



# Intermediate Deep Learning for Engineers

---

Spring 2025, Deep Learning for Engineers  
April 17, 2025, 11th Session

Amir Barati Farimani  
*Associate Professor of Mechanical Engineering and Bio-Engineering*  
*Carnegie Mellon University*

# LLMs

Model	Creators	Year	Model Size	Context Size
GPT-2	OpenAI	2019	1.5 billion	1024
GPT-3	OpenAI	2020	175 billion	2048
PaLM	Google	2022	540 billion	2048
LLaMA	Meta	2023	65 billion	2048
LLaMA-2	Meta	2023	70 billion	4096
Claude-2	Anthropic	2023	? (130 billion)	100k
Claude-2.1	Anthropic	2023	? (130 billion)	200k
GPT-4	OpenAI	2023	? (1.76 trillion)	8192
Mistral	Mistral AI	2023	7 billion	8192 (32k)
Mixtral	Mistral AI	2023	47 billion	8192 (128k)
Gemini (Ultra)	Google	2023	? (1.5 trillion)	32k
LWModel	academia!	2023	7 billion	1 million
Gemini-1.5	Google	2024	? (1.5 trillion)	1 million
GPT-4o	OpenAI	2024	?	128k
LLaMA-3	Meta	2024	405 billion	8192
LLaMA-3.1	Meta	2024	405 billion	128k
Claude-3.5	Anthropic	2024	?	200k

## GPT-2 (a Transformer)

- Technically, it doesn't use words as input but **Byte Pair Encodings** (sub-words), similar to BERT's WordPieces.
- Includes **positional embeddings** as part of the input, too.
- Easy to fine-tune on your own dataset (language)



# GPT-2 (a Transformer)



Image by <http://jalammar.github.io/illustrated-gpt2/>

## Byte Pair Encodings (BPE)

- Invented in 1994 ([Gage](#)) and updated in 2015 ([Sennrich et al.](#))
- Looks at the individual symbols (e.g., characters) and repeated merges the most frequent pairs (a la agglomerative clustering)
- Stop after **N** merges (you specify **N**). GPT uses **N** =40k

Philip Gage. 1994. A New Algorithm for Data Compression. *C Users J.*, 12(2):23–38, February

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015



# GPT-2 (a Transformer)

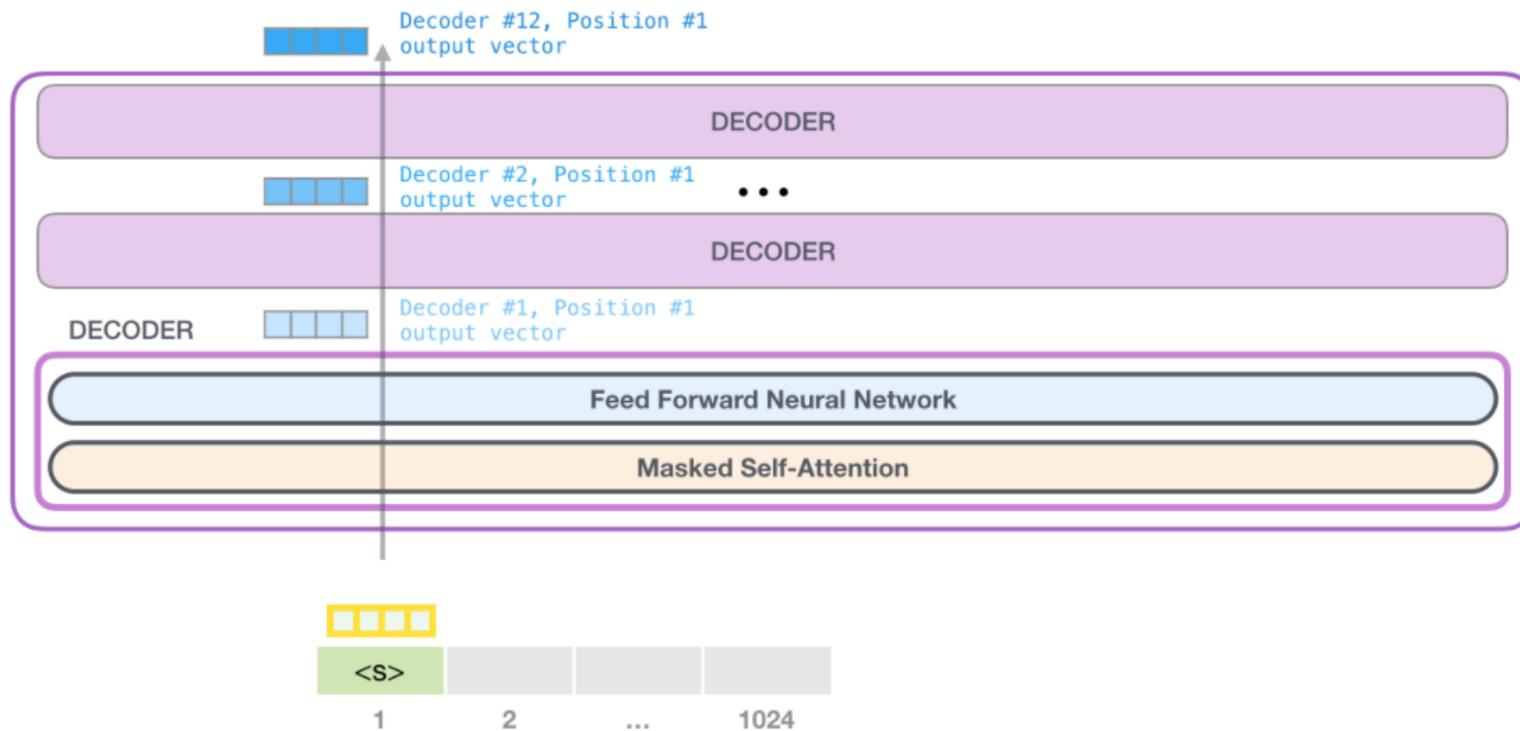


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's Masked Attention

For efficiency, we can still calculate all query-key calculations with matrix multiplications, then mask before softmax'ing.

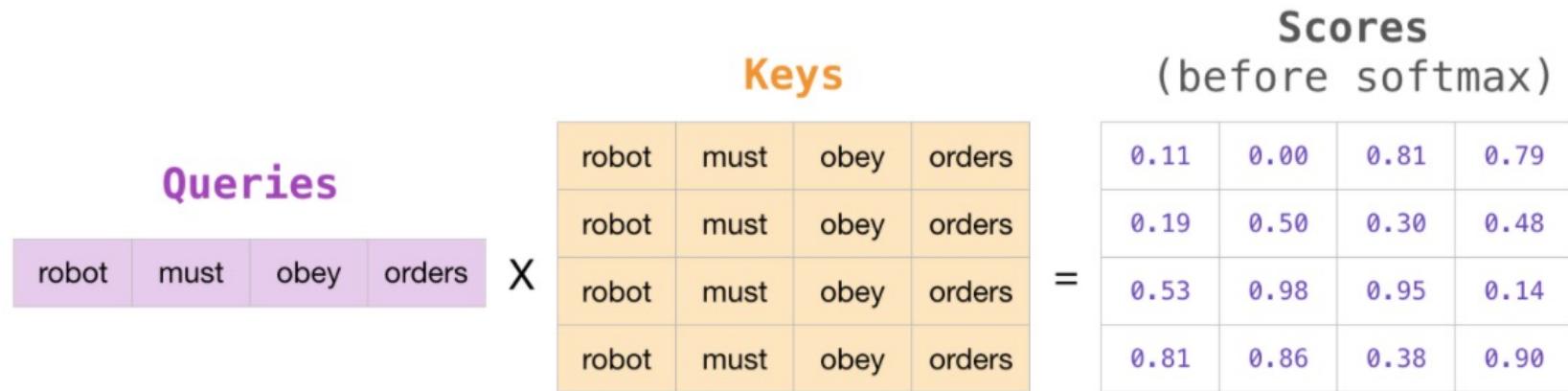


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's Masked Attention

For efficiency, we can still calculate all query-key calculations with matrix multiplications, then mask before softmax'ing.



Image by <http://jalammar.github.io/illustrated-gpt2/>

## GPT-2's Masked Attention

For efficiency, we can still calculate all query-key calculations with matrix multiplications, then mask before softmax'ing.

