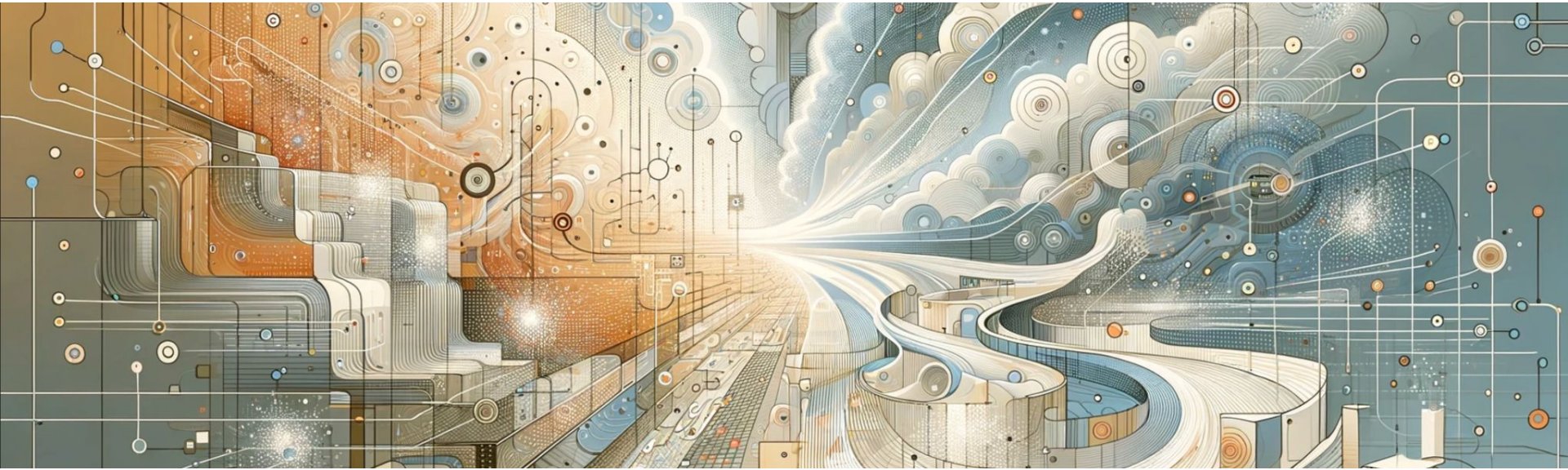
# CSE7850/CX4803 Machine Learning in Computational Biology



## Lecture 10: Large Language Models (LLMs)

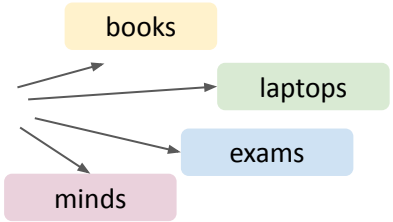
### --- with applications to protein sequences

Yunan Luo

# Language modeling in natural language

- **Language modeling** is the task of predicting what word comes next

*"The students opened their _____"*




- More formally: given a sequence of words $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$, compute the probability distribution of the next word $\boldsymbol{x}^{(t+1)}$:

$$P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{|V|}\}$.

- A system that does this is called a **language model (LM)**

Adapted from CS224N, Stanford

# Language models estimate the probability of text

Formally, let $y_1, y_2, \ldots, y_n$ be tokens in a sentence, and $P(y_1, y_2, \ldots, y_n)$ the probability to see all these tokens (in this order). Using the product rule of probability (aka the chain rule), we get

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

This is what an LM provides

P(I) =

P(I)

Probability of I

https://lena-voita.github.io/resources/lectures/lang_models/general/i_saw_a_cat_prob.gif

# N-gram language model

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

- We need to: define how to compute the conditional probabilities $P(y_t|y_{<t})$
- How: estimate based on global statistics from a text corpora, i.e., count.

$$P(y_t|y_1, \ldots, y_{t-1}) = \frac{N(y_1, \ldots, y_{t-1}, y_t)}{N(y_1, \ldots, y_{t-1})}$$

Example: $\text{P(mat | cat on a)} = \frac{\text{N(cat on a mat)}}{\text{N(cat on a)}}$

Formally, n-gram models assume that

$$P(y_t|y_1, \ldots, y_{t-1}) = P(y_t|y_{t-n+1}, \ldots, y_{t-1}).$$

For example,

- n=3 (trigram model): $P(y_t|y_1, \ldots, y_{t-1}) = P(y_t|y_{t-2}, y_{t-1})$,
- n=2 (bigram model): $P(y_t|y_1, \ldots, y_{t-1}) = P(y_t|y_{t-1})$,
- n=1 (unigram model): $P(y_t|y_1, \ldots, y_{t-1}) = P(y_t)$.

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

# N-gram language model

Before

P(I saw a cat on a mat) =

P(I)
· P(saw | I)
· P(a | I saw)
· P(cat | I saw a)
· P(on | I saw a cat)
· P(a | I saw a cat on)
· P(mat | I saw a cat on a)

After (3-gram)

P(I saw a cat on a mat) =

P(I) → P(I)
· P(saw | I) → · P(saw | I)
· P(a | I saw) → · P(a | I saw)
· P(cat | ~~I~~ saw a) → · P(cat | saw a)
· P(on | ~~I saw~~ a cat) → · P(on | a cat)
· P(a | ~~I saw a~~ cat on) → · P(a | cat on)
· P(mat | ~~I saw a cat~~ on a) → · P(mat | on a)

ignore use

I ____

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

# Generate a Text Using an N-gram Language Model

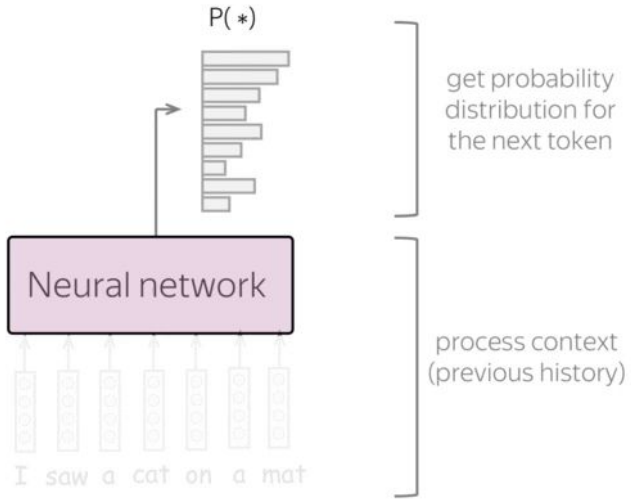
- Once we have a language model, we can use it to generate text. We do it one token at a time: predict the probability distribution of the next token given previous context, and **sample** from this distribution.


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

# Neural language model

$$P(y_1, y_2, \ldots, y_n) = P(y_1) \cdot P(y_2|y_1) \cdot P(y_3|y_1, y_2) \cdot \cdots \cdot P(y_n|y_1, \ldots, y_{n-1}) = \prod_{t=1}^{n} P(y_t|y_{<t})$$

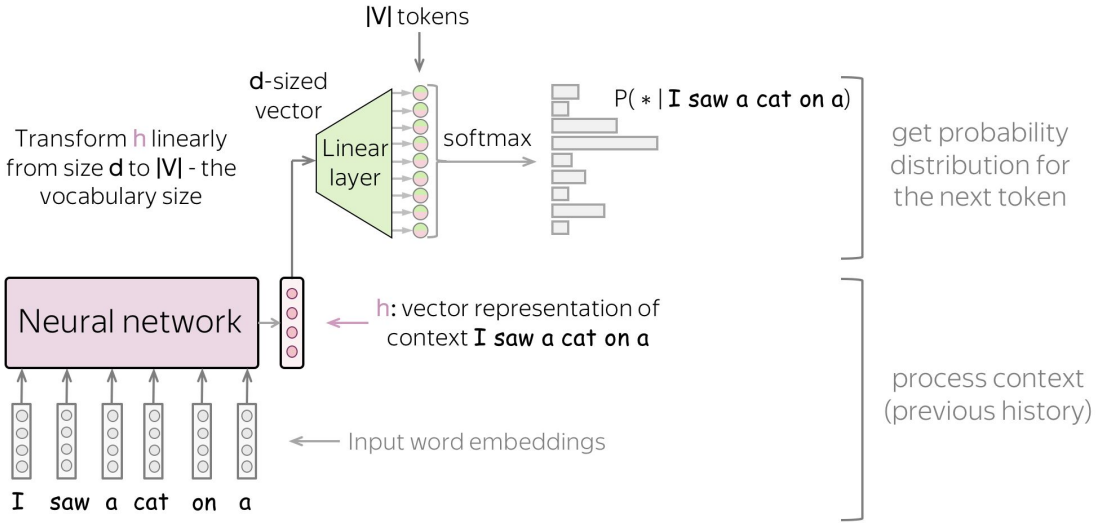
- We need to: define how to compute the conditional probabilities $P(y_t|y_{<t})$
- How: Train a neural network to predict them




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

# Neural network of an LM

- Left-to-right neural language models can be thought of as classifiers
  - 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.




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

# Training an LLM

Neural LMs are trained to predict a probability distributions of the next token given the previous context. Intuitively, at each step we maximize the probability a model assigns to the correct token.

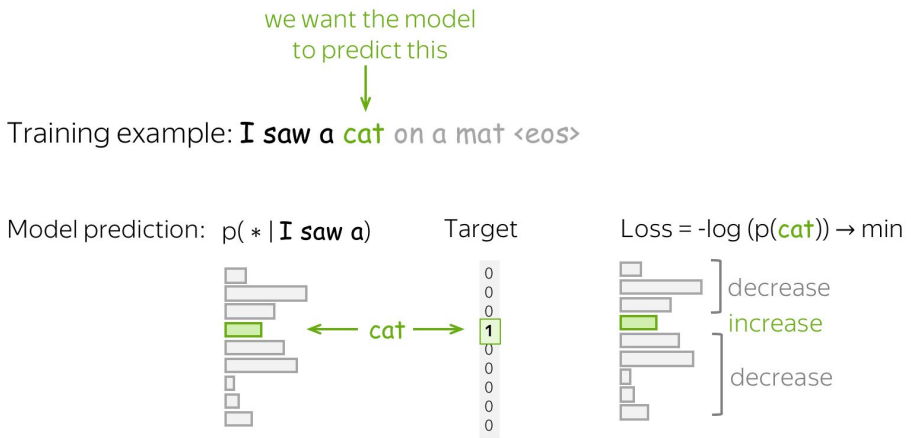
Formally, if $y_1, \ldots, y_n$ is a training token sequence, then at the timestep $t$ a model predicts a probability distribution $p^{(t)} = p(*|y_1, \ldots, y_{t-1})$. The target at this step is $p^* = \textbf{one-hot}(y_t)$, i.e., we want a model to assign probability 1 to the correct token, $y_t$, and zero to the rest.

The standard loss function is the **cross-entropy** loss. Cross-entropy loss for the target distribution ***p**** and the predicted distribution is ***p**** 

$$Loss(p^*, p) = -p^* \log(p) = -\sum_{i=1}^{|V|} p_i^* \log(p_i).$$

Or equivalently

$$Loss(p^*, p) = -\log(p_{y_t}) = -\log(p(y_t|y_{<t})).$$



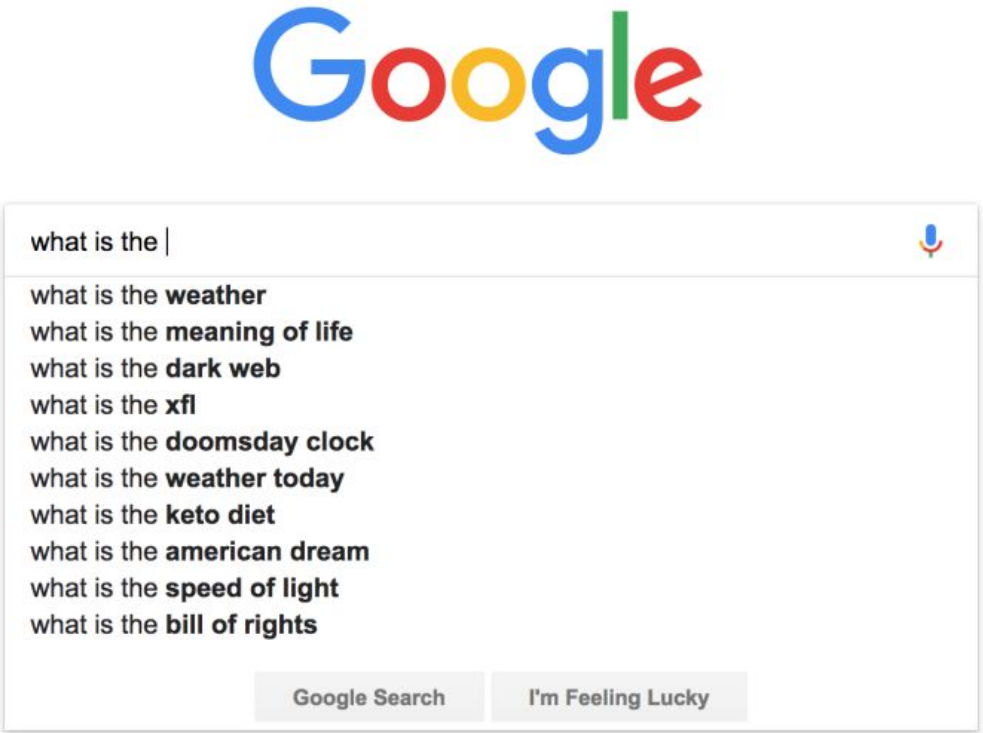

https://lena-voita.github.io/nlp_course/language_modeling.html

# Generate a Text Using a Neural Language Model

- As we saw before, to generate a text using a language model you just need to **sample** tokens from the probability distribution predicted by a model.


