikipedia



Language Models are Unsupervised Multitask Learners

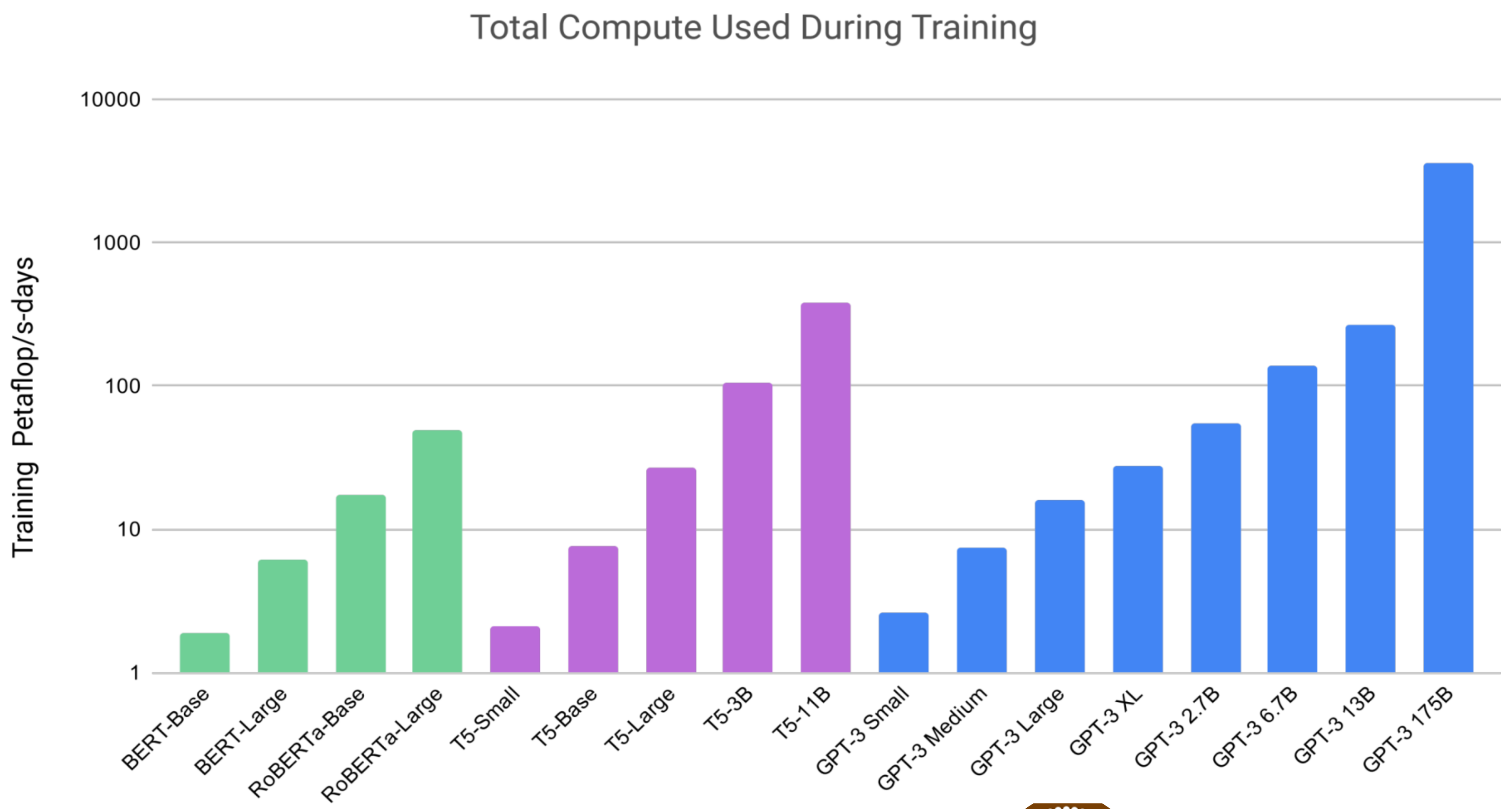
Alec Radford *[1] **Jeffrey Wu** *[1] **Rewon Child**[1] **David Luan**[1] **Dario Amodei** **[1] **Ilya Sutskever** **[1]

Pre-Training Cost (with GCP/AWS)

- **BERT:** Base \$500, Large \$7000
- **GPT-2** (as reported in other work): \$25,000
- This is for a single pre-training run…developing new pre-training techniques may require many runs.
- Fine-tuning these models can typically be done with a single GPU (but may take 1-3 days for medium-sized datasets).

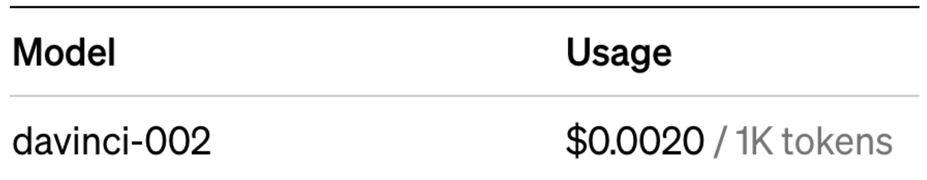
https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