**Masked Scores**  
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

**Softmax**  
(along rows) →

**Scores**

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

# GPT-2's

Representations are propagated upwards through the network

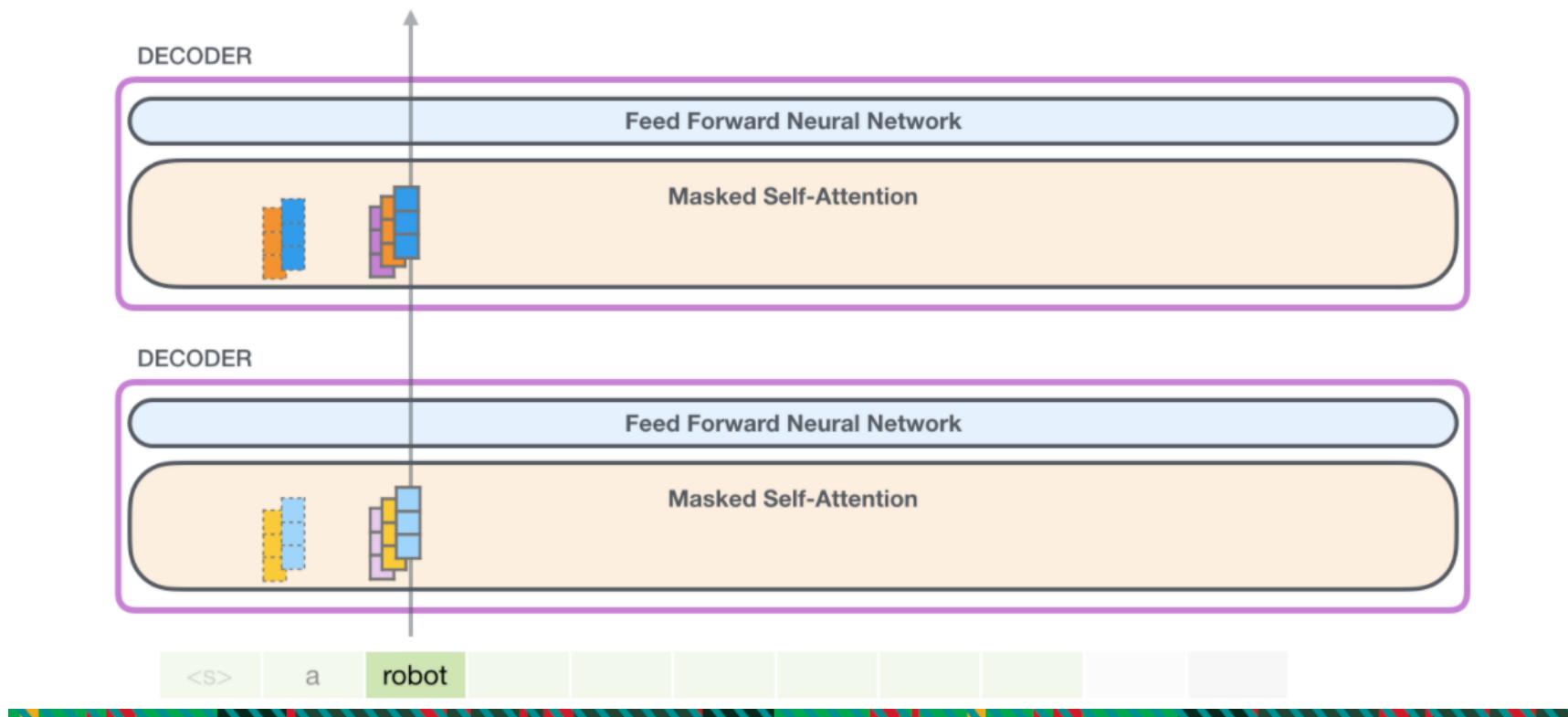


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's

Self-attention is otherwise identical to what we saw in BERT

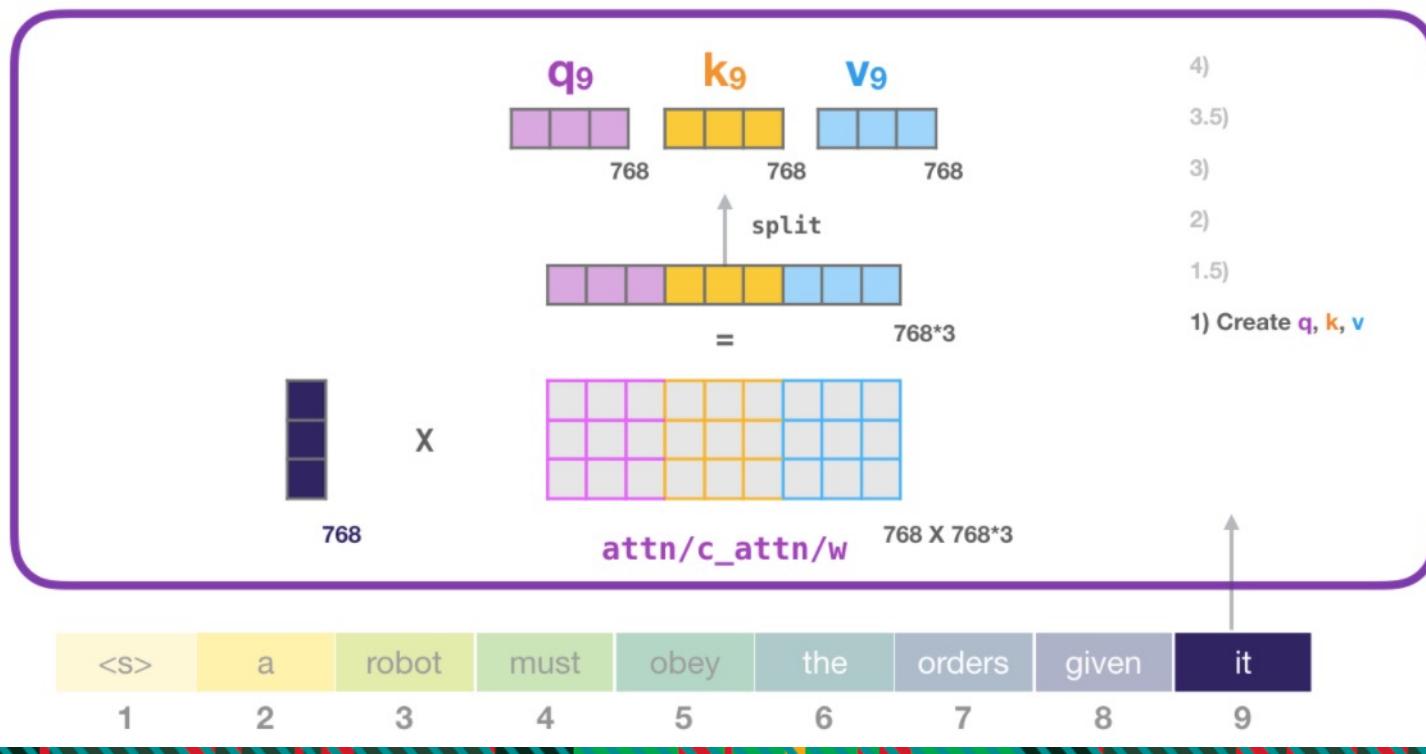


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's

Can have Multiple Self-Attention heads

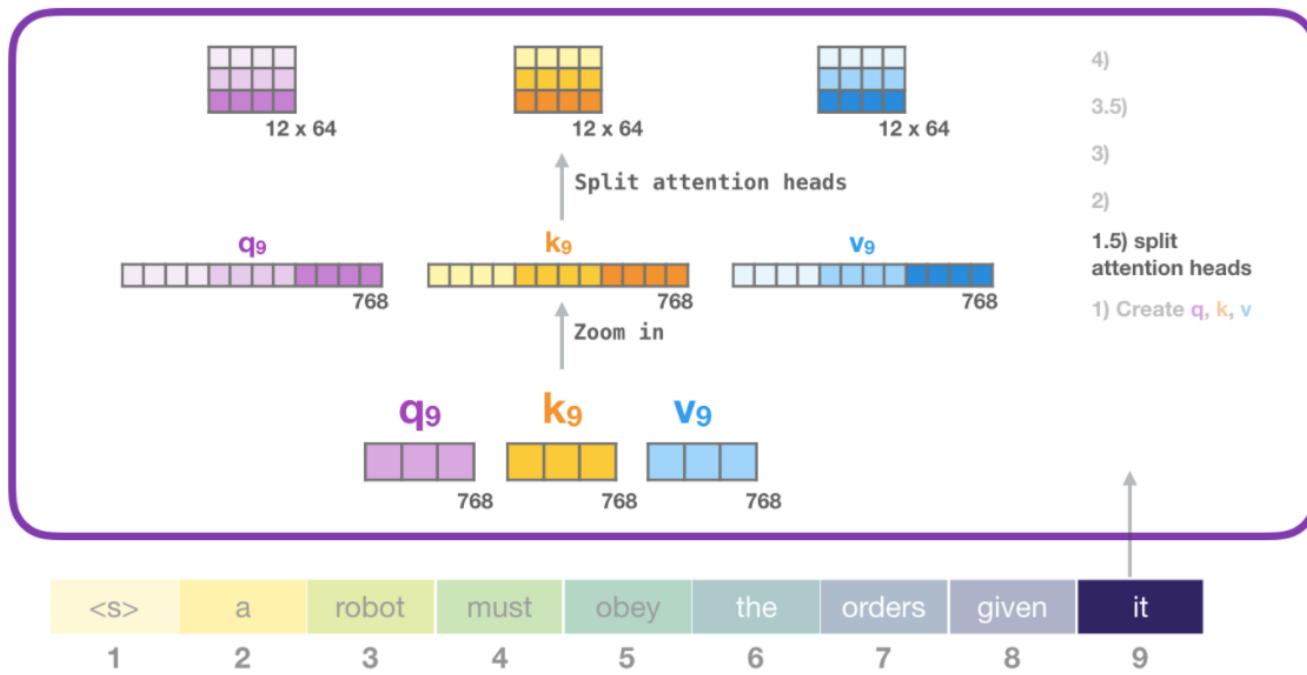


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's

Each Self-Attention head is responsible for exactly 1 resulting,  
output embedding

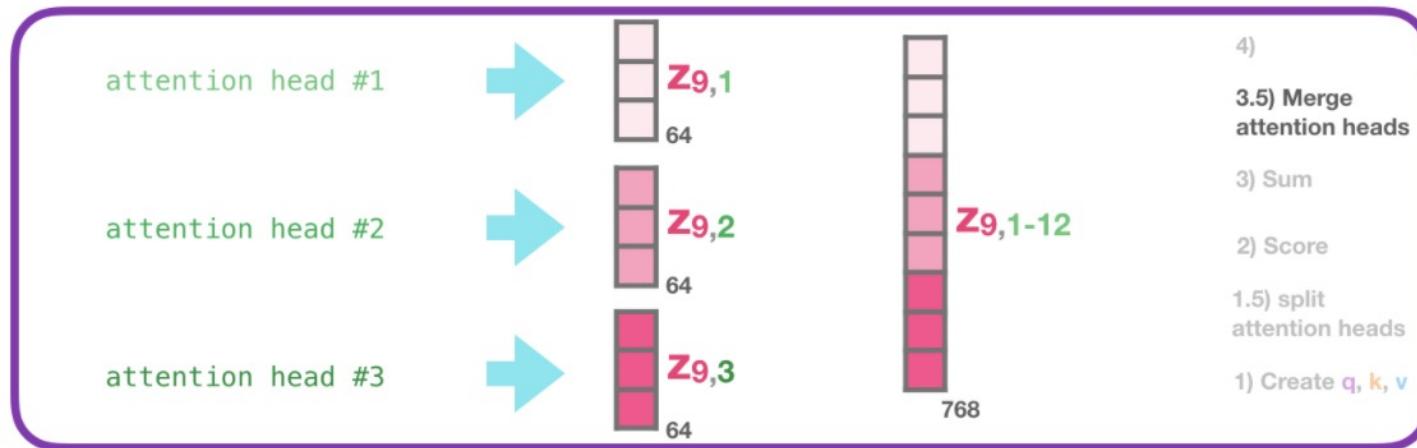


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's

Remember, these Masked Self-Attention layers are fed into a FFNN

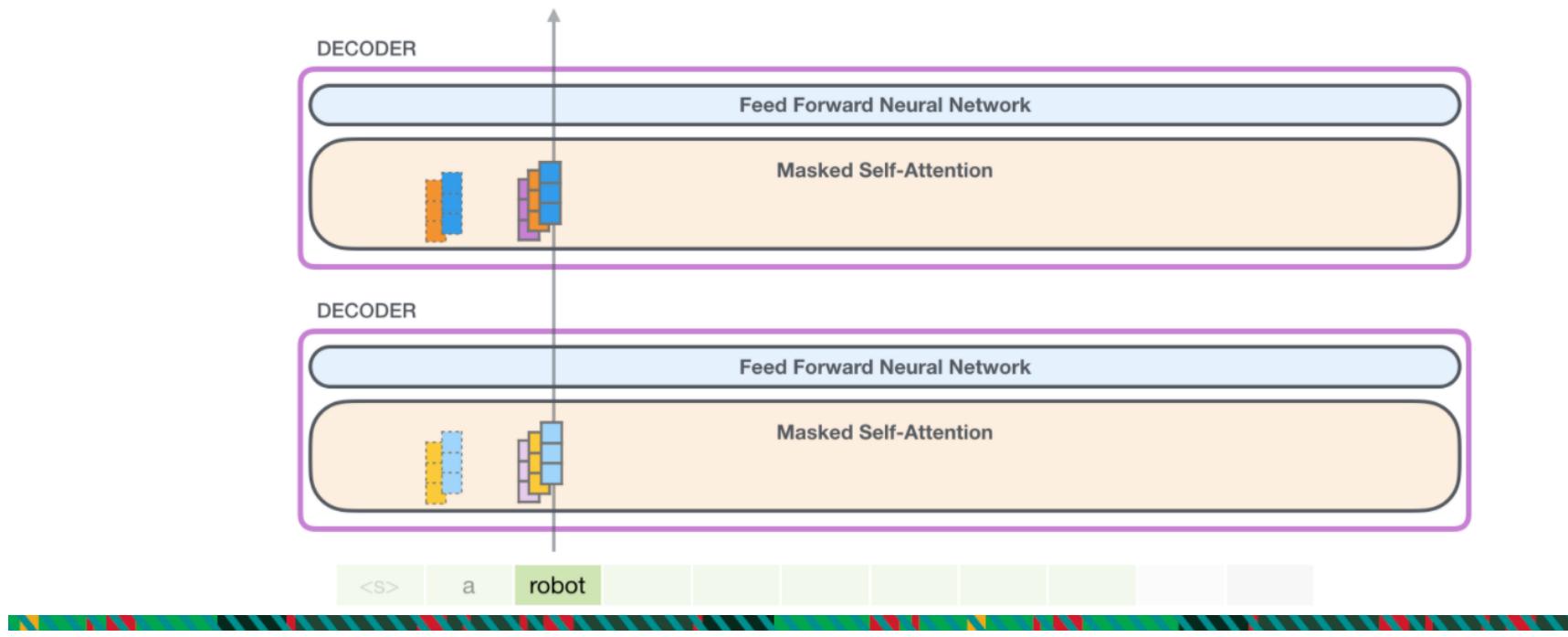
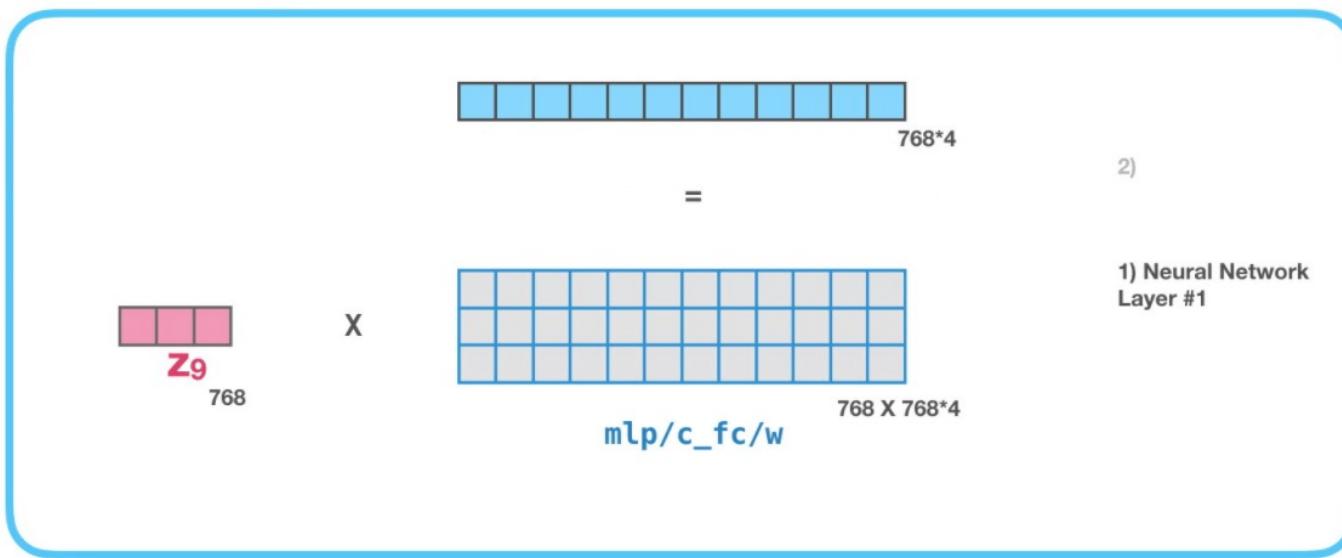