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

# You use LM every day!

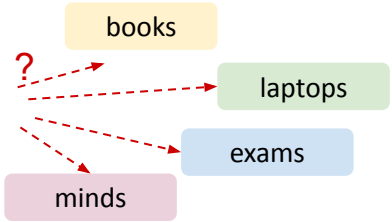


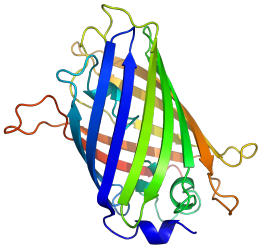

Adapted from CS224N, Stanford

# Protein Language Models

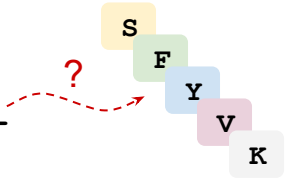
# Learning the language of proteins

*The students opened their _____*



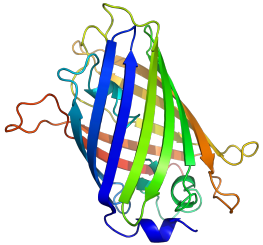
**MSKGEELFTGVVPILVELDGDVNGHKFSV_**



# Protein language models

Autoregressive language models

$$p(x) = \prod_{i=1}^{L} p(x_i | x_1 \dots x_{i-1})$$

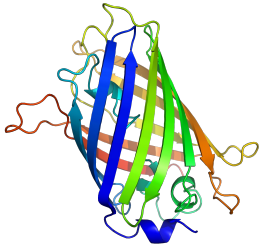


MSKGEELFTGVVPILVELDGDVNGHKFSV_ ? S F Y V K

---

Masked language models

$$p(x) = \prod_{i=1}^{L} p(x_i | x_1 \dots x_{i-1}, x_{i+1} \dots x_L)$$



S F Y V K

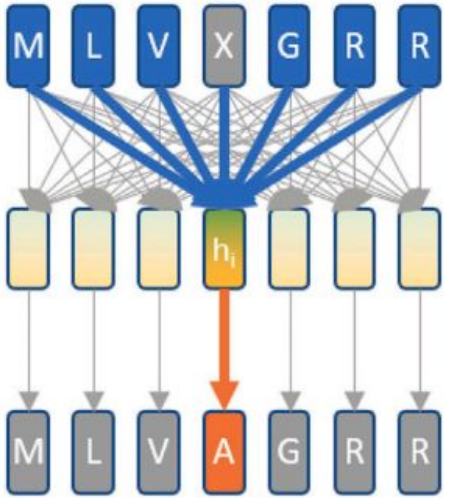
? MSKGEELFTGVVPI_VELDGDVNGHKFSVY

Can also mask multiple AAs at a time: MSKGEE??TGVVPI????DGDVNGHKFSVY

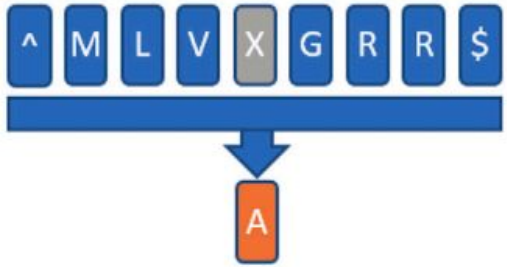
# How to train a protein LM?

- Given a protein sequence $x = (x_1, x_2, \ldots, x_L)$
  - Mask a token $x_i$ randomly (or multiple tokens)
  - Use other tokens as input
  - Train the neural network to predict $x_i$
  - Objective: $\mathbb{E}_{x\sim X}\mathbb{E}_M \sum_{i\in M} -\log p(x_i|x_{/M})$
- This training strategy is called **self-supervised learning**
  - Given only data (X), no labels (Y)
  - Simulate labels from the X itself
  - The model trains itself to learn one part of the input from another part of the input

$$p(x) = \prod_{i=1}^{L} p(x_i|x_1 \ldots x_{i-1}, x_{i+1} \ldots x_L)$$




Processes whole sequence




$$p(x_i = A|x_1 \ldots x_{i-1}, x_{i+1} \ldots x_L)$$

Bepler and Berger, *Cell Systems*, 2021

# Demo: Protein Language Model

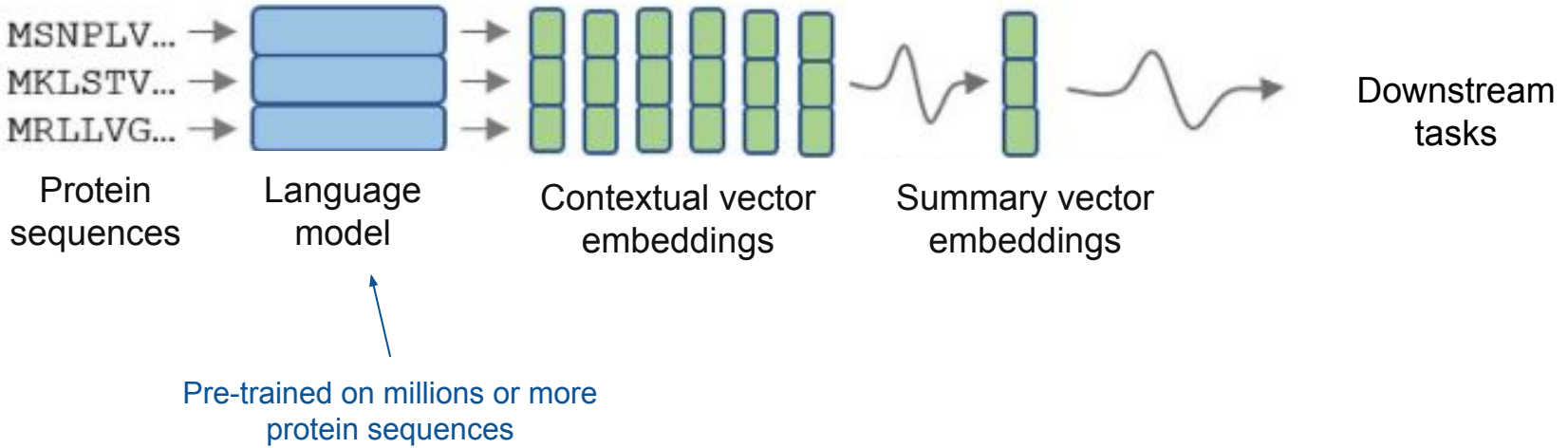
Google Colab

# Exercise

Following the idea in the Colab Notebook, use LM to analyze the mutations in SARS-CoV-2 Spike protein

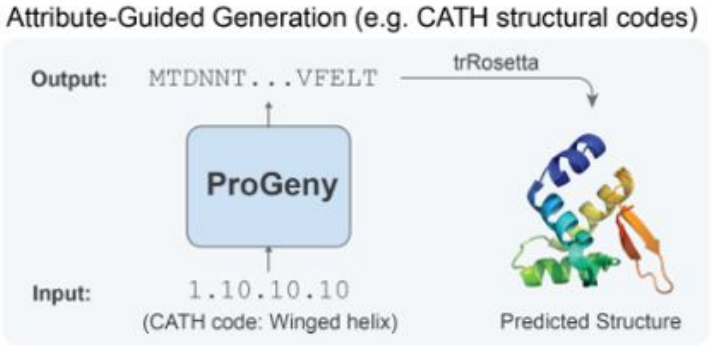
- Get real mutations in Delta, Omicron variants
- Did those variants mutate amino acids to those with very low probability (predicted by LM)?
- Investigate other interesting questions you may have

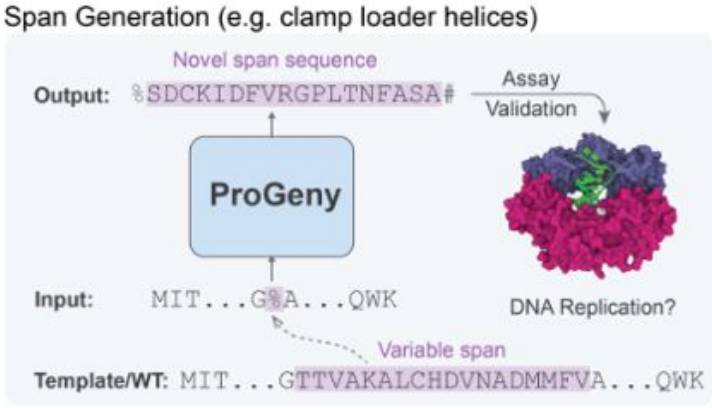
# Protein language models




Bepler and Berger, *Cell Systems*, 2021

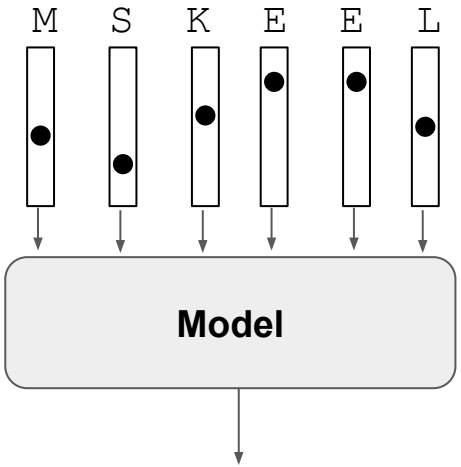
# Application of protein language models

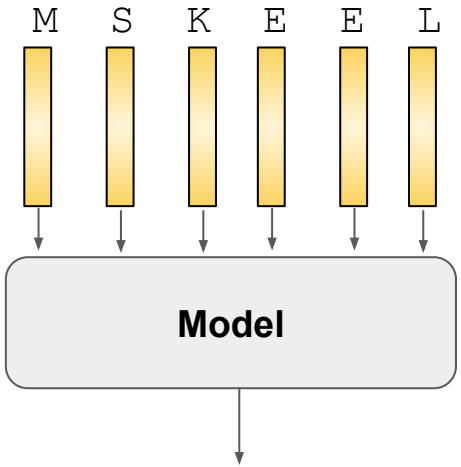
# Application 1: Regression/Classification

**Conventional approach: One-hot encoding**

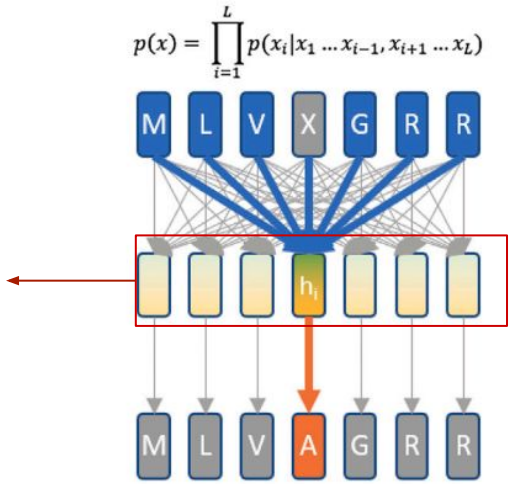



Regression target / Class label

**Using LM representations to replace one-hot**



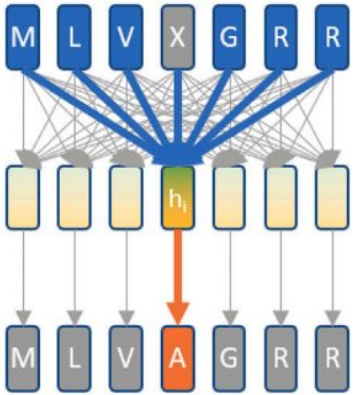

Regression target / Class label



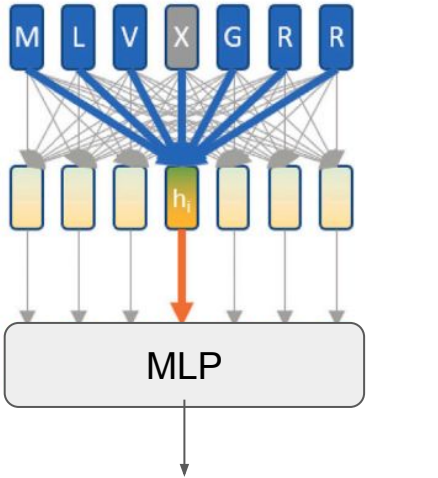
# Discussion: are LM representations better than one-hot?