GPT-3

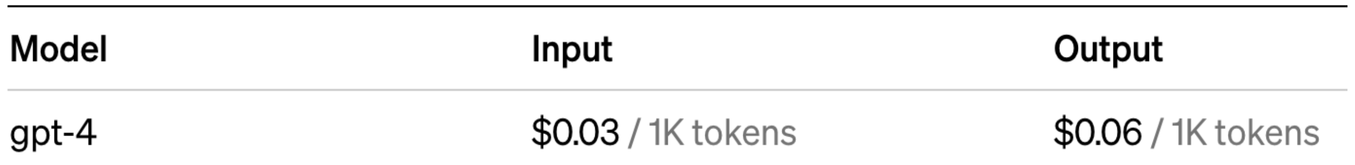


- **175B parameter model**
    - 96 layers, 96 heads, 12k-dim vectors
- Trained on Microsoft Azure, estimated to cost roughly **$10M**

GPT-4



Model	Usage
davinci-002	$0.0020 / 1K tokens



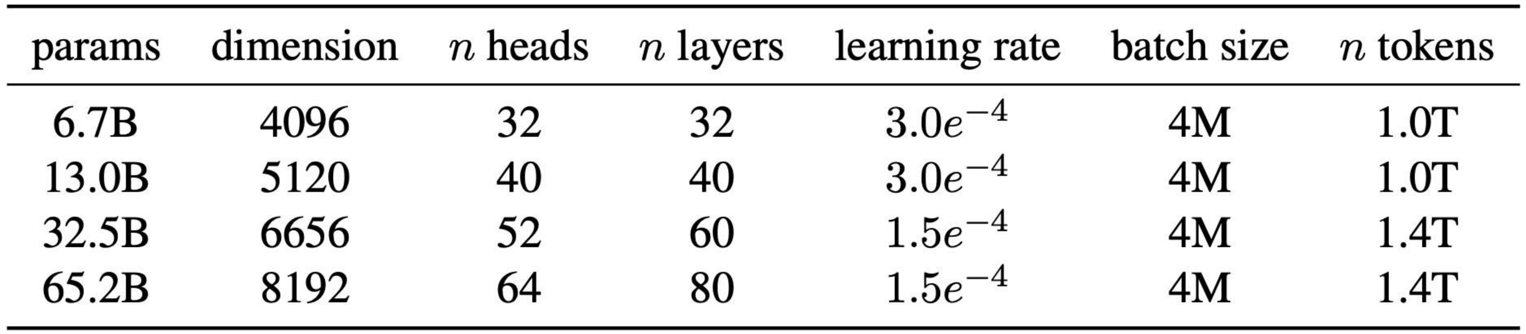
Model	Input	Output
gpt-4	$0.03 / 1K tokens	$0.06 / 1K tokens

- Transformer-based
  - The rest is …. mystery!
  - If we’re going based on costs, GPT-4 is ~15-30 times costlier than GPT3. That should give you an idea how its likely size!

- Note, these language models involve more than just pre-training.
  - Pre-training provides the foundation based on which we build the model.
  - We will discuss the later stages next week.

Llama: A Family of Open-Source LLMs from Meta AI

- **Llama-1 + Llama-2**



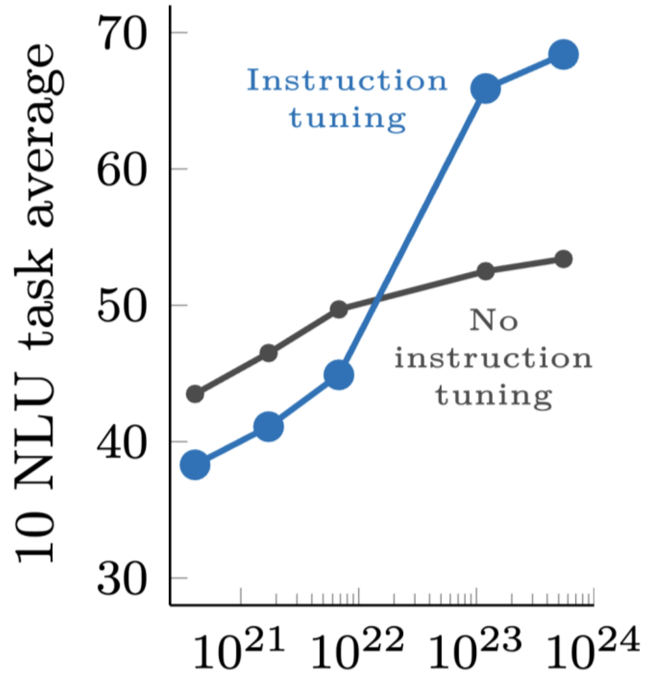
params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

Table 2: **Model sizes, architectures, and optimization hyper-parameters.**

- Models have mostly gotten smaller since GPT-3, but haven't changed much.
- **Tokenizer: Byte-Pair Encoding (BPE)** [Recall: we have already discussed this algorithm in lecture on 'Tokenization Strategies']
- **Rotary positional encodings**, a few other small architecture changes
- **Optimized mix of pre-training data**: Common Crawl, GitHub, Wikipedia, Books, etc.

Next Week: How to Make Pre-Trained LMs Work?

- Instruction Tuning
  - Finetune the pre-trained model to follow instructions
- Prompting and In-context Learning
  - Give few examples of the task that you want the model to solve
- Reinforcement Learning from Human Feedback (RLHF)
  - Train the LMs to align their outputs with human preferences
  - Also called 'Preference Optimization'



Training flops