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's

Remember, these Masked Self-Attention layers are fed into a FFNN



First hidden layer expands to 4x in size of the input

Image by <http://jalammar.github.io/illustrated-gpt2/>

## GPT-2's

2<sup>nd</sup> (final) layer of the FFNN projects it back to the original size

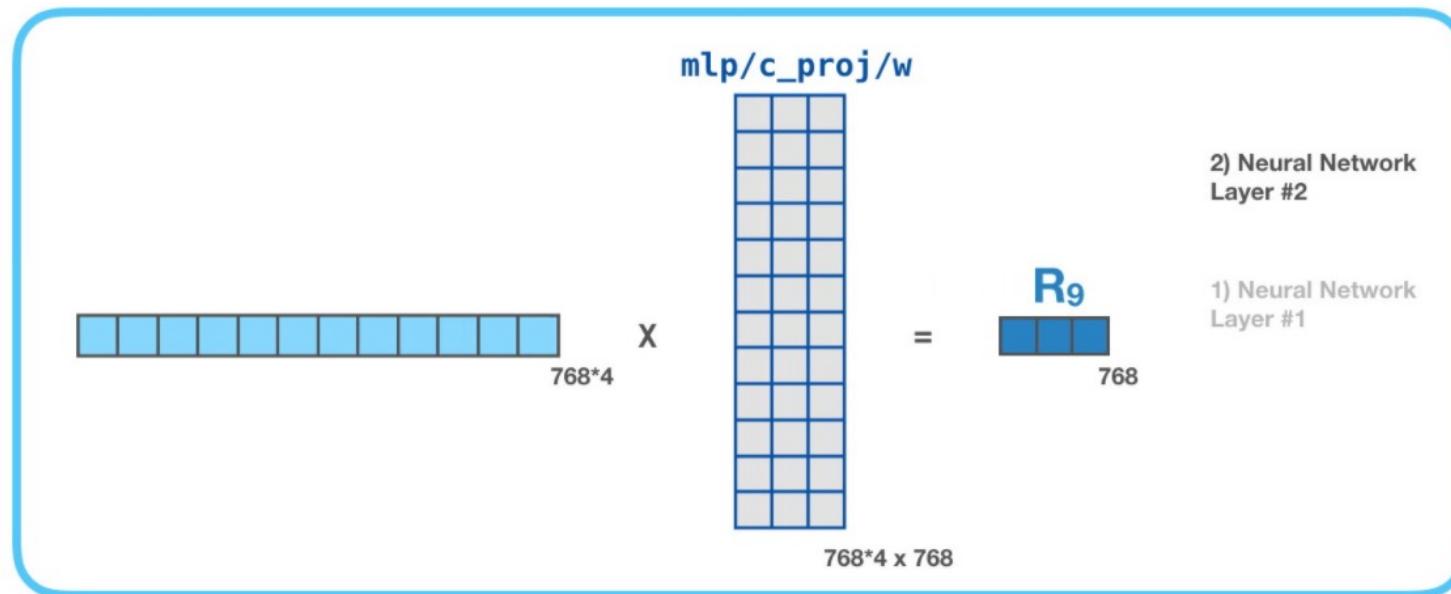


Image by <http://jalammar.github.io/illustrated-gpt2/>

# GPT-2's

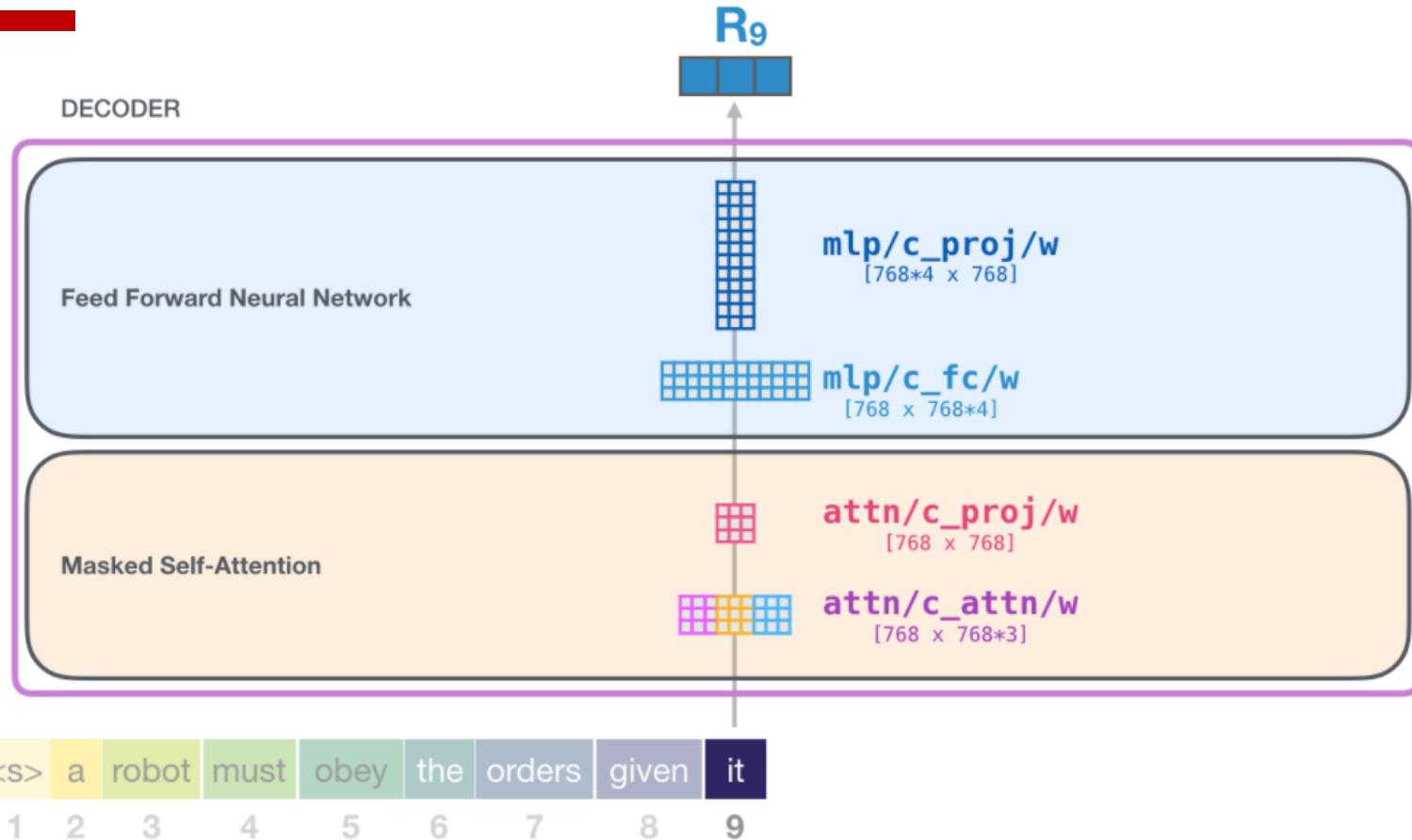


Image by <http://jalamar.github.io/illustrated-gpt2/>

Each Decoder block has its own weights (e.g.,  $W_k, W_q, W_v$ )

But the entire model only has 1 token-embedding weight matrix and positional encoding weight matrix. This helps all the blocks to work together and supplement their captured aspects

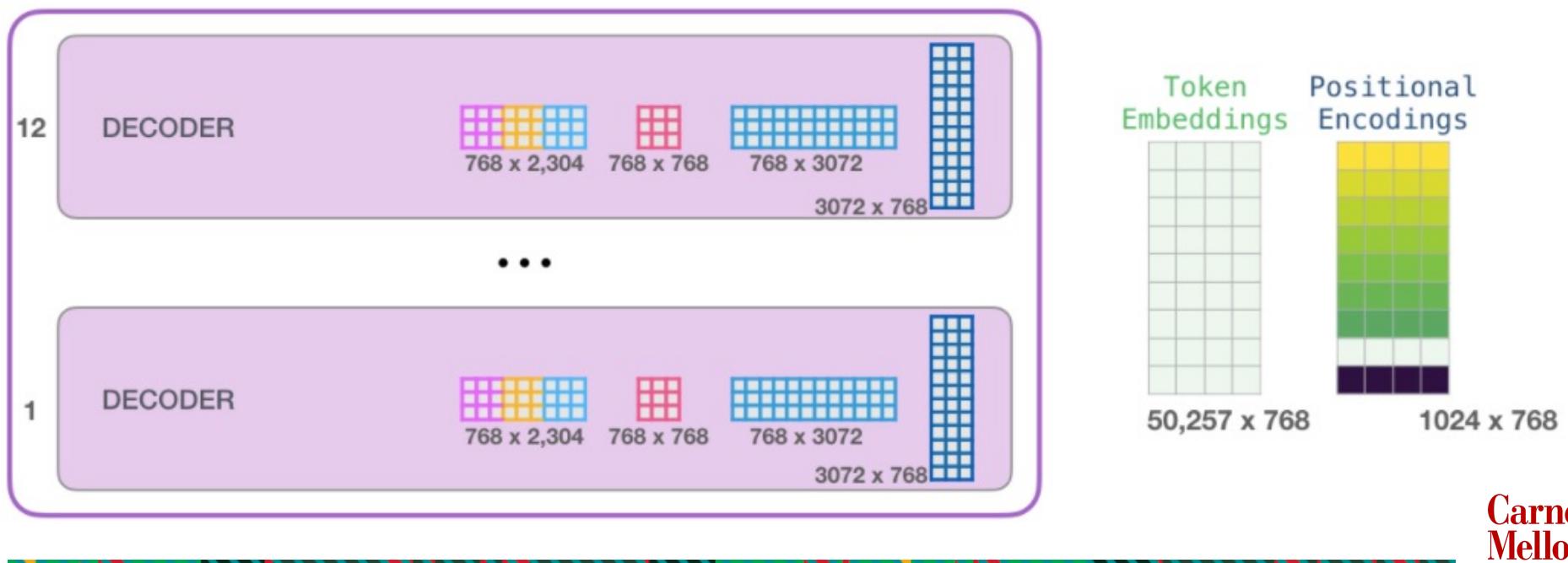


Image by <http://jalammar.github.io/illustrated-gpt2/>

The authors of GPT-2 created 4 different version (sizes) of the model

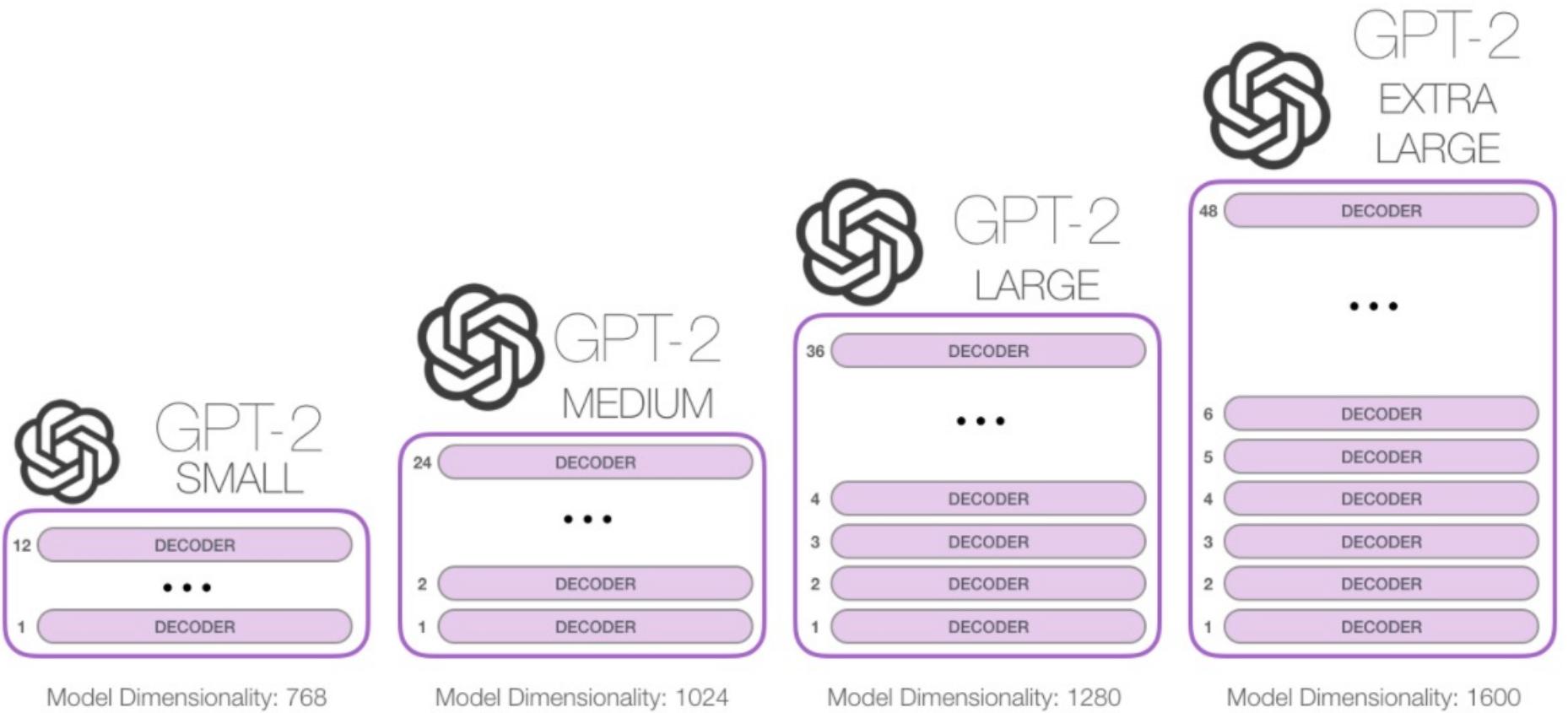


Image by <http://jalammar.github.io/illustrated-gpt2/>

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

- has Layer normalization in between each sub-block (as we've already seen)
- 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 \*<sup>1</sup> Jeffrey Wu \*<sup>1</sup> Rewon Child<sup>1</sup> David Luan<sup>1</sup> Dario Amodei \*\*<sup>1</sup> Ilya Sutskever \*\*<sup>1</sup>

gic  
University

You can finagle the system to yield synthetic predictions.

**Children's Book Test (CBT)** is a classification task.  
Fill-in-the-blank, and you predict which of the 10 possible choices is correct.

You can compute the probability of each choice + its ending.



You can finagle the system to yield synthetic predictions.

**LAMBADA** dataset tests model's ability to understand long-range dependencies.

**Task:** predict the final word of sentences which humans need 50+ tokens of context in order to accurately predict.



# GPT-2 Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3
117M	<b>35.13</b>	45.99	<b>87.65</b>	<b>83.4</b>	<b>29.41</b>	65.85	1.16	1.17	37.50
345M	<b>15.60</b>	55.48	<b>92.35</b>	<b>87.1</b>	<b>22.76</b>	47.33	1.01	<b>1.06</b>	26.37
762M	<b>10.87</b>	<b>60.12</b>	<b>93.45</b>	<b>88.0</b>	<b>19.93</b>	<b>40.31</b>	<b>0.97</b>	<b>1.02</b>	22.05
1542M	<b>8.63</b>	<b>63.24</b>	<b>93.30</b>	<b>89.05</b>	<b>18.34</b>	<b>35.76</b>	<b>0.93</b>	<b>0.98</b>	<b>17.48</b>

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

You can finagle the system to yield synthetic predictions.

**Summarization.** The add the text “TL;DR:” after an article, then generate 100 tokens with top-2 random sampling, then extract the first 3 sentences.



## GPT-2 Results

	R-1	R-2	R-L	R-AVG
Bottom-Up Sum	<b>41.22</b>	<b>18.68</b>	<b>38.34</b>	<b>32.75</b>
Lede-3	40.38	17.66	36.62	31.55
Seq2Seq + Attn	31.33	11.81	28.83	23.99
GPT-2 TL; DR:	29.34	8.27	26.58	21.40
Random-3	28.78	8.63	25.52	20.98
GPT-2 no hint	21.58	4.03	19.47	15.03

*Table 4.* Summarization performance as measured by ROUGE F1 metrics on the CNN and Daily Mail dataset. Bottom-Up Sum is the SOTA model from (Gehrmann et al., 2018)



# GPT-2 Results

Question	Generated Answer	Correct	Probability
Who wrote the book the origin of species?	Charles Darwin	✓	83.4%
Who is the founder of the ubuntu project?	Mark Shuttleworth	✓	82.0%
Who is the quarterback for the green bay packers?	Aaron Rodgers	✓	81.1%
Panda is a national animal of which country?	China	✓	76.8%
Who came up with the theory of relativity?	Albert Einstein	✓	76.4%
When was the first star wars film released?	1977	✓	71.4%
What is the most common blood type in sweden?	A	✗	70.6%
Who is regarded as the founder of psychoanalysis?	Sigmund Freud	✓	69.3%
Who took the first steps on the moon in 1969?	Neil Armstrong	✓	66.8%
Who is the largest supermarket chain in the uk?	Tesco	✓	65.3%
What is the meaning of shalom in english?	peace	✓	64.0%
Who was the author of the art of war?	Sun Tzu	✓	59.6%
Largest state in the us by land mass?	California	✗	59.2%
Green algae is an example of which type of reproduction?	parthenogenesis	✗	56.5%
Vikram samvat calender is official in which country?	India	✓	55.6%
Who is mostly responsible for writing the declaration of independence?	Thomas Jefferson	✓	53.3%

# GPT-2 Results

**Easy to fine-tune on your own dataset (language)**

## SYSTEM PROMPT (HUMAN-WRITTEN)

A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.

## MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)

The incident occurred on the downtown train line, which runs from Covington and Ashland stations.

In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.

“The theft of this nuclear material will have significant negative consequences on public and environmental health, our workforce and the economy of our nation,” said Tom Hicks, the U.S. Energy Secretary, in a statement. “Our top priority is to secure the theft and ensure it doesn’t happen again.”

The stolen material was taken from the University of Cincinnati’s Research Triangle Park nuclear research site, according to a news release from Department officials.

The Nuclear Regulatory Commission did not immediately release any information.

---

## Context (WebText test)

### Pinky

The pink ghost's AI is designed to "feel" opposite of the red ghost's behavior. Pinky actually attempts to get out in front of Pac-Man. This is accomplished by setting the target 4 tiles ahead of Pac-Man's current location in the direction that Pac-Man is travelling. One exception to this is when Pac-Man is traveling up. Due to an overflow bug in the code, the calculation includes a left offset equal to the expected up offset.

The use of Pac-Man's expected position allows the player to exploit the navigation code for Pinky. This was most often seen in situations where Pac-Man and Pinky are in close proximity to each other. For example, if they were headed toward one another, once they are less than 4 tiles apart, the target tile is now behind Pinky. If he encounters an intersection while in this configuration, he will attempt to double-back (by turning) in order to reach the target tile.

Player's would often perform what was referred to as a "head fake" by changing directions rapidly just as Pinky was approaching an intersection. By displacing the target tile just as Pinky



## Context (WebText test)

Pinky

The pink  
get out in t  
the directio  
bug in the

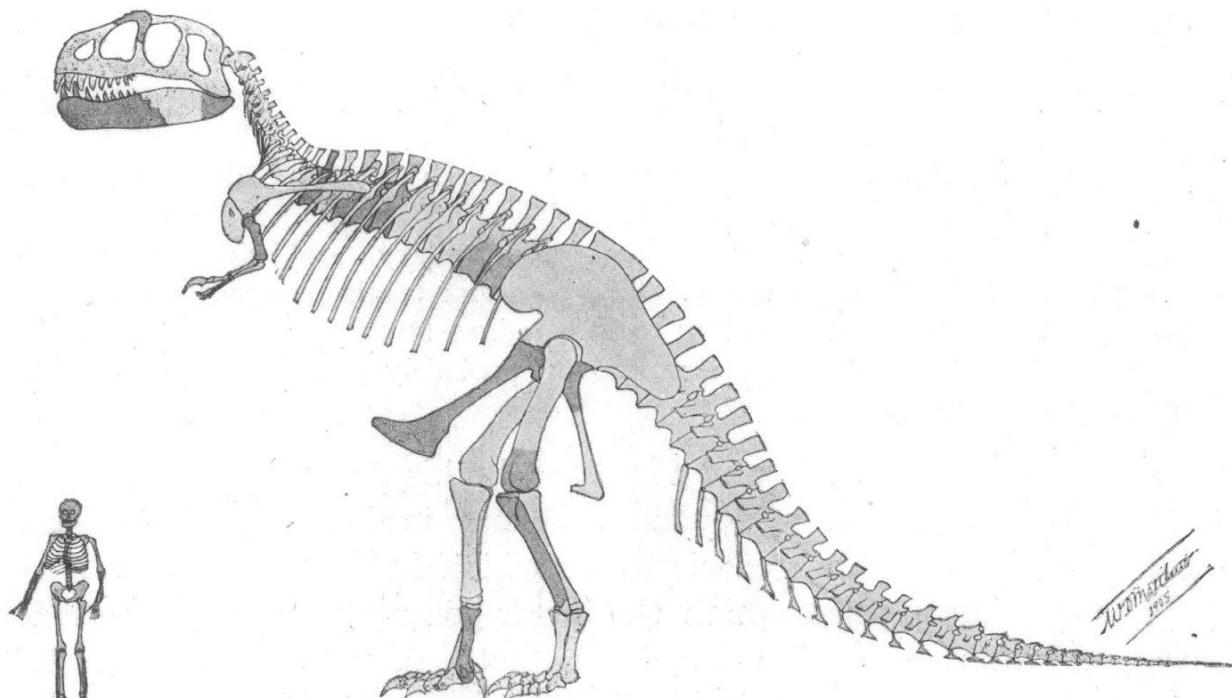
The use of  
often seen  
headed tow  
an intersec

Player's w  
as Pinky w

was about to cross the intersection, Pac-Man would be able to dodge the ghosts' projectiles and return to the safe location of the safe house. Buster The yellow ghost's AI is specifically designed to try to avoid Pac-Man from leaving the safe house. The player must press the SPACE bar to reveal the safe house's doors, then go into them to find the yellow ghost hiding in the next room. The player must also note that the ghost will not attempt to exit the safe house if it sees Pac-Man already there. In fact, the ghost will not attempt an exit in the first place unless they're already in the safe house. These two behaviors are based on the fact that the player does not always attempt to reach the safe house before Pac-Man (or Blinky). The player must also beware of this glitch if a ghost (or another player) is already in the safe house and attempting to get out as soon as Pac-Man or Blinky is gone.

# Comparison between GPT2 and GPT3

---



**GPT-2**  
**1.5B Parameters**

**GPT-3**  
**175B Parameters**

Carnegie  
Mellon  
University

# GPT-3 (Brown et al. 2020)

---

## Language Models are Few-Shot Learners

---

Tom B. Brown\*

Benjamin Mann\*

Nick Ryder\*

Melanie Subbiah\*

Jared Kaplan†

Prafulla Dhariwal

Arvind Neelakantan

Pranav Shyam

Girish Sastry

Amanda Askell

Sandhini Agarwal

Ariel Herbert-Voss

Gretchen Krueger

Tom Henighan

Rewon Child

Aditya Ramesh

Daniel M. Ziegler

Jeffrey Wu

Clemens Winter

Christopher Hesse

Mark Chen

Eric Sigler

Mateusz Litwin

Scott Gray

Benjamin Chess

Jack Clark

Christopher Berner

Sam McCandlish

Alec Radford

Ilya Sutskever

Dario Amodei

OpenAI



# GPT-3 various architectures

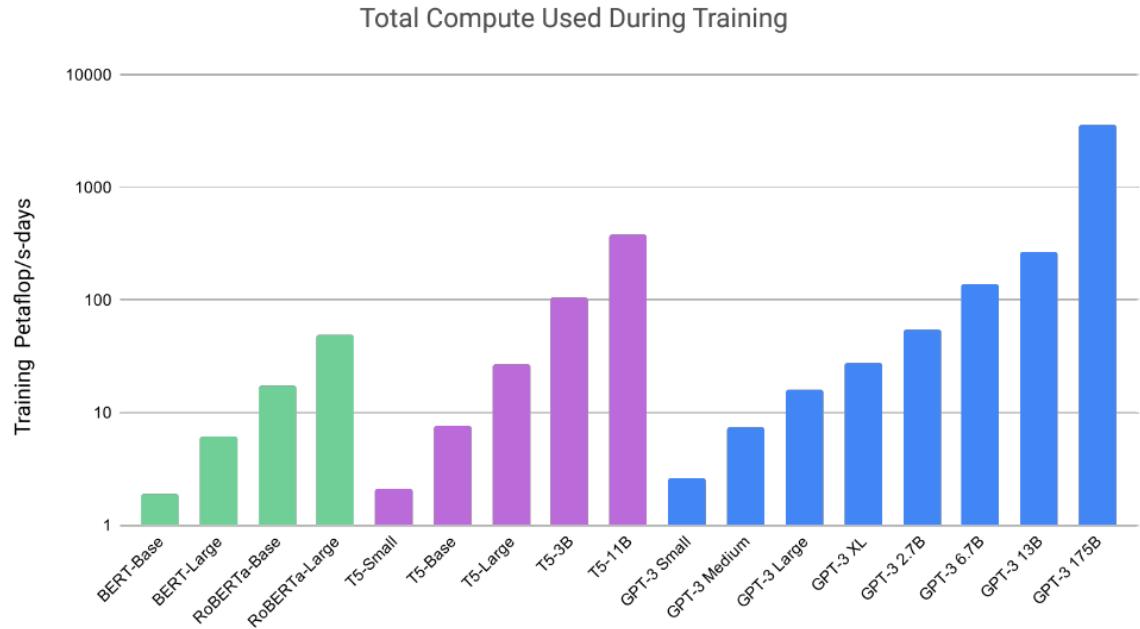
---

Model Name	$n_{\text{params}}$	$n_{\text{layers}}$	$d_{\text{model}}$	$n_{\text{heads}}$	$d_{\text{head}}$	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	$6.0 \times 10^{-4}$
GPT-3 Medium	350M	24	1024	16	64	0.5M	$3.0 \times 10^{-4}$
GPT-3 Large	760M	24	1536	16	96	0.5M	$2.5 \times 10^{-4}$
GPT-3 XL	1.3B	24	2048	24	128	1M	$2.0 \times 10^{-4}$
GPT-3 2.7B	2.7B	32	2560	32	80	1M	$1.6 \times 10^{-4}$
GPT-3 6.7B	6.7B	32	4096	32	128	2M	$1.2 \times 10^{-4}$
GPT-3 13B	13.0B	40	5140	40	128	2M	$1.0 \times 10^{-4}$
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	$0.6 \times 10^{-4}$

**Table 2.1:** Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

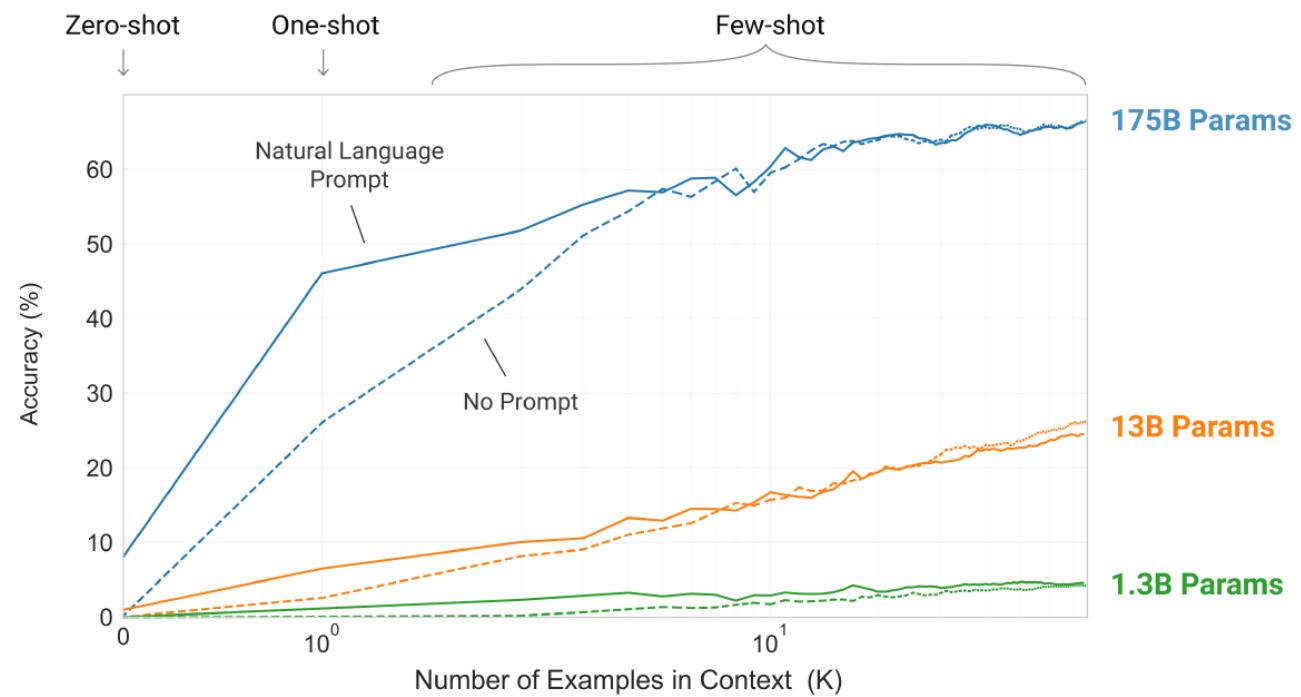
# Training compute

List price of compute to train  
GPT-3 175B: ~\$4.5M



**Figure 2.2: Total compute used during training.** Based on the analysis in Scaling Laws For Neural Language Models [KMH<sup>+</sup>20] we train much larger models on many fewer tokens than is typical. As a consequence, although GPT-3 3B is almost 10x larger than RoBERTa-Large (355M params), both models took roughly 50 petaflop/s-days of compute during pre-training. Methodology for these calculations can be found in Appendix D.