# Fine-tune the LM for a target task

Pre-training




Fine-tuning

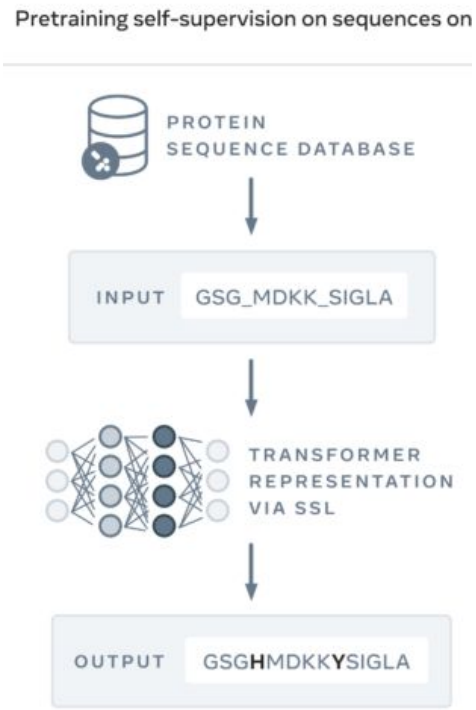



Regression target / Class label

Meta

Meta Research

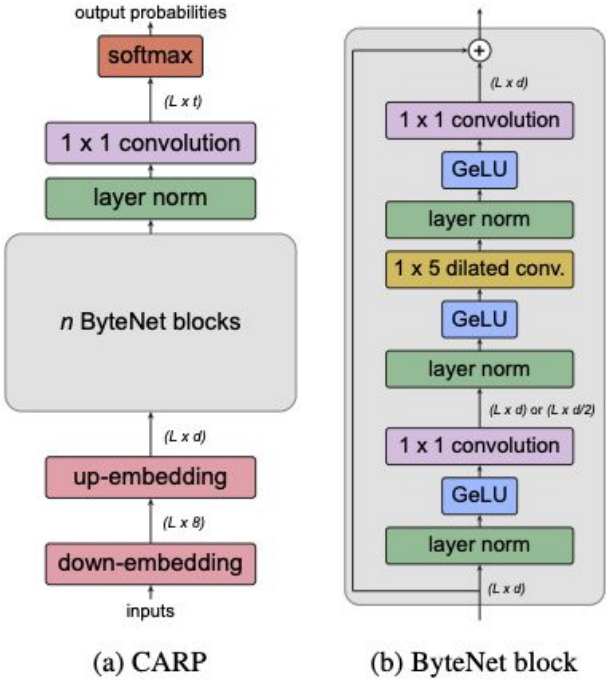
# PLM examples: ESM




ESM (evolutionary scaling model):

- “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”
- https://www.pnas.org/doi/10.1073/pnas.2016239118

# PLM examples: CARP




CARP (Convolutional Autoencoding Representations of Proteins):

- “Convolutions are competitive with transformers for protein sequence pretraining”
- https://www.biorxiv.org/content/biorxiv/early/2022/11/21/2022.05.19.492714.full.pdf

# PLM examples: ProGen

nature biotechnology

Article

https://doi.org/10.1038/s41587-022-01618-2

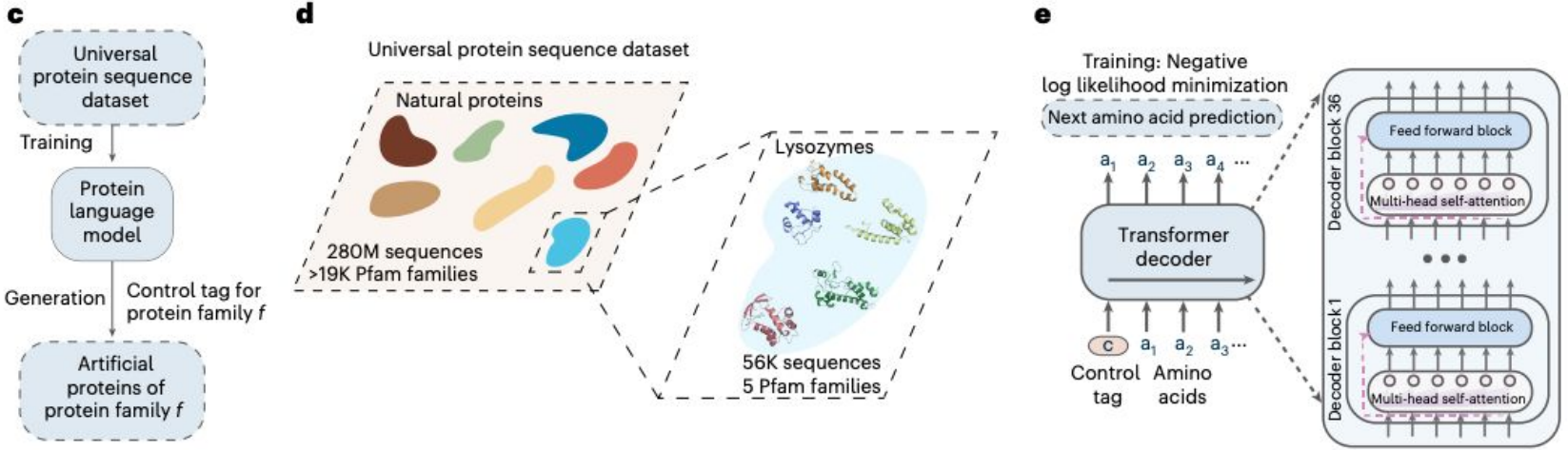
## Large language models generate functional protein sequences across diverse families

Received: 12 July 2022

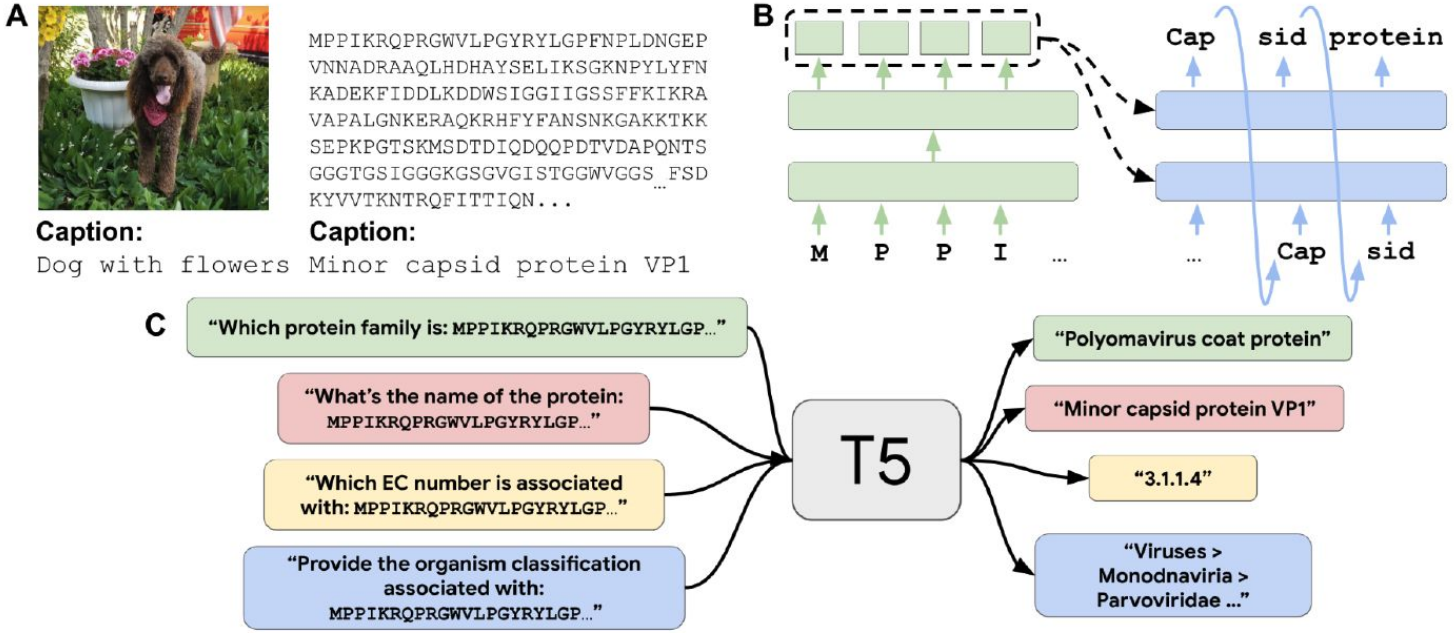
Accepted: 17 November 2022

Published online: 26 January 2023

Ali Madani[1,2], Ben Krause[1,10], Eric R. Greene[3,10], Subu Subramanian[4,5], Benjamin P. Mohr[6], James M. Holton[7,8,9], Jose Luis Olmos Jr.[3], Caiming Xiong[1], Zachary Z. Sun[6], Richard Socher[1], James S. Fraser[3] & Nikhil Naik[1]



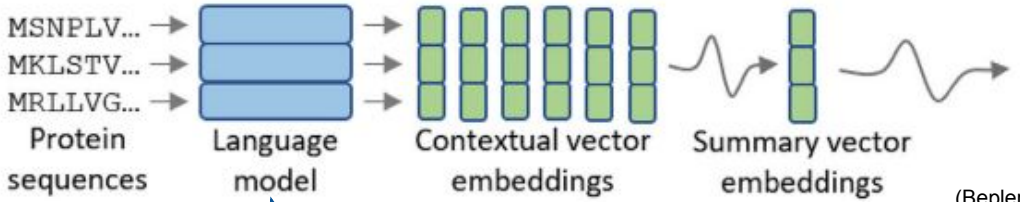
# PLM examples: ProtNLM




ProtNLM (Protein Natural Language Model) :

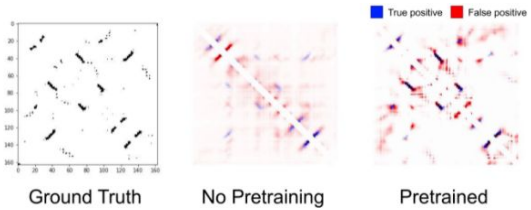
- "ProtNLM: Model-based Natural Language Protein Annotation"
- https://www.uniprot.org/help/ProtNLM

# Application: improving downstream prediction tasks

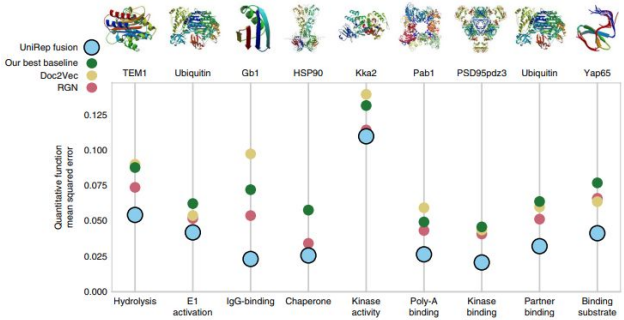



(Bepler and Berger, *Cell Systems*, 2021)

## Protein contact prediction
(Rao et al., NeurIPS, 2019)



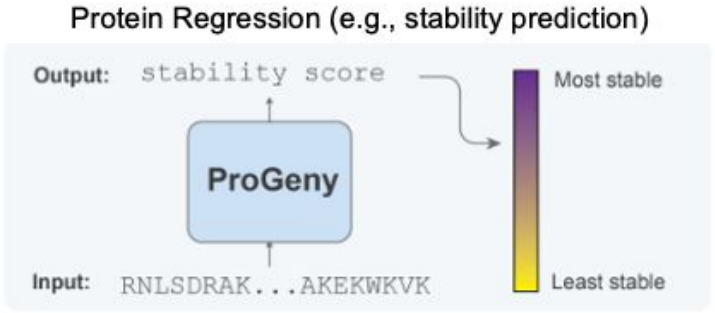

## Protein function prediction
(Alley et al., Nature Methods, 2019)

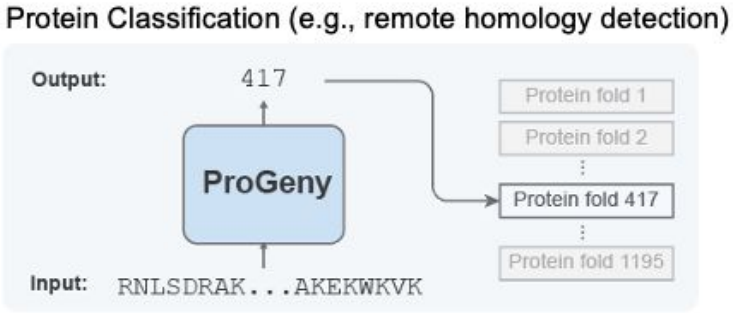


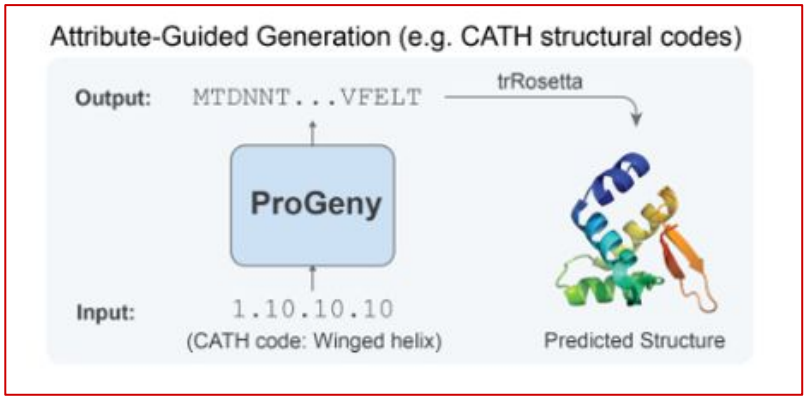

## Remote homology detection
(Rives et al., PNAS, 2021)

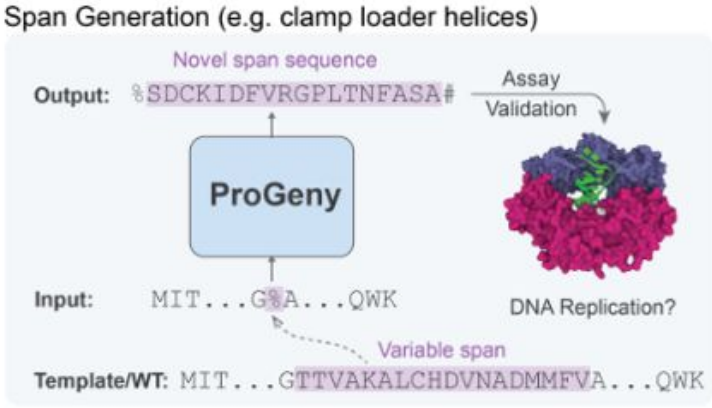
| | | Hit-10 | | AUC | |
|---|---|---|---|---|---|
| | Pretraining | Fold | SF | Fold | SF |
| HHblits* | | 0.584 | 0.965 | 0.831 | 0.951 |
| LSTM (S) | UR50/S | 0.558 | 0.760 | 0.801 | 0.863 |
| LSTM (L) | UR50/S | 0.574 | 0.813 | 0.805 | 0.880 |
| Transformer-6 | UR50/S | 0.653 | 0.878 | 0.768 | 0.901 |
| Transformer-12 | UR50/S | 0.639 | 0.915 | 0.778 | 0.942 |
| Transformer-34 | (None) | 0.481 | 0.527 | 0.755 | 0.807 |
| Transformer-34 | UR100 | 0.599 | 0.841 | 0.753 | 0.876 |
| Transformer-34 | UR50/D | 0.617 | 0.932 | 0.822 | 0.932 |
| Transformer-34 | UR50/S | 0.639 | 0.931 | 0.825 | 0.933 |
| ESM-1b | UR50/S | 0.532 | 0.913 | 0.770 | 0.880 |