# GPT3 performance



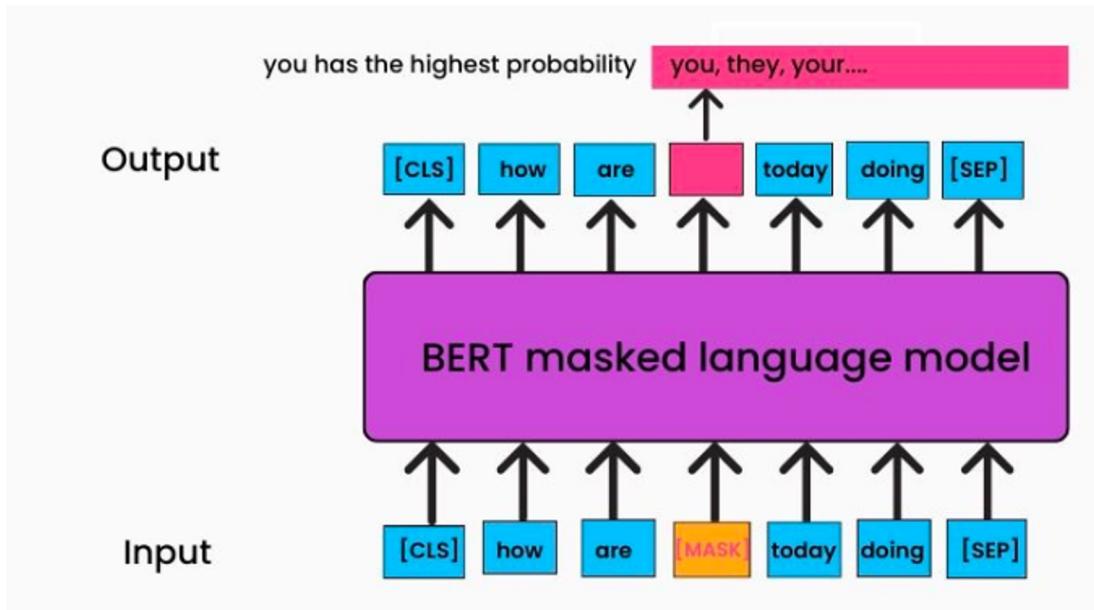
## To Sum Up

---

1. GPT 3 WITH 175 B PARAMETERS IS CAPABLE OF LEARNING CONTEXT
2. GPT3 IS USED IN CHATGPT WITH POWERFUL CAPABILITY OF PERFORMING DIFFERENT TASKS

# BERT

Encoder-only architecture. 340 million parameters.



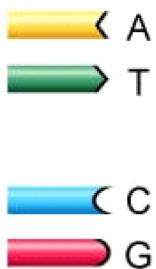
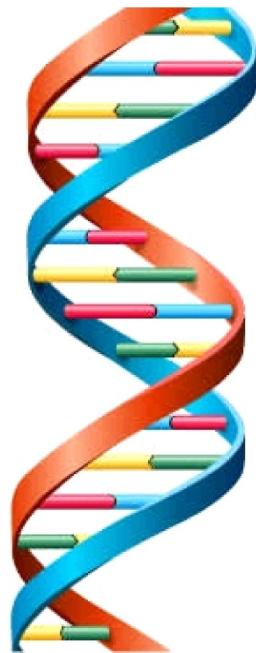
Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018)

Taken from <https://www.turing.com/kb/how-bert-nlp-optimization-model-works>

# BERT in Biology

<https://arxiv.org/abs/2103.07162>

## DNA BERT



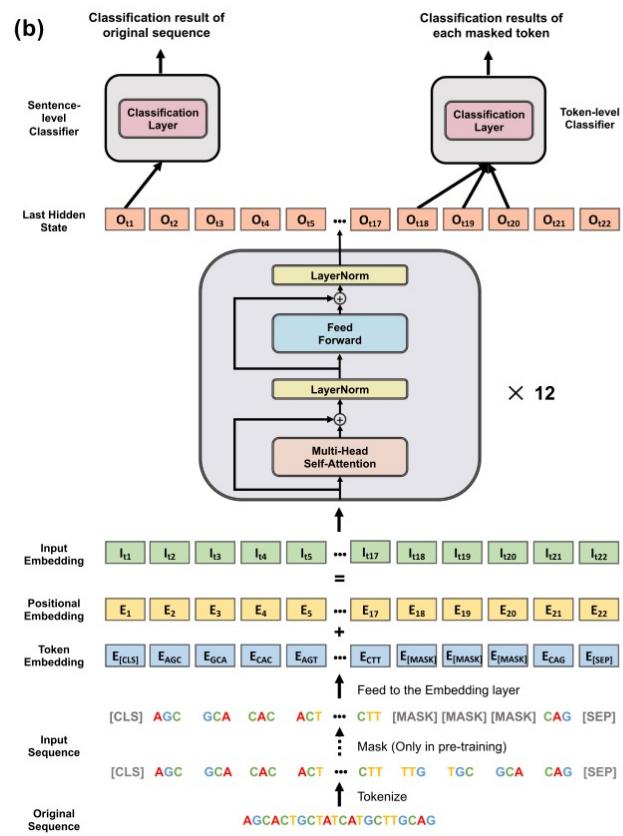
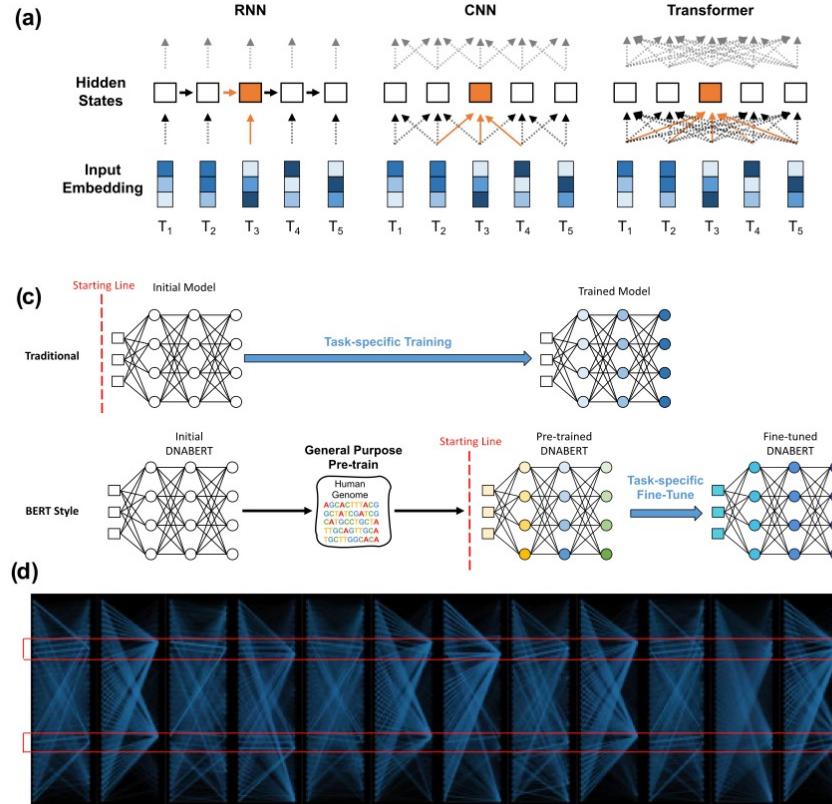
EI	CCAGCTGCATCACAGGAGGCCAGCGAGCAGGTCTGTTCCAAGGGCC1
EI	AGACCCGCCGGGAGGCAGGACCTGCAGGGTGAGCCCCACCGCCC
IE	AACGTGGCCTCCTGTGCCCTCCCCACAGTGCCCTTCCAGGACAA
IE	CCACTCAGCCAGGCCCTTCTTCTCCTCCAGGTCCCCACGGCCCTTCAG
IE	CCTGATCTGGGTCTCCCCTCCCACCCTCAGGGAGCCAGGGCTCGGCATT
IE	AGCCCTCAACCCTCTGTCTCACCCCTCCAGCCTAAAGCTCCTGACAAC
IE	CCACTCAGCCAGGCCCTTCTCCTCCAGGTCCCCACGGCCCTTCAG
N	CTGTGTTACCAACATCAAGCGCCGGACATCGTGCTCAAGTGGGAGC1
N	GTGTTACCGAGGGCATTCTAACAGTCTTACTACGGCCTCCGCCGA
N	TCTGAGCTCTGCATTGTCTATTCTCCAGCTGACCCCTGGTTCTCTCTT,



# BERT in Biology

<https://arxiv.org/abs/2103.07162>

DN



# BERT in Biology

## PROTEIN BERT

### PeptideBERT: A Language Model Based on Transformers for Peptide Property Prediction

Chakradhar Guntuboina, Adrita Das, Parisa Mollaei, Seongwon Kim, and Amir Barati Farimani\*

 Cite this: *J. Phys. Chem. Lett.* 2023, 14, 46, 10427–

10434

Publication Date: November 13, 2023 

<https://doi.org/10.1021/acs.jpclett.3c02398>

Copyright © 2023 The Authors. Published by American Chemical Society. This publication is licensed under

[CC-BY 4.0](#).

Open Access

Article Views

3647

[LEARN ABOUT THESE METRICS](#)

Altmetric

2

Citations

2

Share



Add to

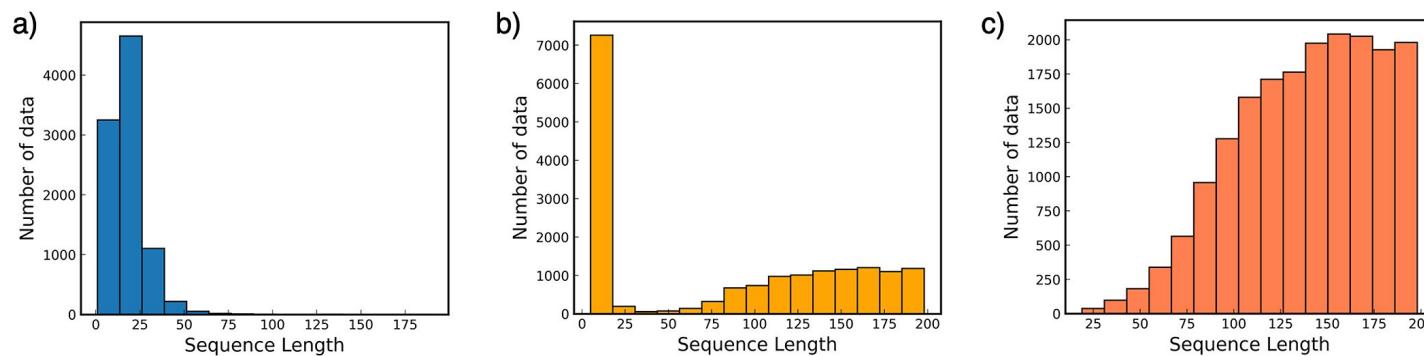


Export



# BERT in Biology

## PROTEIN BERT

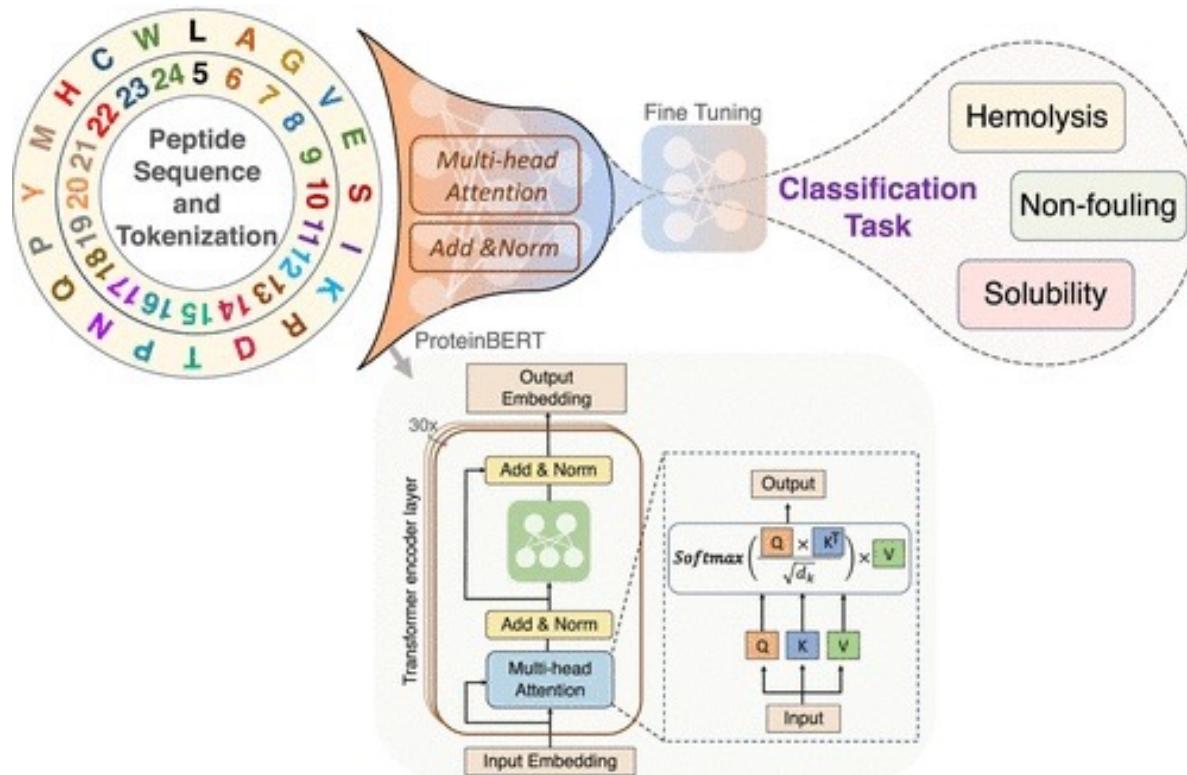


Sequence length of each peptide property data set: (a) hemolysis, (b) nonfouling, and (c) solubility.



# BERT in Biology

## PROTEIN BERT

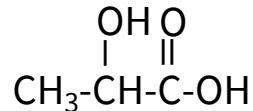


## BERT IN CHEMISTRY

### SMILES

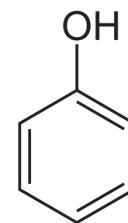
- THE SIMPLIFIED MOLECULAR-INPUT LINE-ENTRY SYSTEM (SMILES) IS USED TO DESCRIBE THE STRUCTURE OF MOLECULES USING SHORT ASCII STRINGS

Structural formula



SMILES

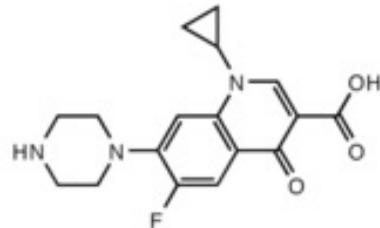
CC(O)C(=O)O



Oc1ccccc1  
OC1=CC=CC=C1

## SMILES

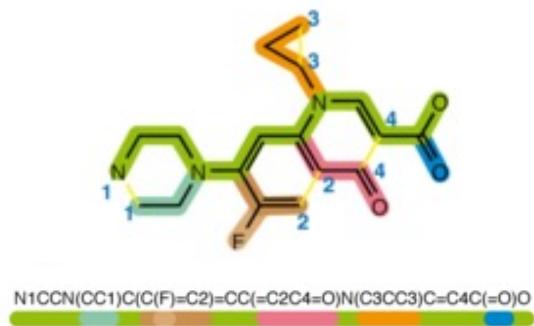
- THE SIMPLIFIED MOLECULAR-INPUT LINE-ENTRY SYSTEM (SMILES) IS USED TO DESCRIBE THE STRUCTURE OF MOLECULES USING SHORT ASCII STRINGS



Structural formula



Breaking rings



Determining the main sequence and branches

## SMILES-BASED MRL METHODS

- SMILES-BASED MRL METHODS...
  - take SMILES strings as input
  - use language models (BERT, Transformer) as their base models
  - output hidden layers as molecule embeddings
- Examples:
  - MolBERT (Fabian et al., 2020)
  - ChemBERTa (Chithrananda et al., 2020)
  - SMILES-Transformer (Honda et al., 2019)
  - SMILES-BERT (Wang et al., 2019)
  - Molecule-Transformer (Shin et al., 2019)
  - SA-BiLSTM (Zheng et al., 2019)

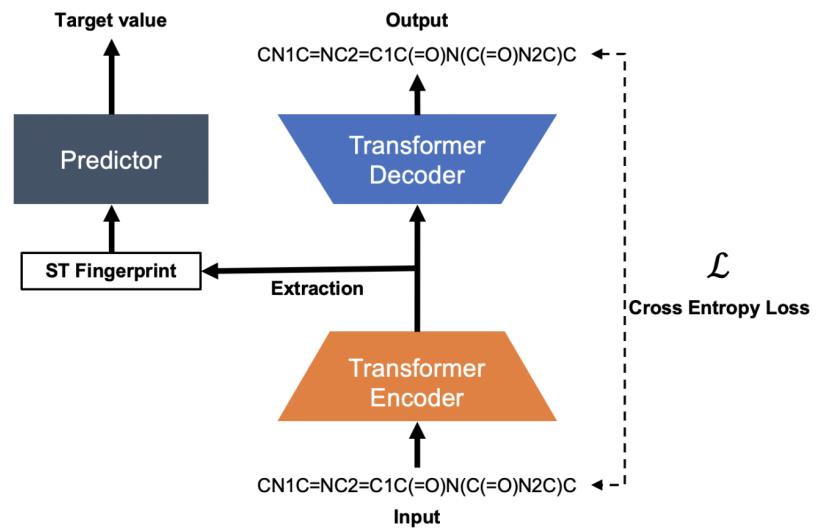
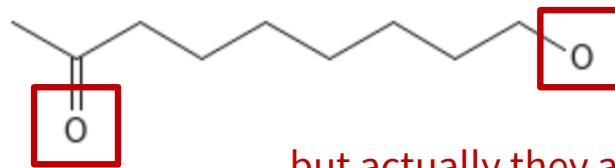


Illustration of SMILES-Transformer (Honda et al., 2019)

## LIMITATION OF SMILES-BASED MRL METHODS

- SMILES ARE 1D LINEARIZATION OF MOLECULE STRUCTURES, WHICH MAKES THEM HARD TO LEARN THE ORIGINAL STRUCTURAL INFORMATION OF MOLECULES



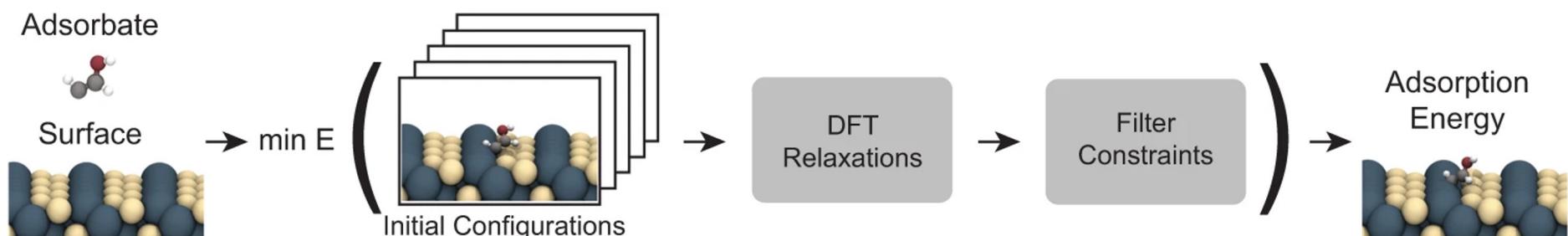
...but actually they are far from each other



These two O's are close in SMILES string...



# Multiple DFT Relaxations Needed to Determine Adsorption Energy



Lan, J., Palizhati, A., Shuaibi, M. et al., npj Comput Mater 9, 172 (2023)

- Adsorption energy is a key reactivity descriptor, which can be determined by identifying the most stable configuration among numerous possible adsorption configurations.
- The current approach requires evaluating numerous configurations to find the one with the lowest energy.
- Therefore, starting with a well-selected/system-specific candidate set is crucial for efficiently determining the adsorption energy.

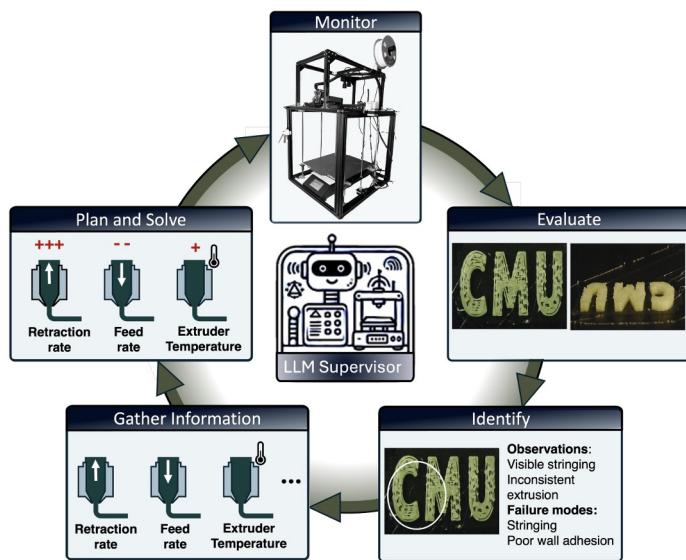
## Why LLM agent?



***“A language model is the knob on the door to the vast expanse of human knowledge and data”***

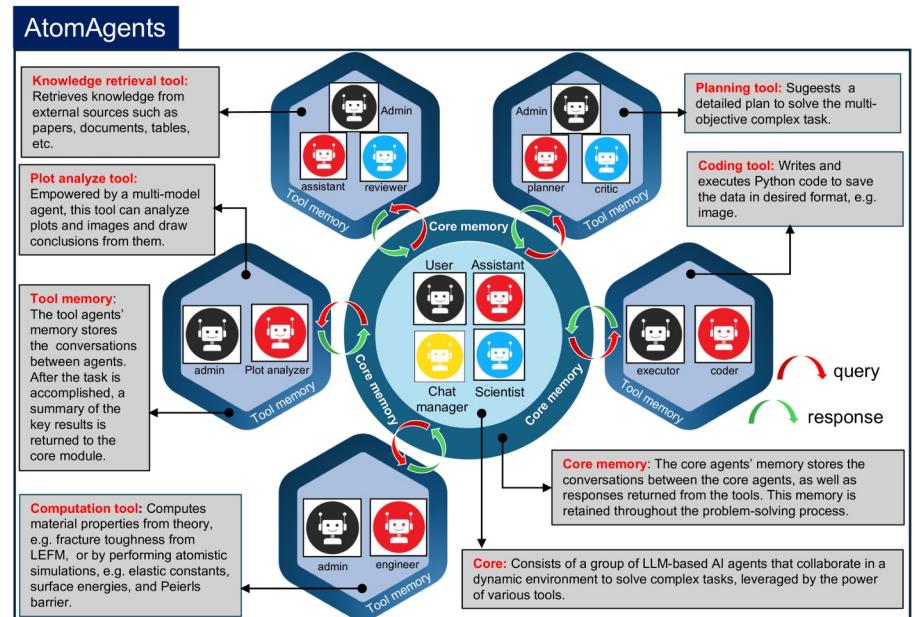
# LLM-Agent: Demonstrating Reasoning and Planning Capabilities Across Scientific Domains

## ● 3D Printer Control



Jadhav, Y. et al. arXiv. <https://arxiv.org/abs/2408.14307>, (2024)

## ● Alloy design/discovery

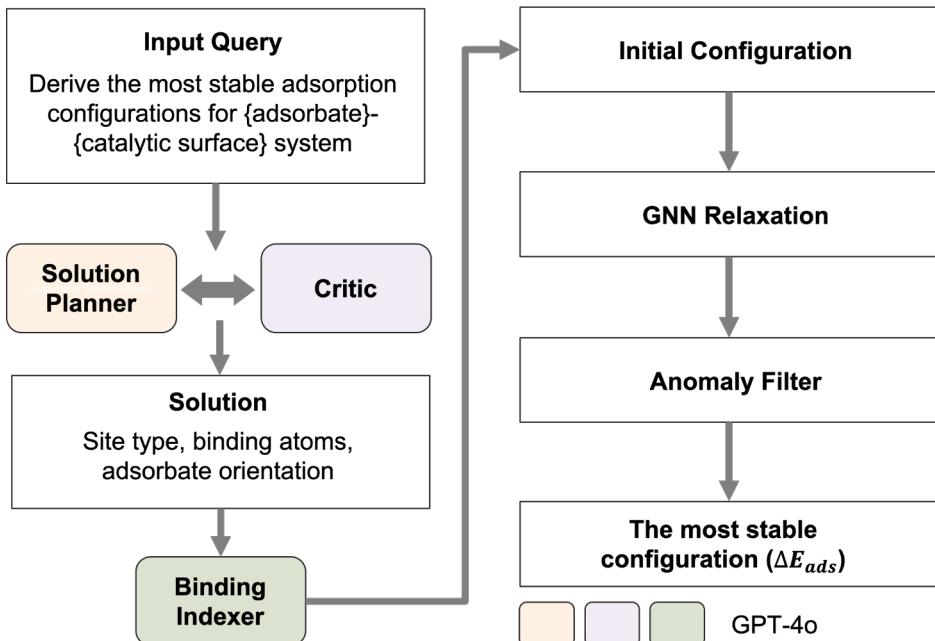


Ghafarollahi, A. et al. arXiv. <https://arxiv.org/abs/2407.10022>, (2024)

Carnegie  
Mellon  
University

● LLM-agent offers an excellent interface for interacting with human users and accessing existing text-based information

# Adsorb-Agent Framework



## Solution Planner

- Consider questions like
  - Are the adsorption sites providing the adsorbate with sufficient metal atoms to bond with?
  - Is the coordination number appropriate for the adsorbate, considering its bonding preferences and the type of metal surface?
  - Are the bonds between the adsorbate and the surface strong enough to ensure stability?
  - Does the adsorbate align in such a way that maximizes overlap with the metal's d-orbitals?
  - ...

## Critic

- Review whether derived solution, including site type, binding surface atoms, binding adsorbate atoms, orientation, is self-consistent

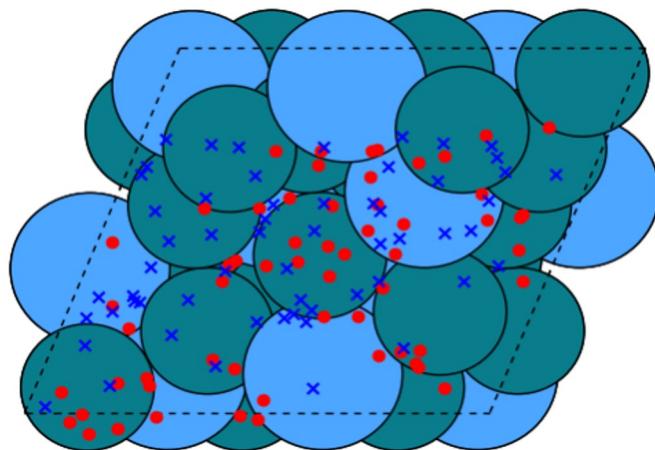
## Binding Indexer

- Designate the index of binding atoms in the adsorbate to place the adsorbate according to the specified binding atoms and orientation.

## Adsorb-Agent: Narrowing Down Initial Candidate Adsorption Sites

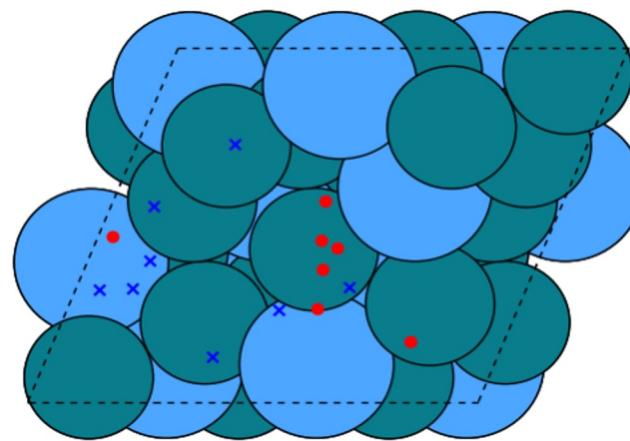
- Rh48Ta24 (1, 2, 1) surface example

Conventional



Random: 59 / Heuristics:  
58

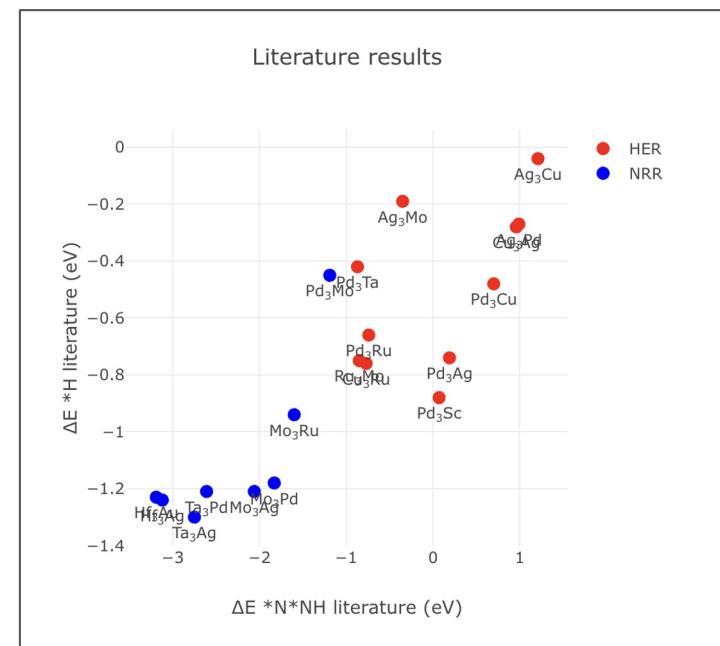
LLM-assisted



Random: 7 / Heuristics: 8  
Bridge site (Rh, Ta)

# Example case study

- SYSTEM
  - \*N\*NH on the CuPd<sub>3</sub> (1 1 1) surface
  - \*N\*NH on the Mo<sub>3</sub>Pd (1 1 1) surface
- RATIONALE
  - The **Nitrogen Reduction Reaction (NRR)** offers a sustainable alternative to the energy-intensive Haber-Bosch process for **producing ammonia, aiming to reduce greenhouse gas emissions and revolutionize fertilizer production.**
  - A major challenge with NRR is the **competing hydrogen evolution reaction (HER)**, which can interfere with the process.