# Application of protein language models

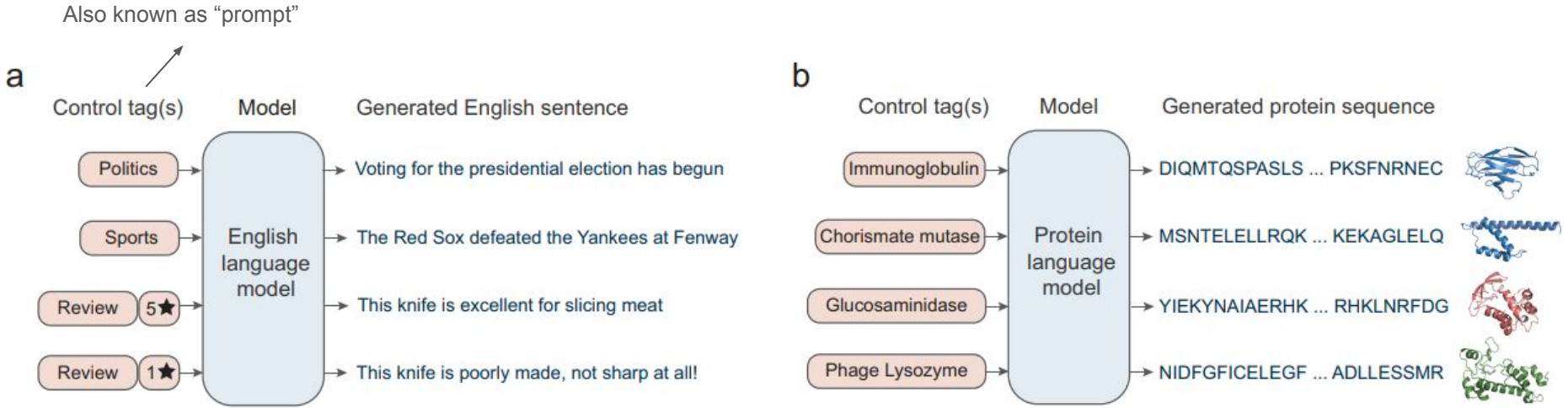


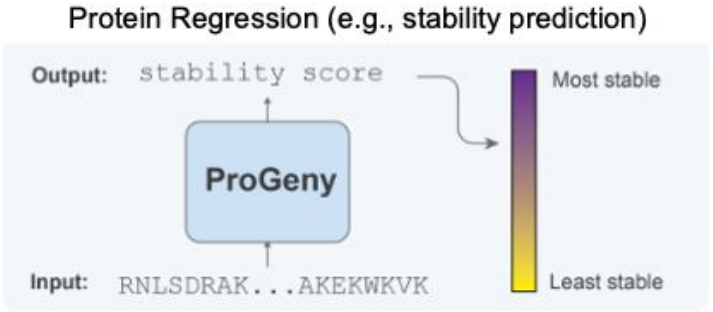



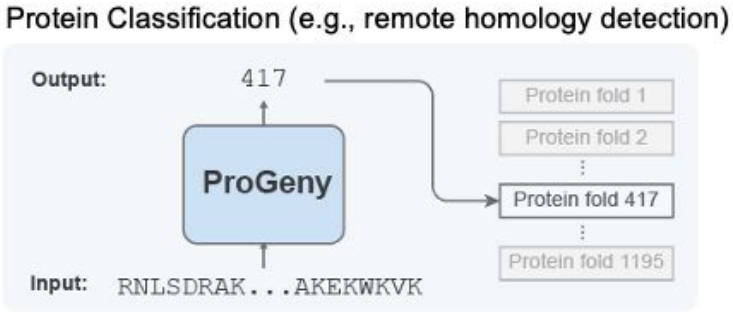
# Application: generating novel functional protein sequences

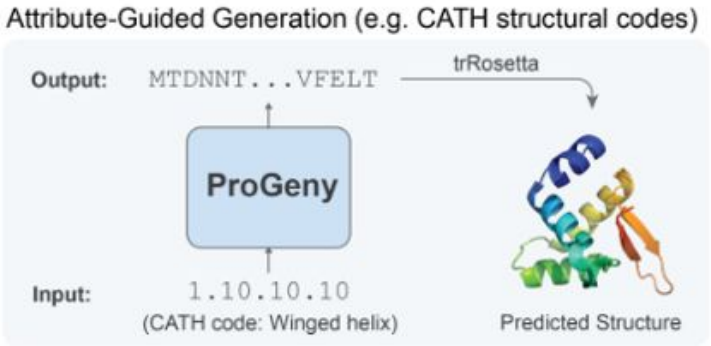


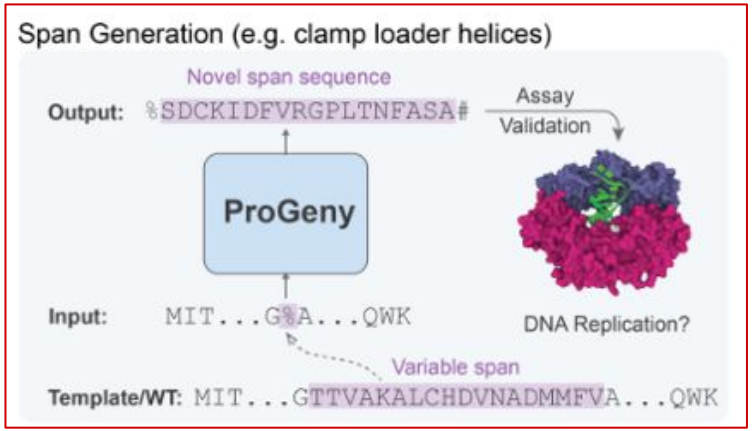

ProGen: Controllable generation of protein sequences
(Madani et al., Nature Biotech, 2023)

# Application of protein language models

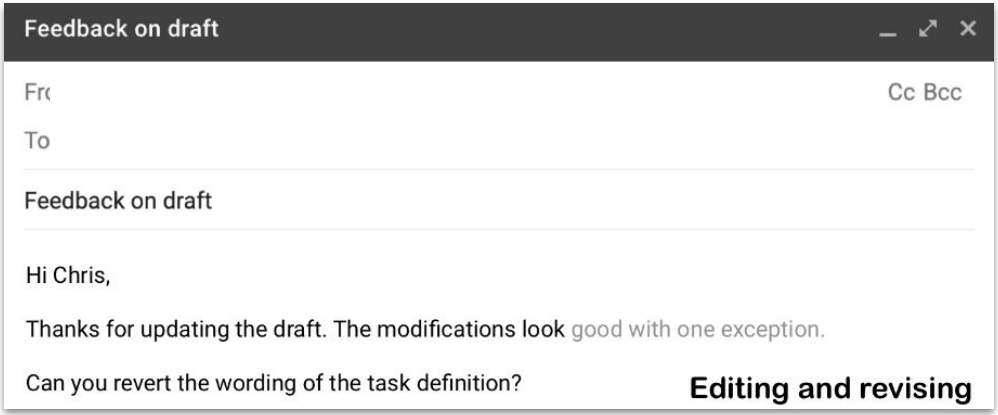


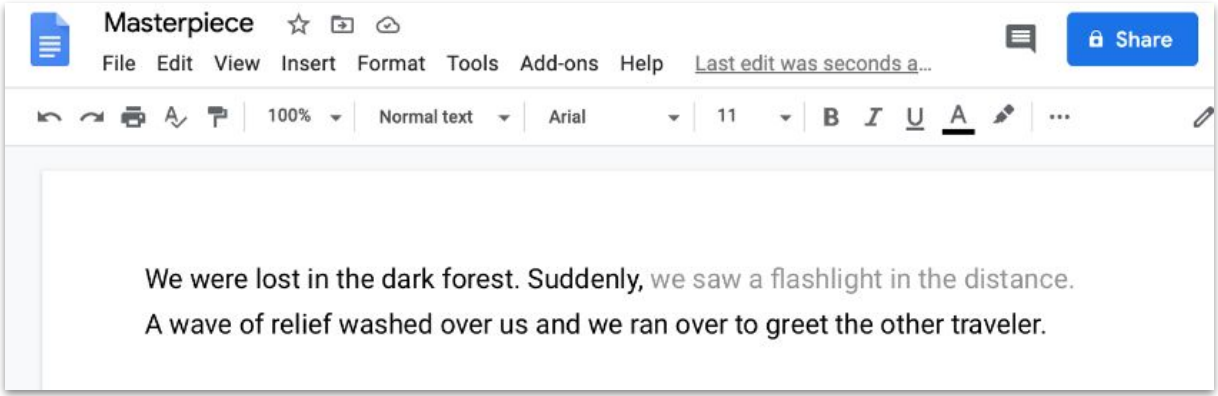



# Fill in the Blanks using LMs



# Fill in the Blanks using LMs

**Fill in the blanks?**

Consider the following sentence with blanks:

> *She ate ___ for ___*

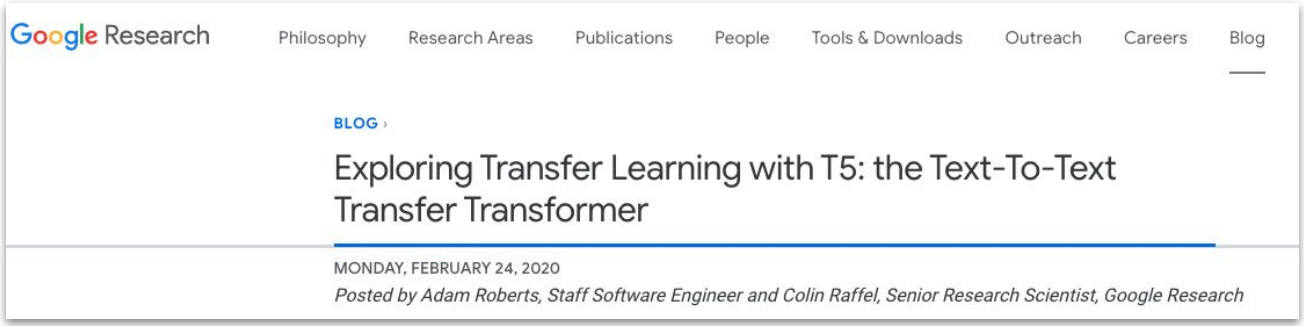
To fill in the blanks, one needs to consider both preceding and subsequent text (in this case, *"She ate"* and *"for"*). There can be many reasonable ways to fill in the blanks:

> *She ate **leftover pasta** for **lunch***
>
> *She ate **chocolate ice cream** for **dessert***
>
> *She ate **toast for breakfast before leaving** for **school***
>
> *She ate **rather quickly** for **she was in a hurry that evening***

The task of filling in the blanks is known as *text infilling* in the field of Natural Language Processing (NLP). It is the task of predicting blanks (or missing spans) of text at any position in text.