# Adsorb-Agent: Identifying Lower Energy States with Fewer Initial Configurations

System	Placement Strategy	Number of initial sites	Min Energy [eV]
${}^*N^*NH\text{-CuPd}_3 (1\bar{1}\bar{1})$	Conventional algorithms	98	0.755
	Adsorb-Agent	22	0.733
${}^*N^*NH\text{-Mo}_3Pd (1\bar{1}\bar{1})$	Conventional algorithms	59	-0.954
	Adsorb-Agent	4	-1.367

# Solution (1) - CuPd3

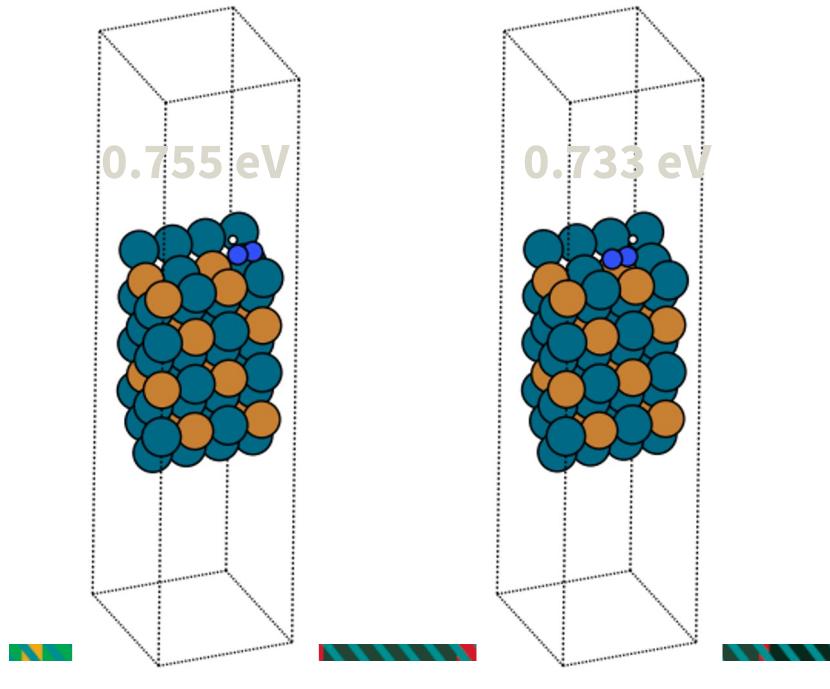
```
1  {'system': ['mp-1184119', '(1,1,1)', 0.063, None, '*N*NH', 'CuPd3'],
2   'initial_solution': {
3     'site_type': 'hollow',
4     'site_atoms': ['Cu', 'Pd', 'Pd'],
5     'num_site_atoms': 3,
6     'ads_bind_atoms': ['N', 'N', 'H'],
7     'orient': 'side-on',
8     'reasoning': 'The hollow site on the CuPd3 (1,1,1) surface provides a triatomic arrangement
9                  that maximizes the interaction between the adsorbate and the surface.
10                 The presence of both Cu and Pd atoms offers a mixed-metal environment
11                 that enhances charge transfer efficiency, allowing for robust bonding.
12                 The side-on orientation of the NNH adsorbate facilitates optimal overlap
13                 with the d-orbitals of the metal atoms, enhancing stability.
14                 The combination of Cu and Pd at the hollow site yields a balanced interaction,
15                 avoiding dominance by one metal type, and supports strong adsorption energy,
16                 indicative of a stable configuration.'}
17 }
```

# Visualization of Relaxed Adslabs

**\*N\*NH-CuPd3 (1 1 1)**

OCP-demo

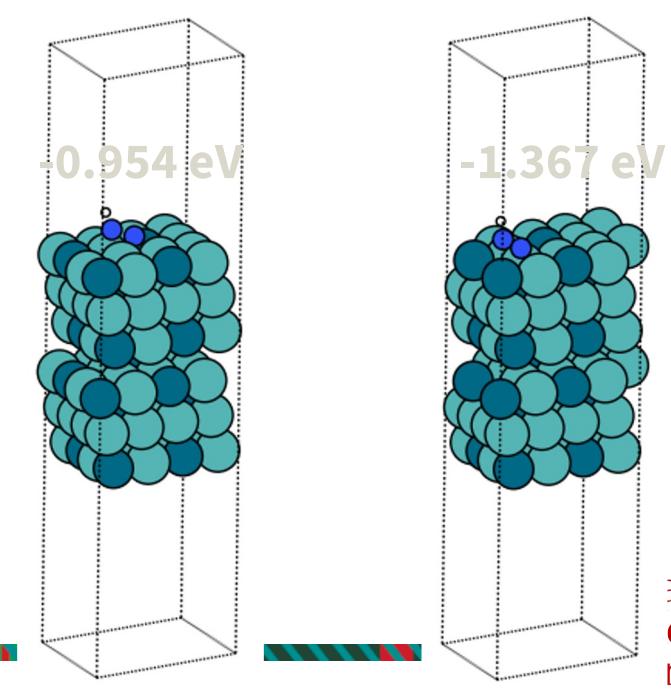
Adsorb-Agent



**\*N\*NH-Mo3Pd (1 1 1)**

OCP-demo

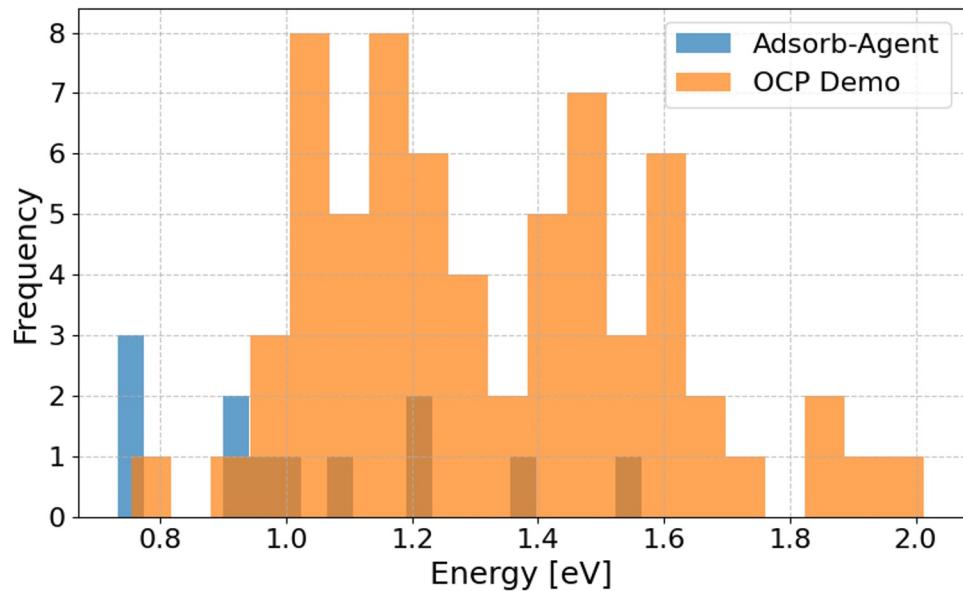
Adsorb-Agent



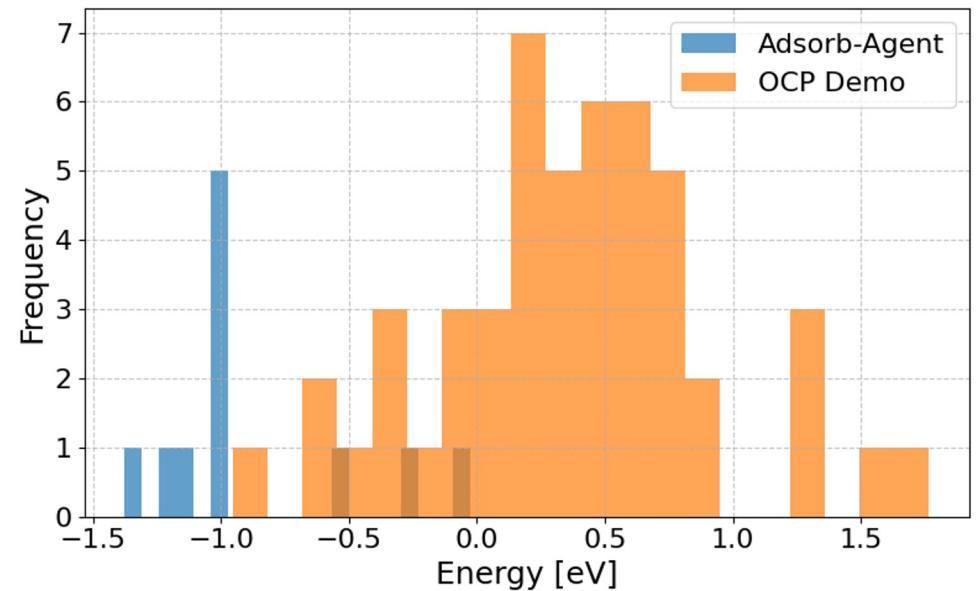
Carnegie  
 Mellon  
 University

# Visualization of Energy Distribution

**\*N\*NH-CuPd3 (1 1 1)**

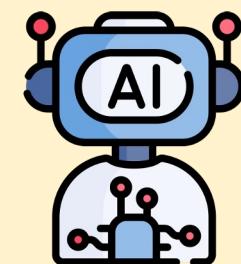


**\*N\*NH-Mo3Pd (1 1 1)**



# Outlook

## Interactive & Accessible Catalyst Design



Catalyst-Agent

### Dataset & model & API

Open Catalyst demo

Select a catalyst material and an adsorbate for adsorption energy calculations.

Step 1. Specify a catalyst bulk material  
Search for an element, chemical formula, or MNiO from the Materials Project.  
Search e.g. Cu or m-niO  
Note: we currently support ~15k bulk materials and are working on expanding this list. If you don't see your favorite material, please let us know.  
NEXT

Step 2. Select a surface  
The Materials Project  
Harnessing the power of supercomputing and state-of-the-art methods, the Materials Project provides open web-based access to computed information on known and predicted materials as well as powerful analysis tools to inspire and design novel materials.  
Login or Register See a Random Material Browse Apps

### Literature & Web



SerpApi Documentation Integrations Features Pricing Use cases Team FAQ Contact us

Google Search API

### Atomic simulation tools

**pymatgen**

**ASE**

**SUNCAT-Center/CatKit**

General purpose tools for high-throughput catalysis

A 6 Contributors G 11 Used by ⭐ 83 Stars ⚡ 50 Forks

### Gen AI

**Iantunes/CrystaLLM**

A Large Language Model of the CIF for Crystal Structure Generation

A 1 Contributor I 2 Issues ⭐ 48 Stars ⚡ 10 Forks

**AdeeshKolluru/AdsorbDiff**

[ICML'24] Adsorbate Placement via Conditional Denoising Diffusion

A 1 Contributor I 0 Issues ⭐ 8 Stars ⚡ 0 Forks

facebookresearch/crystal-text-llm

Large language models to generate stable crystals.

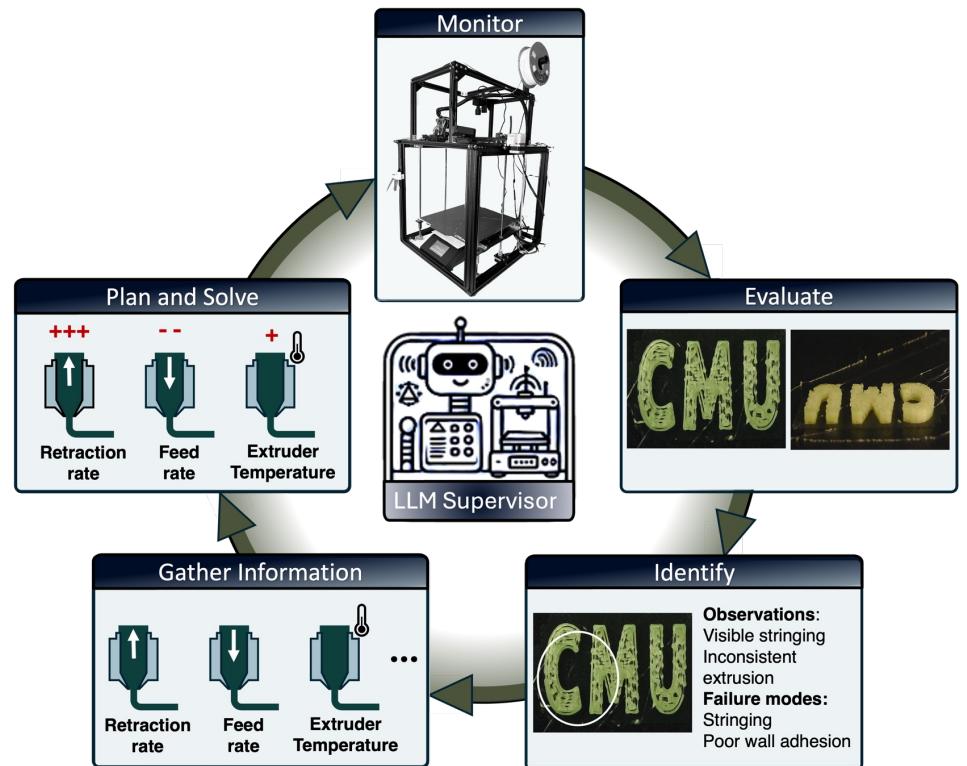
# Problem Statement:

- How can we do it with least amount of data generation?
- How can we optimize the print quality and optimize over unseen/unknown parameters?
- How can we make it super generalizable (printer setup, part geometry, defects) ?
- What other functionality can we add to track defects?