- This is exactly what LM does!
  - The model trains itself to learn one part of the input from another part of the input

# Fill in the Blanks using LMs



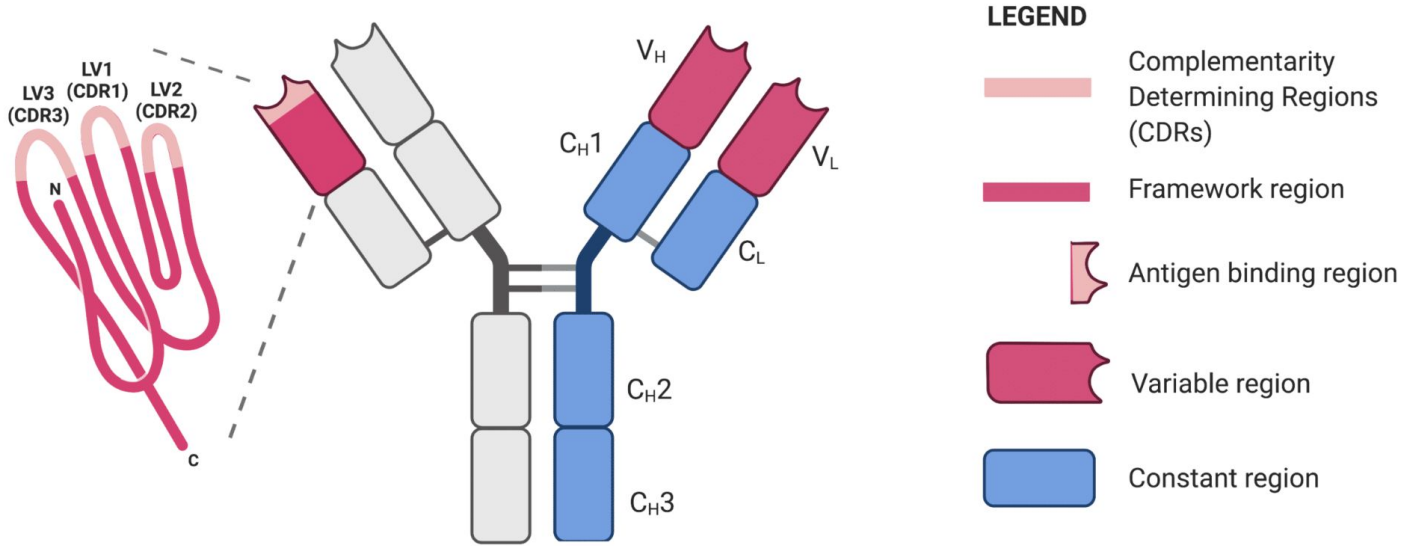

**Input**: I love peanut butter and __ sandwiches

| N=1 | N=2 | N=4 | N=8 | N=16 | N=32 | N=64 | N=512 |
|---|---|---|---|---|---|---|---|

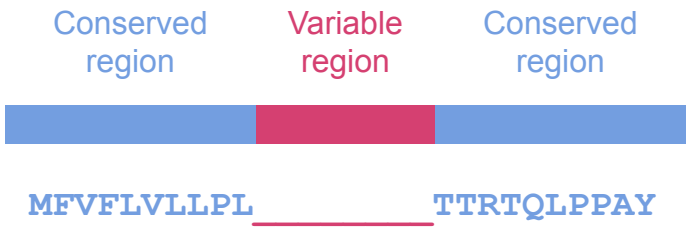
I love peanut butter and *jelly* sandwiches.

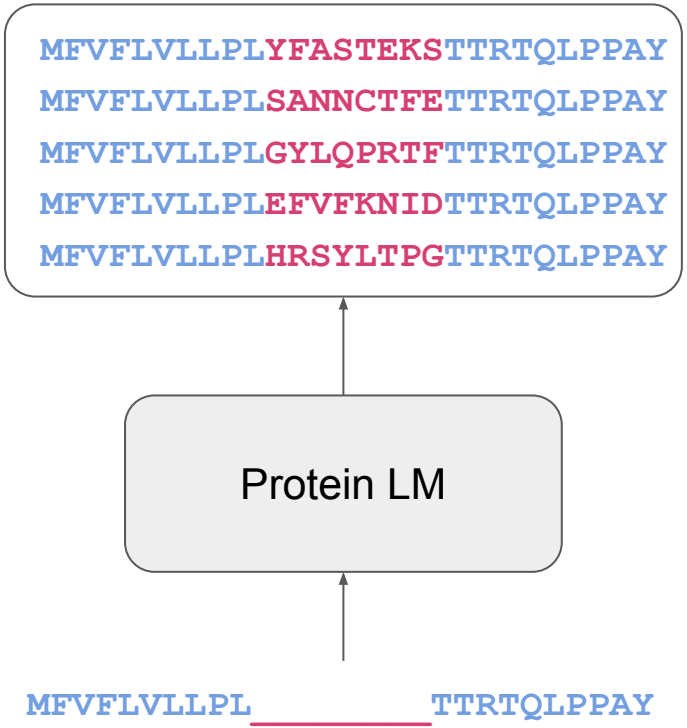
# Fill in Blanks in Protein Sequences!

Application: antibody design



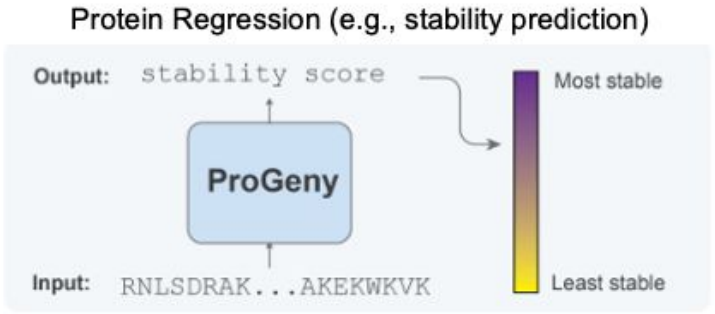
# Application: Antibody Design

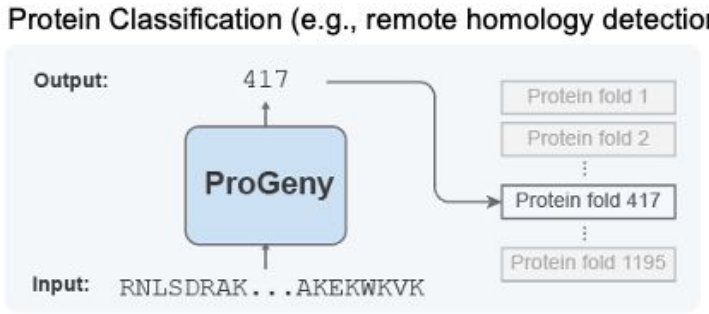


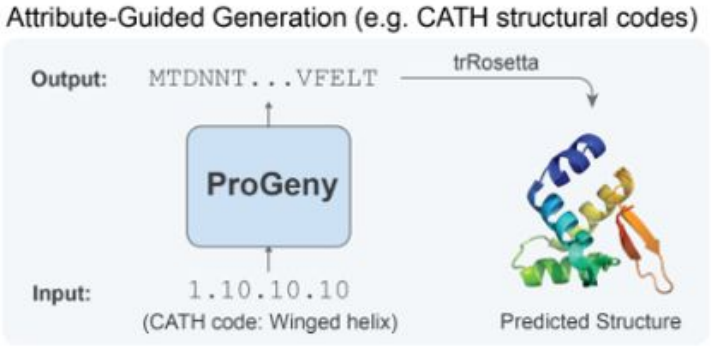



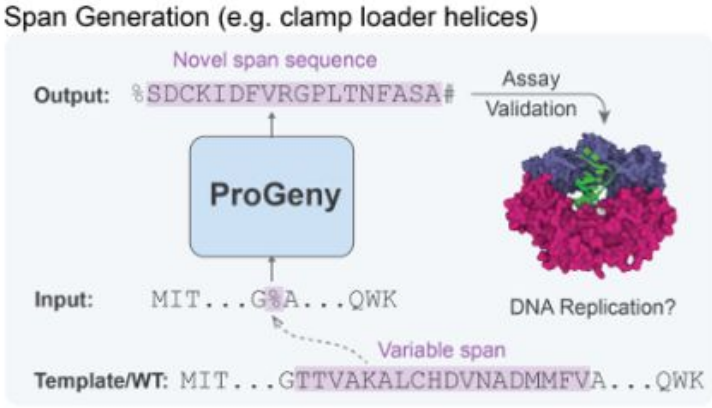

Sequences on this slides are not real antibody sequences

# Application of protein language models



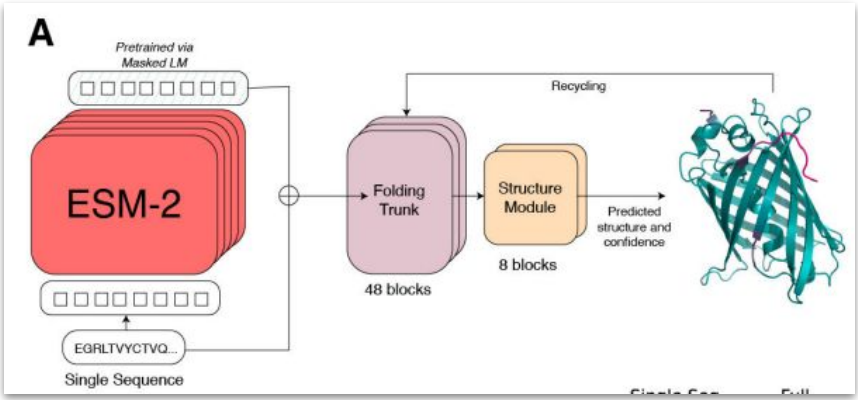



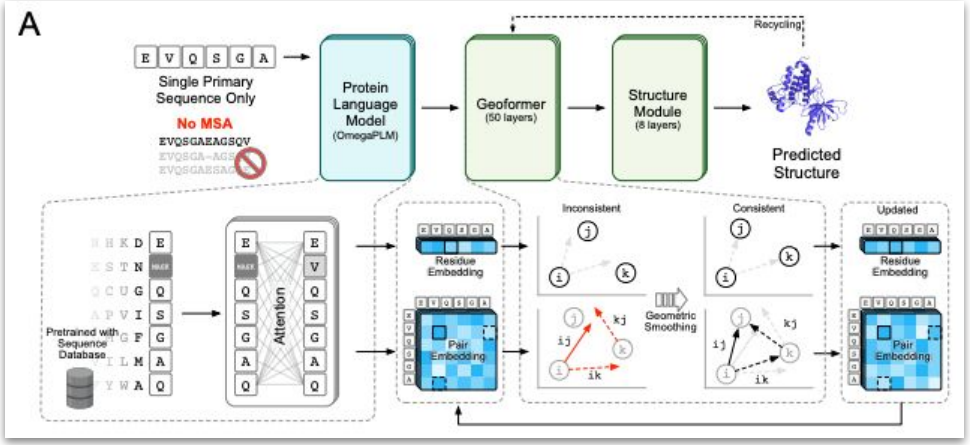



# Research papers on PLMs

# Use LM to assist protein structure prediction



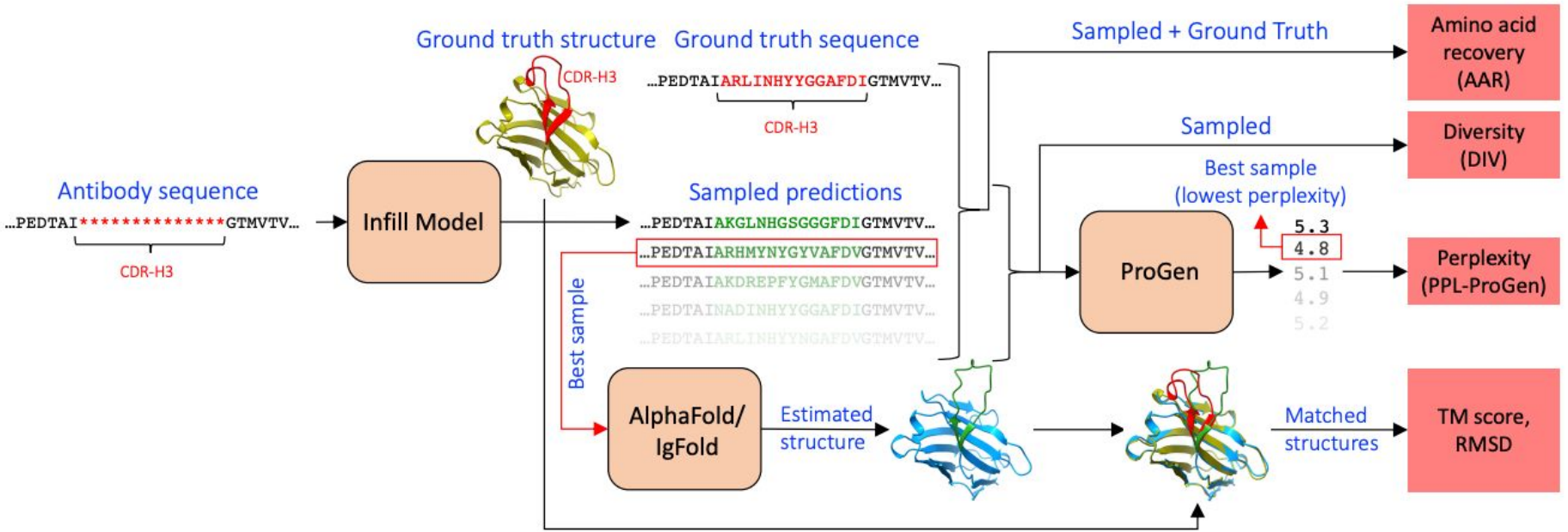

ESM2:

- “Language models of protein sequences at the scale of evolution enable accurate structure prediction”
- https://www.biorxiv.org/content/10.1101/2022.07.20.500902v1.full.pdf

OmegaFold:

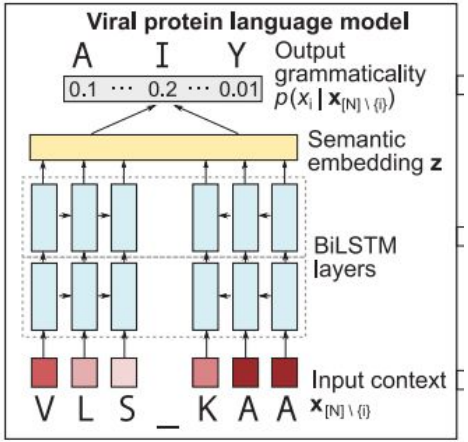
- “High-resolution de novo structure prediction from primary sequence”
- https://www.biorxiv.org/content/10.1101/2022.07.21.500999v1.full.pdf

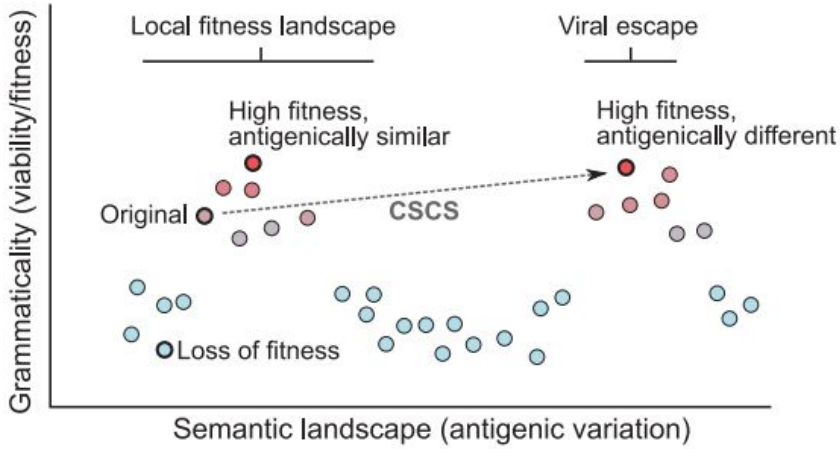
# Sequence & Structure Co-Design




Reprogramming Large Pretrained Language Models for Antibody Sequence Infilling
https://arxiv.org/abs/2210.07144

# Unsupervised prediction of viral escape





Science

Current Issue First release papers Archive About Submit manu

HOME > SCIENCE > VOL. 371, NO. 6526 > LEARNING THE LANGUAGE OF VIRAL EVOLUTION AND ESCAPE

REPORT

**Learning the language of viral evolution and escape**

BRIAN HIE, ELLEN D. ZHONG, BONNIE BERGER, AND BRYAN BRYSON Authors Info & Affiliations

*SCIENCE* · 15 Jan 2021 · Vol 371, Issue 6526 · pp. 284-288 · DOI: 10.1126/science.abd7331

LM predicts viral escape
(Hie et al., Science, 2021)

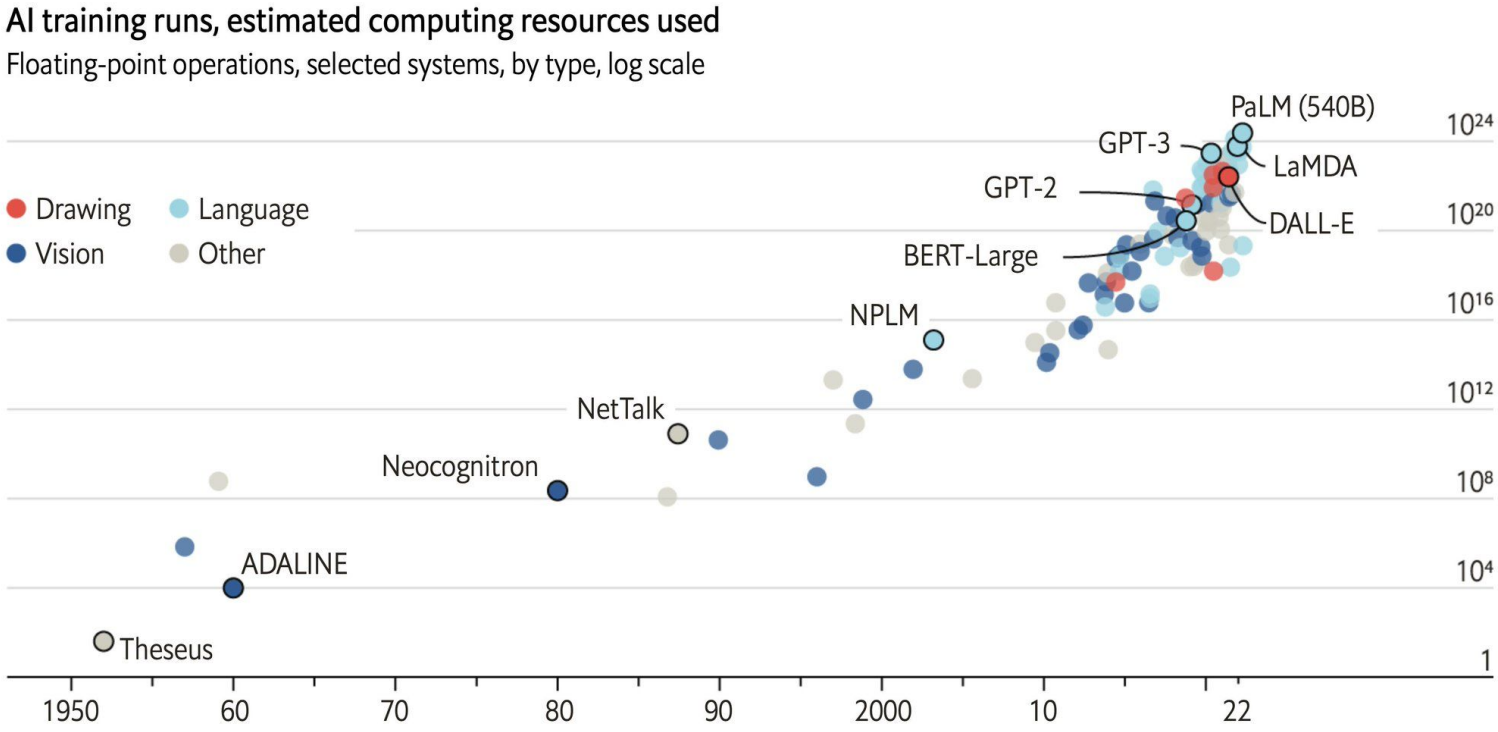
# Project ideas

- Use LM representations to predict properties of proteins
  - See tasks in https://arxiv.org/abs/1906.08230
    - Stability (regression)
    - Fitness (regression)
    - Remote homology (classification)
    - Secondary structure (classification)
  - Structure classification (database)
  - Protein fitness prediction (database)

- Fine-tune LM to generate novel protein sequences
  - Generating proteins with a particular function (example)
  - Antibody design (example)
  - Those tasks are more challenging!

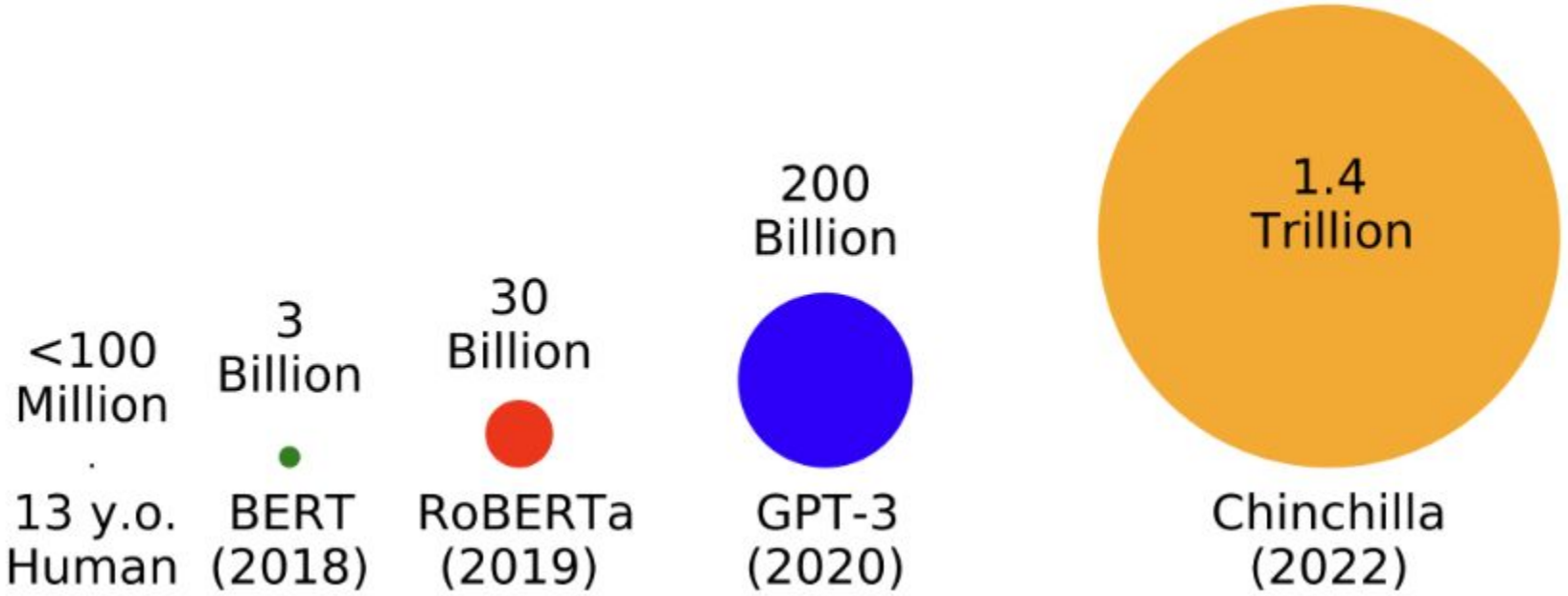
# Bonus slides:
# From language modeling to ChatGPT

# Larger and larger LMs




Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

# Trained on more and more data




https://babylm.github.io/

# What kinds of things does pretraining learn?

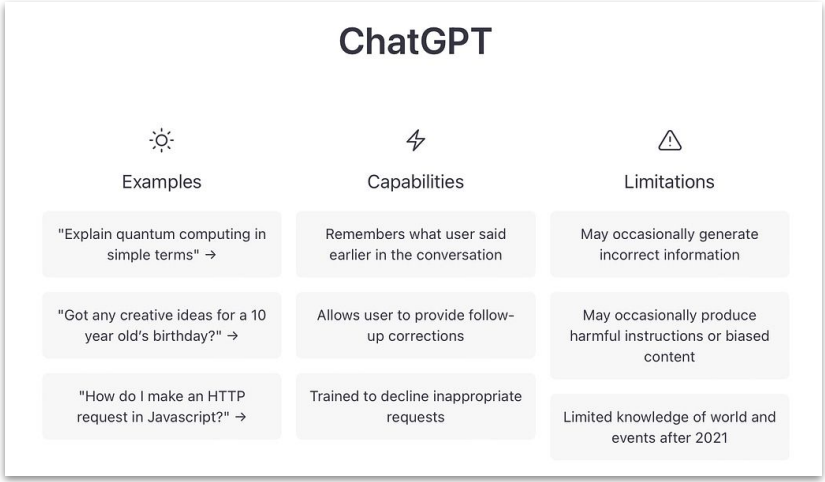
- Georgia Institute of Technology is located in _____, Georgia [Trivia]
- I put ___ fork down on the table. [syntax]
- The woman walked across the street, checking for traffic over ___ shoulder. [coreference]
- I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was ___. [sentiment]
- Iroh went into the kitchen to make some tea. Standing next to Iroh, Zuko pondered his destiny. Zuko left the _____. [some reasoning – this is harder]
- I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ____ [some basic arithmetic; they don't learn the Fibonnaci sequence]

# Language models as multitask assistants?

How do we get from this

Georgia Institute of Technology is located in _____, Georgia.

to this?



# Outline

- From language modeling to ChatGPT
  - **Instruction finetuning**
  - Alignment

# Language modeling ≠ assisting users

Prompt *Explain the moon landing to a 6 year old in a few sentences.*

Completion GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

Language models are *not aligned* with user intent. InstructGPT (2022)

https://openai.com/research/instruction-following

# Language modeling ≠ assisting users

Prompt

*Explain the moon landing to a 6 year old in a few sentences.*

Completion

**Human**

*A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.*

Language models are *not aligned* with user intent. InstructGPT (2022)

Finetuning to the rescue!

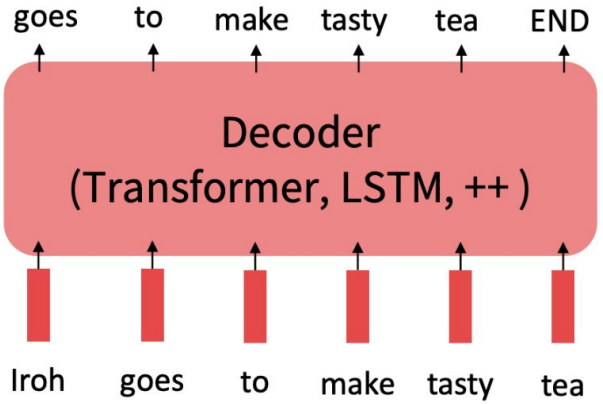
https://openai.com/research/instruction-following

# Recap: The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

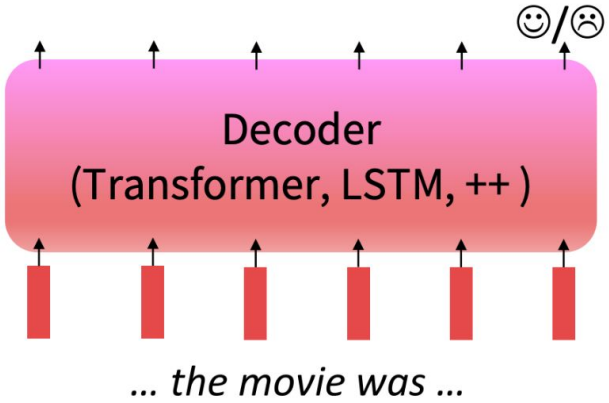
**Step 1: Pretrain (on language modeling)**

Lots of text; learn general things!




**Step 2: Finetune (on your task)**

Not many labels; adapt to the task!

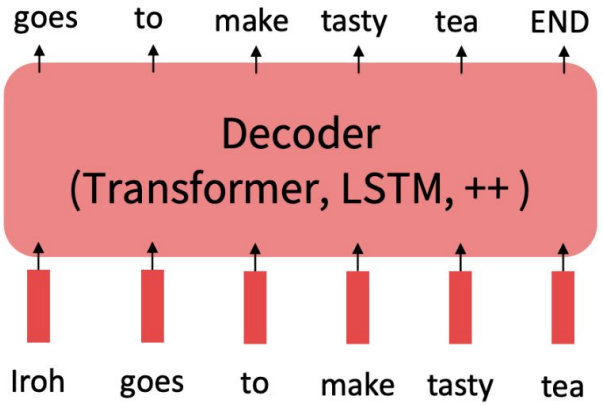


# Scaling up finetuning

Pretraining can improve NLP applications by serving as parameter initialization.

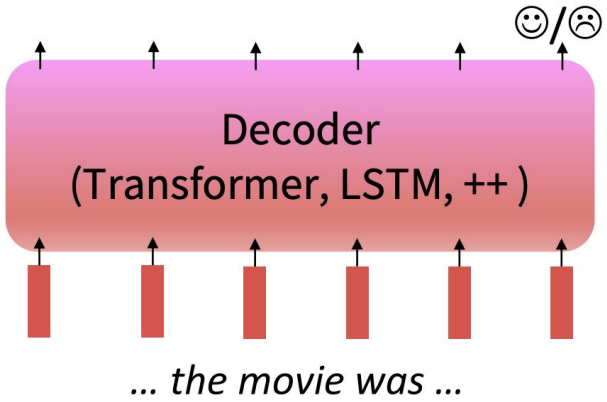
**Step 1: Pretrain (on language modeling)**

Lots of text; learn general things!



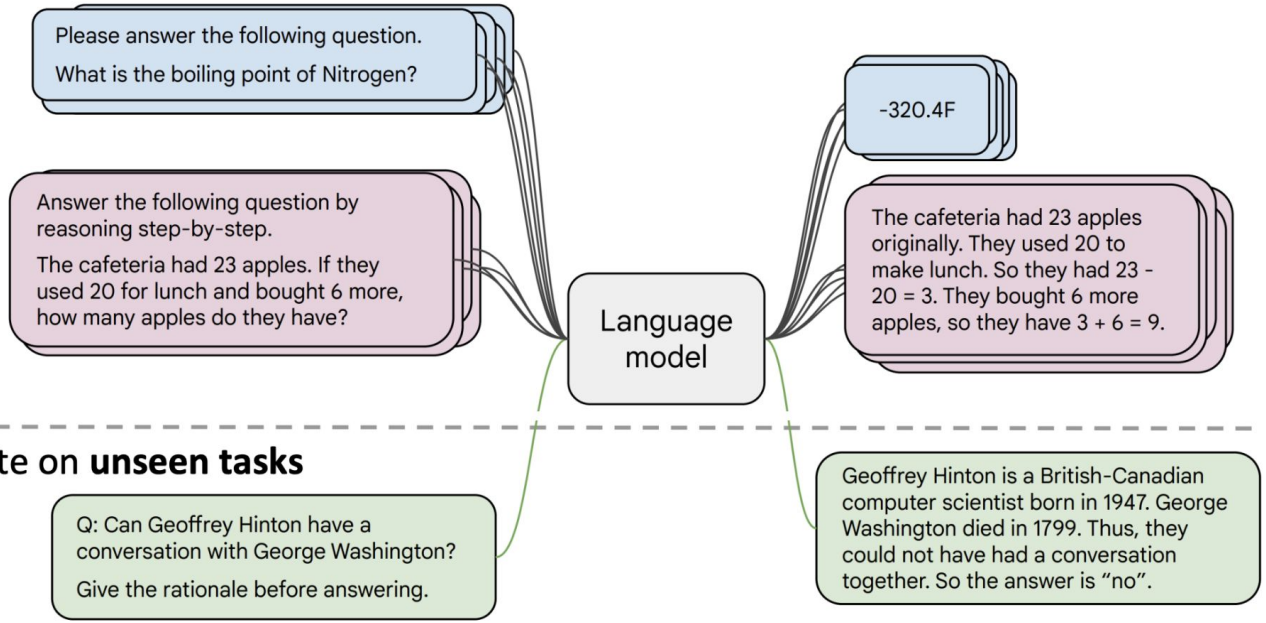

**Step 2: Finetune (on many tasks)**

~~Not~~ many labels; adapt to the tasks!



# Instruction finetuning

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM




As is usually the case, **data + model scale** is key for this to work!

[FLAN-T5; Chung et al., 2022]

# Instruction finetuning

**Model input (Disambiguation QA)**

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

**Before instruction finetuning**

The reporter and the chef will discuss their favorite dishes.
The reporter and the chef will discuss the reporter's favorite dishes.
The reporter and the chef will discuss the chef's favorite dishes.
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✖ **(doesn't answer question)**

Try FLAN-T5 at: https://huggingface.co/google/flan-t5-xxl

[FLAN-T5; Chung et al., 2022]

# Instruction finetuning

**Model input (Disambiguation QA)**

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

**After instruction finetuning**

The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C). ✅

Try FLAN-T5 at: https://huggingface.co/google/flan-t5-xxl

[FLAN-T5; Chung et al., 2022]

# Limitations of instruction finetuning?

One limitation of instruction finetuning is obvious: it's **expensive** to collect groundtruth data for tasks. But there are other, subtler limitations too.

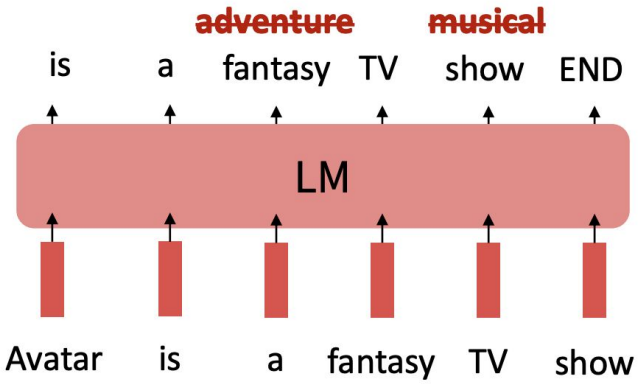
***Problem 1***: tasks like open-ended creative generation have no right answer.

- "Write me a story about a dog and her pet grasshopper"

***Problem 2***: language modeling penalizes all token-level mistakes equally, but some errors are worse than others.

Even with instruction finetuning, there a mismatch between the LM objective and the objective of "satisfy human preferences"!

Can we **explicitly attempt to satisfy human preferences**?



# Outline

- What is LLM?
- Model architectures & (Pre-)Training
- Fine-tuning
- From language modeling to ChatGPT
  - Instruction finetuning
  - **Alignment**

# Reinforcement Learning from Human Feedback (RLHF)

# Optimizing for human preferences

- Let's say we were training a language model on some task (e.g. summarization).
- For each LM sample $s$, imagine we had a way to obtain a *human reward* of that summary: $R(s) \in \mathbb{R}$, higher is better.

```
SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.
```

```
An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.
```

$$s_1$$
$$R(s_1) = 8.0$$

```
The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.
```

$$s_2$$
$$R(s_2) = 1.2$$

- Now we want to maximize the expected reward of samples from our LM:

$$\mathbb{E}_{\hat{s} \sim p_\theta(s)}[R(\hat{s})]$$

# Optimizing for human preferences

- How do we actually change our LM parameters $\theta$ to maximize this?

$$\mathbb{E}_{\hat{s}\sim p_\theta(s)}[R(\hat{s})]$$

- Let's try doing gradient ascent!

$$\theta_{t+1} := \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s}\sim p_{\theta_t}(s)}[R(\hat{s})]$$

How to take gradient w.r.t an expectation?

What if our reward function is nondifferentiable??

Reinforcement learning (RL) methods (e.g., REINFORCE; [Williams, 1992]) give us tools for estimating and optimizing this objective.

# A brief introduction to REINFORCE

- We want to obtain

(defn. of expectation) (linearity of gradient)

$$\nabla_\theta \mathbb{E}_{\hat{s}\sim p_\theta(s)}[R(\hat{s})] = \nabla_\theta \sum_s R(s) p_\theta(s) = \sum_s R(s) \nabla_\theta p_\theta(s)$$

- Here we'll use a very handy trick known as the **log-derivative trick**. Let's try taking the gradient of $\log p_\theta(s)$

$$\nabla_\theta \log p_\theta(s) = \frac{1}{p_\theta(s)} \nabla_\theta p_\theta(s) \quad \Rightarrow \quad \nabla_\theta p_\theta(s) = \nabla_\theta \log p_\theta(s)\, p_\theta(s)$$

(chain rule)

- Plug back in:

This is an expectation of this

$$\sum_s R(s) \nabla_\theta p_\theta(s) = \sum_s p_\theta(s) R(s) \nabla_\theta \log p_\theta(s)$$

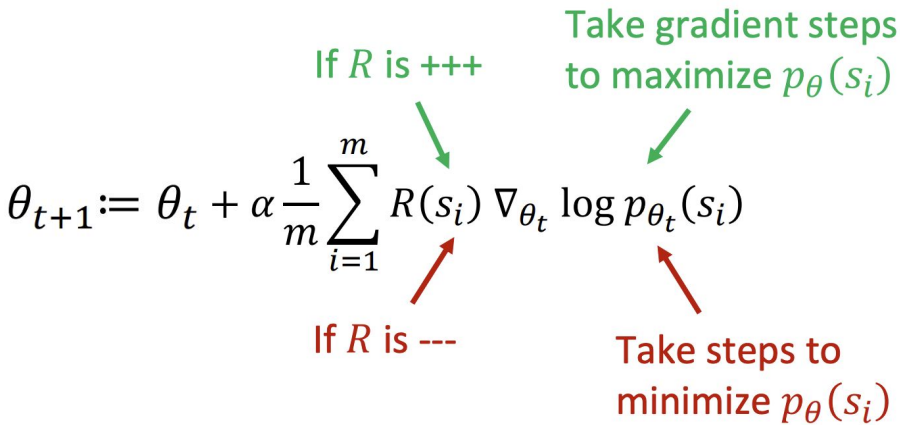
$$= \mathbb{E}_{\hat{s}\sim p_\theta(s)}[R(\hat{s}) \nabla_\theta \log p_\theta(\hat{s})]$$

# A brief introduction to REINFORCE

- Now we have put the gradient “inside” the expectation, we can approximate this objective with Monte Carlo samples:

$$\nabla_\theta \mathbb{E}_{\hat{s}\sim p_\theta(s)}[R(\hat{s})] = \mathbb{E}_{\hat{s}\sim p_\theta(s)}[R(\hat{s})\,\nabla_\theta \log p_\theta(\hat{s})] \approx \frac{1}{m}\sum_{i=1}^{m} R(s_i)\,\nabla_\theta \log p_\theta(s_i)$$

- Giving us the update rule:




(This is why it’s called “reinforcement learning”: we reinforce good actions, increasing the chance they happen again.)

# How do we model human preferences?

Now for any arbitrary, non-differentiable reward function $R(s)$ , we can train our language model to maximize expected reward.

- Not so fast! (Why not?)
- **Problem 1**: human-in-the-loop is expensive!
  - Solution: instead of directly asking humans for preferences, model their preferences as a separate problem! [Knox and Stone, 2009]

$s_1$: `An earthquake hit San Francisco. There was minor property damage, but no injuries.`

$R(s_1) = 8.0$

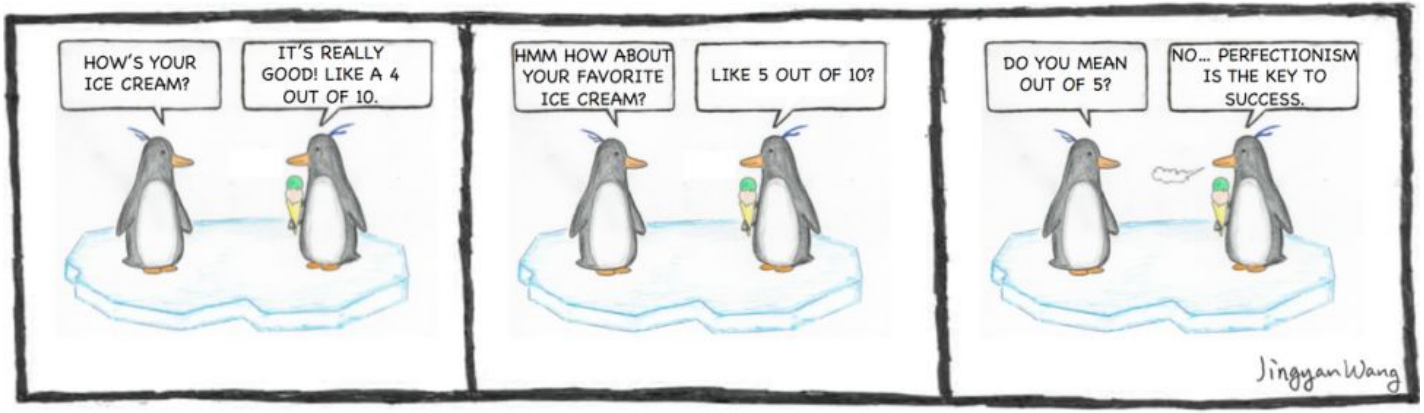
$s_2$: `The Bay Area has good weather but is prone to earthquakes and wildfires.`

$R(s_2) = 1.2$

Train an LM $RM_\phi(s)$ to predict human preferences from an annotated dataset, then optimize for $RM_\phi$ instead.

# How do we model human preferences?

**Problem 2**: human judgments are noisy and miscalibrated!




source

# How do we model human preferences?

**Problem 2**: human judgments are noisy and miscalibrated!

An earthquake hit San Francisco. There was minor property damage, but no injuries.

$s_1$

$R(s_1) = 8.0$

The Bay Area has good weather but is prone to earthquakes and wildfires.

$s_2$

$R(s_2) = 1.2$

# How do we model human preferences?

**Problem 2**: human judgments are noisy and miscalibrated!

An earthquake hit San Francisco. There was minor property damage, but no injuries.

$s_1$

$R(s_1) = 8.0$

A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.

$s_3$

$R(s_3) =$ 4.1? 6.6? 3.2?

The Bay Area has good weather but is prone to earthquakes and wildfires.

$s_2$

$R(s_2) = 1.2$

# How do we model human preferences?

**Problem 2**: human judgments are noisy and miscalibrated!

An earthquake hit San Francisco. There was minor property damage, but no injuries.

$s_1$

$R(s_1) = 8.0$

$>$

A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.

$s_3$

$R(s_3) =$ 4.1? 6.6? 3.2?

$>$

The Bay Area has good weather but is prone to earthquakes and wildfires.

$s_2$

$R(s_2) = 1.2$

**Solution**: instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable

# How do we model human preferences?

**Problem 2**: human judgments are noisy and miscalibrated!

An earthquake hit San Francisco. There was minor property damage, but no injuries.

$s_1$

$R(s_1) = 8.0$

$>$

A 4.2 magnitude earthquake hit San Francisco, resulting in massive damage.

$s_3$

$R(s_3) =$ 4.1? 6.6? 3.2?

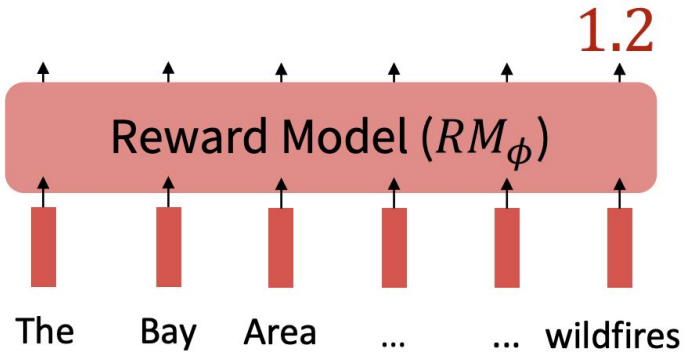
$>$

The Bay Area has good weather but is prone to earthquakes and wildfires.

$s_2$

$R(s_2) = 1.2$

**Solution**: instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable




Bradley-Terry [1952] paired comparison model

$$J_{RM}(\phi) = -\mathbb{E}_{(s^w, s^l)\sim D}\left[\log \sigma(RM_\phi(s^w) - RM_\phi(s^l))\right]$$

"winning" sample

"losing" sample

$s^w$ should score higher than $s^l$

# RLHF: Putting it all together

- Finally, we have everything we need:
  - A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
  - A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
  - A method for optimizing LM parameters towards an arbitrary reward function.

- Now to do RLHF:
  - Initialize a copy of the model $p_\theta^{RL}(s)$ , with parameters $\theta$ we would like to optimize
  - Optimize the following reward with RL:

$$R(s) = RM_\phi(s) - \underbrace{\beta \log \left( \frac{p_\theta^{RL}(s)}{p^{PT}(s)} \right)}$$

Pay a price when $p_\theta^{RL}(s) > p^{PT}(s)$

This is a penalty which prevents us from diverging too far from the pretrained model. In expectation, it is known as the **Kullback-Leibler (KL)** divergence between $p_\theta^{RL}(s)$ and $p^{PT}(s)$.

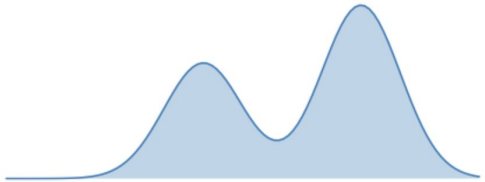
# ML review: Kullback-Leibler (KL) divergence

- A measure of how one probability distribution P is different from a second, reference probability distribution Q.

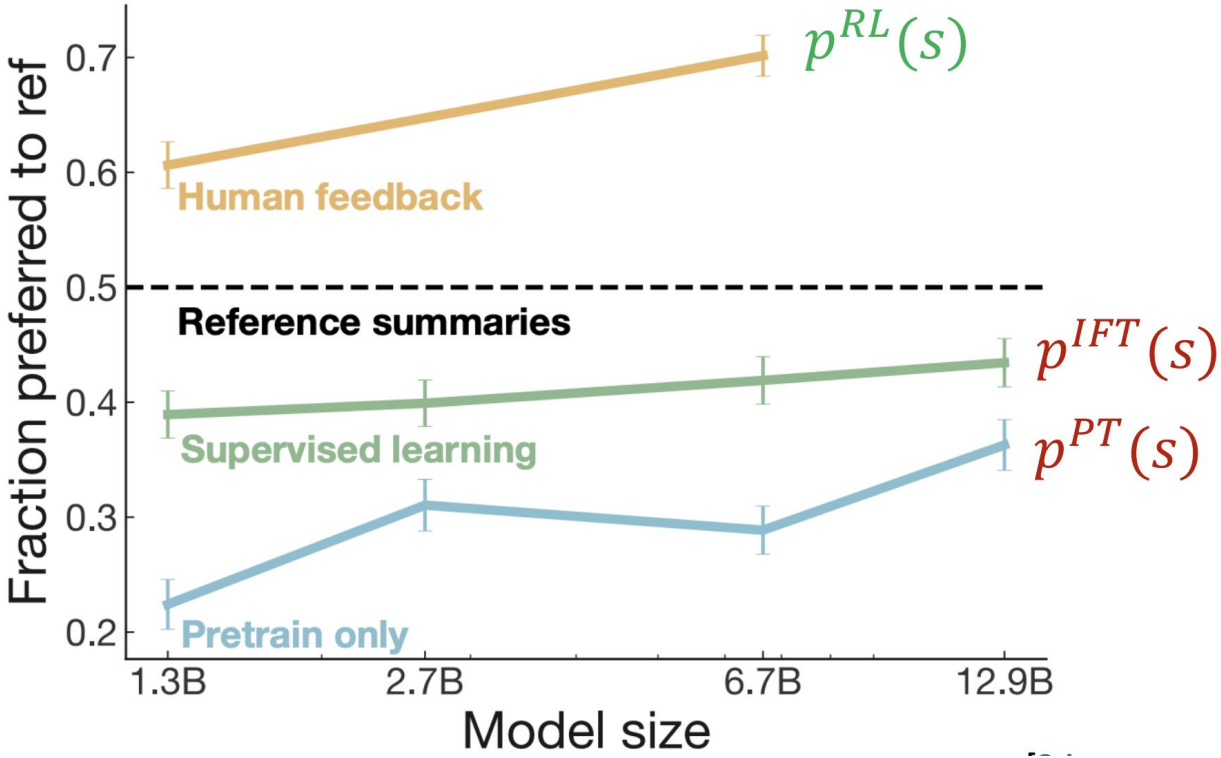
$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

- Asymmetric: $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$

- **Exercise**: There is some true distribution P(X) that we're trying to approximate using our approximate distribution Q(X)
  - Assume P(X) is a bimodal distribution and Q(X) is a unimodal normal distribution
- What would Q look like if we minimize $D_{KL}(P||Q)$ or $D_{KL}(Q||P)$?

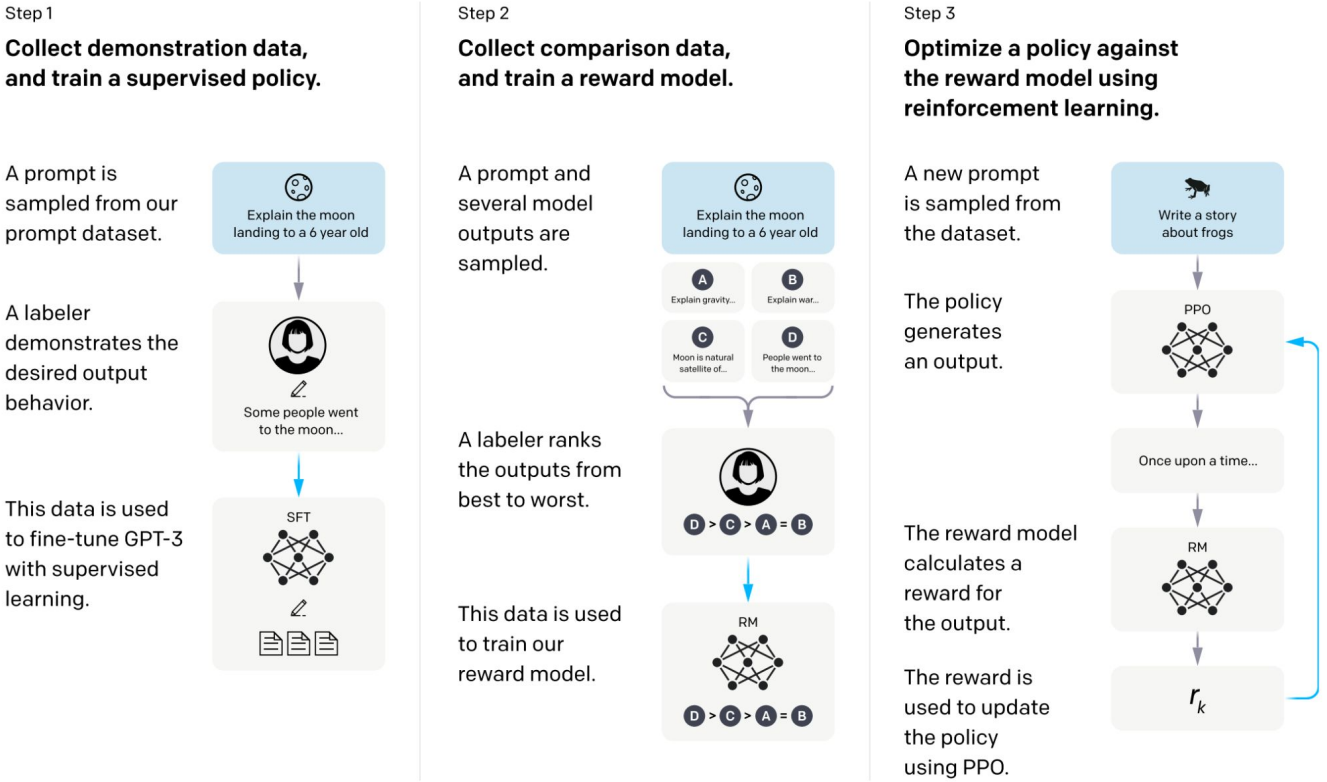


# RLHF provides gains over pretraining + finetuning




https://arxiv.org/abs/2009.01325

# InstructGPT: scaling up RLHF to tens of thousands of tasks




https://arxiv.org/abs/2203.02155

# InstructGPT

**PROMPT** *Explain the moon landing to a 6 year old in a few sentences.*

**COMPLETION**

GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and
sent them back to the earth so we could all see them.
```

https://openai.com/research/instruction-following

# InstructGPT

**PROMPT** *Write a short poem about a wise frog.*

**COMPLETION**

GPT-3

```
Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you
have lost.

Write a short story in which a character has a bad dream.
```

InstructGPT

```
The frog is a symbol of wisdom
He knows all the secrets of the world
He is a master of disguise
And a great teacher of life
He is a symbol of transformation
And the bringer of change
He is the frog who has seen it all
And knows the meaning of it all
```

https://openai.com/research/instruction-following

# ChatGPT: Instruction Finetuning + RLHF for dialog agents

**ChatGPT: Optimizing Language Models for Dialogue**

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge…

## Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

**(Instruction finetuning!)**

https://openai.com/blog/chatgpt/

# ChatGPT: Instruction Finetuning + RLHF for dialog agents

**ChatGPT: Optimizing Language Models for Dialogue**

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge…

## Methods

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using Proximal Policy Optimization. We performed several iterations of this process.

**(RLHF!)**

https://openai.com/blog/chatgpt/

# Large language models

- How to train language models
- Protein language models
  - Project ideas
- How to build ChatGPT