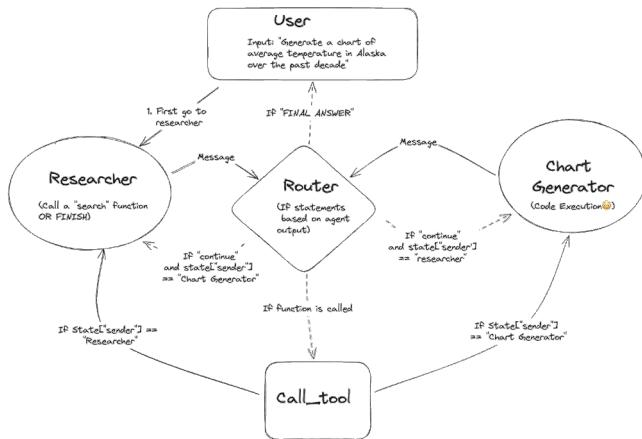
# Proposed Solution:

- **How can we do it with least amount of data generation?**
  - Multi-Modal LLMs have been trained on large data from the “wild”. It might contain relevant information for printer defects. Eliminates the need for data generation.
- **How can we optimize the print quality and optimize over trained parameters?**
  - Leverage existing knowledge of the LLMs that already contain defects and solutions, allowing to optimize print quality without retraining.
- **How can we make it super generalizable (printer setup, part geometry, defects) ?**
  - Use the adaptability of LLMs, which can generalize across various setups, geometries, and defects due to their diverse training data.
  - Provide printer manual to the LLM to reduce trial and error.
- **What other functionality can we add to track defects?**
  - LLM-generated commentary on the printing process, enhances transparency, build trust, and improve tracking of defects in parts. No need for destructive testing and sampling.



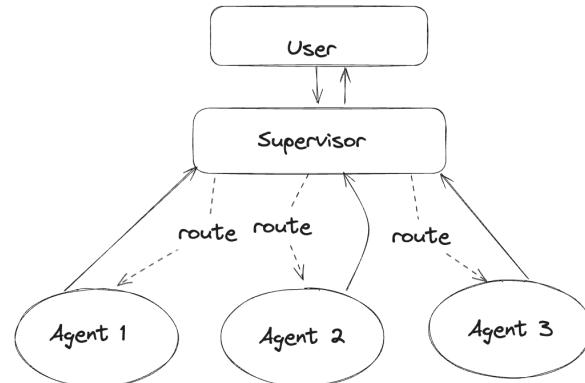
# Multi Agent Frameworks:

## Collaborative agents



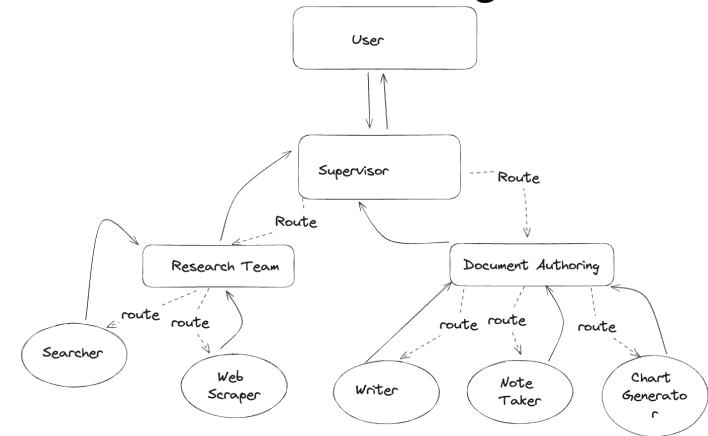
- Uses "divide-and-conquer" approach.
- Divides the task in multiple smaller problems.
- Basic router determines which agent to invoke next.
- Good for simple problems with specific workflow.

## Agent Supervisor



- Similar to divide and conquer.
- The supervisor is an LLM which reasons over the information from individual agents and routes to the best “expert”.
- The supervisor keeps track of shared dictionary which all agents have access to.

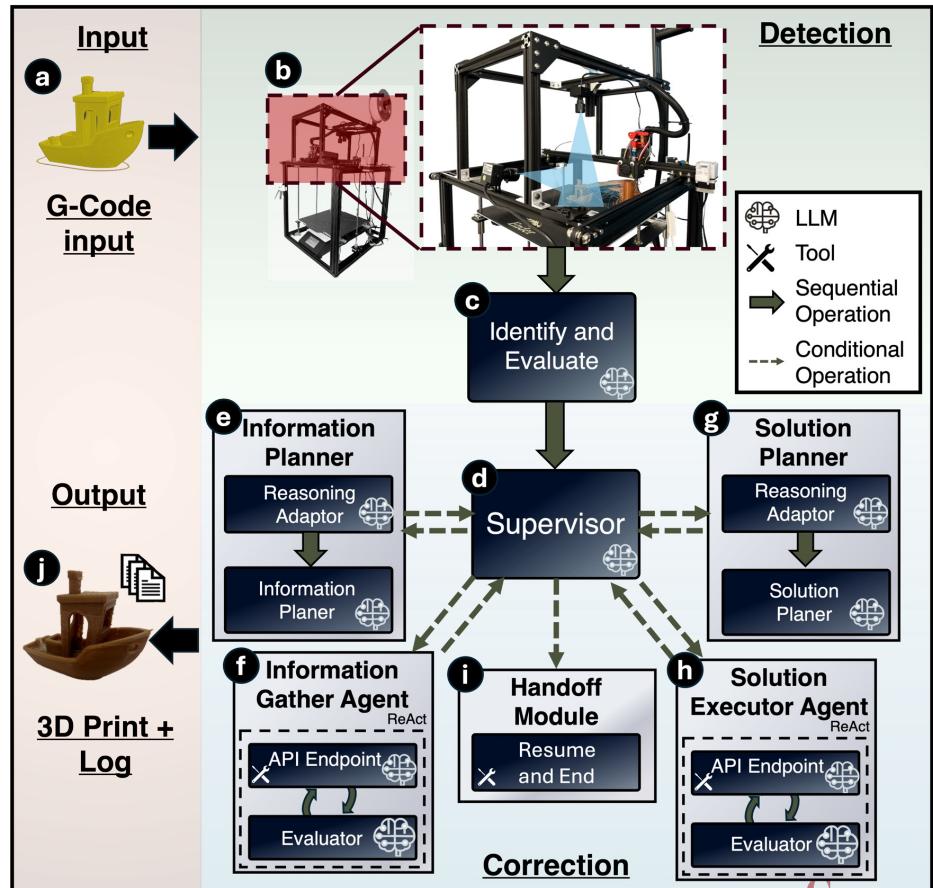
## Hierarchical agents



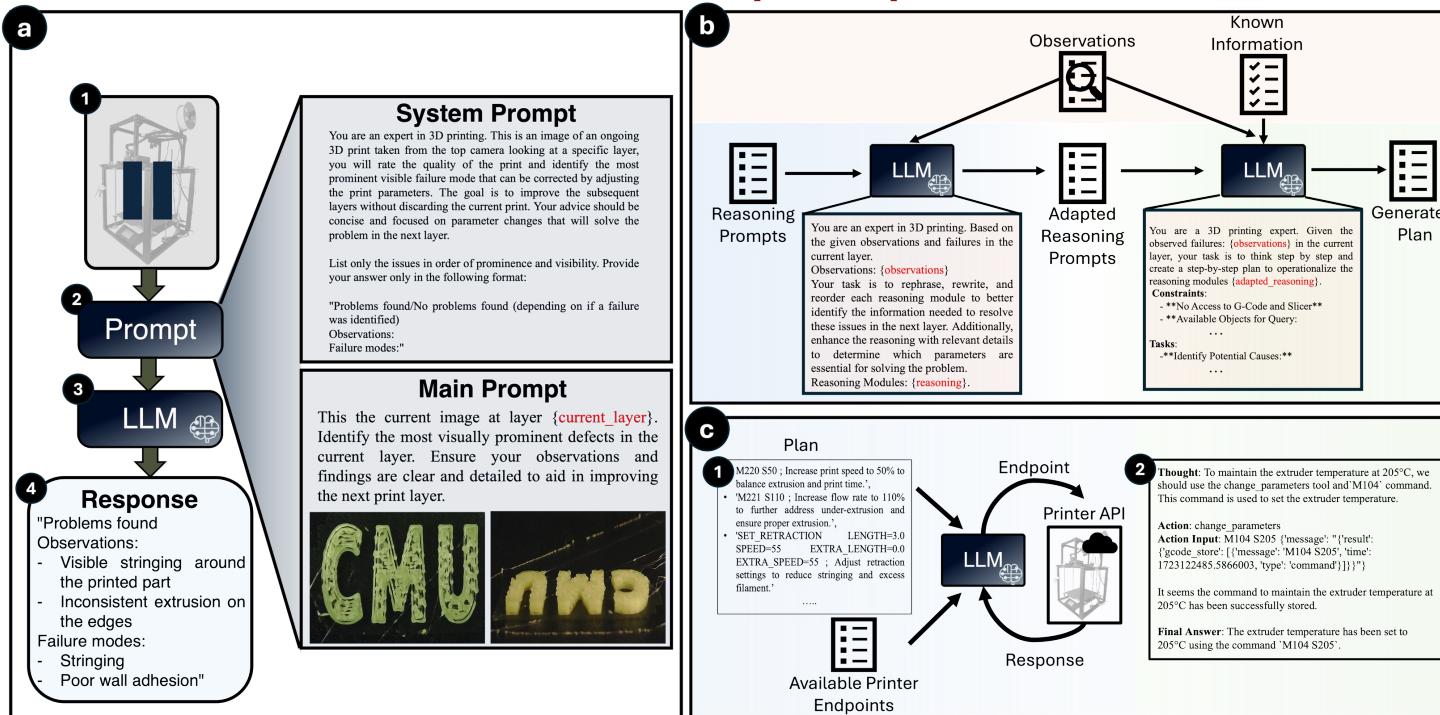
- Uses hierarchical supervisors.
- Task is routed to a “team” based on information available and then routed to individual workers/agents.
- Important when a task needs to be iteratively refined.

# Framework Overview:

- Uses agent supervisor framework.
- Two frame mounted cameras take images after each layer is printed.
- ChatGPT 4o vision module makes observations and identifies defects.
- Depending on defects the supervisor routes to the best agent to solve the task.
- Modular framework allows for adding more modules/agents, supervisor if needed.
- 7 Modules: 2 planning, 3 executors, 1 supervisor and one “user”.



# Prompt Pipeline: Overview



- The detection prompt just talks about known information like printer state, layer number and what object we are trying to print.
- The reasoning module gets adapted to the current problem.
- Known information / printer documentation is provided to planner and executor. Narrows down iterations for ReAct.
- Planner generates a plan to gather information about the current status of the printer and generate a solution plan.
- The executor uses generalizable “tools” to execute the plan.

# Prompt Pipeline: Bad Agent!!!

> Finished chain.

Thought: The response indicates that the printer is in the middle of an SD print and has processed 59961 bytes out of 108154 bytes. This suggests that the print is still ongoing or paused but not finished. Since the printer status is "Ready" but the SD card is busy, I might need to reset the printer to clear the busy state and then resume the print.

Action:

```
```
{
  "action": "change_parameters",
  "action_input": {
    "printer_url": "http://172.26.79.51",
    "gcode_command": "M112"
  }
}
```
{"message": {"result": {"gcode_store": [{"message": '//',
  "Klipper state: Shutdown', 'time': 1721335461.195151, 'type': 'response'}]}}}
```

# Prompt Pipeline: Reasoning

- Best way to talk to LLM is to speak its language.
- Provide a set of basic general reasoning questions and ask LLM to select, rephrase, reorder and add questions based on the problem statement.
- Reasoning provides a structure to the LLM output and makes it produce more logical and relevant outputs.

## Information Reasoning:

1. What are the possible reasons for the failure?,
2. How to solve the problem in the next layer without human intervention?,
3. What information do you already know?,
4. What additional information do you need to solve the task effectively and autonomously?,
5. What information can be acquired from the basic 3D printer without human intervention to identify the cause of failure?,
6. What information can be required human intervention?,
7. What are the environmental conditions?,
8. What are the controllable parameters that can be changed to solve the failure in the next layer?
9. Based on the observations what are the failure modes you can identify in the ongoing print?
10. Based on the failure modes and observations, which issues are most likely to cause significant issues to the print quality and integrity?
11. What printer parameters is most responsible for these observations and issues?
12. What are the parameters which when changed will have the most impact on the quality of the 3D print.

## Solution Reasoning:

- "1. What are the possible reasons for the observations and failure in the last layer?",
- "2. Analyze the information and data you already have regarding the documentation, print parameters, and material properties during the failure.",
- "3. Given the current print parameters [given information known] as observed during the last layer, what are the best parameters to adjust to solve the failure in the next layer without human intervention?",
- "4. What are the controllable parameters that can be changed to solve the failure in the next layer?",
- "5. How would you solve the problems without human intervention?",
- "6. Break down the problem into smaller parts and identify the most significant contributing factors to the failure. Consider aspects such as material properties, print parameters, and potential G-code commands to use.",
- "7. What solution does a human need to perform to solve the problem in the next layer without discarding the print?",
- "8. What solution does the machine need to perform to solve the problem in the next layer without discarding the print?",
- "9. Provide step-by-step instructions to solve the problem in the next layer without discarding the print or using human intervention.",
- "10. Given the commands specific to the printer, what commands should be used to solve the problem in the next layer without discarding the print?",
- "11. What parameters, when changed, will impact the quality of the 3D print?",
- "12. What should be the correct values for the parameter changes?",
- "13. Are the parameter values like bed temperature, extruder temperature, speed factor, extrusion factor, etc., correct for the given filament type?",
- "14. How can you optimize the cooling settings to improve print quality?",
- "15. How can adjusting the retraction settings help solve the observed failures?",
- "16. What adjustments to the layer height or width could improve print consistency?",
- "17. Based on the observations what is failure modes?",
- "18. Given current parameters values and information known what would be the best values for them so that the problem is solved in the next segment/layer?",
- "19. Are there other parameters to consider? what are those? what should be the values?"

# Prompt Pipeline: Executor

- Executor uses ReAct framework, where if irrelevant endpoint is called the LLM judges if the response has a valid information and iterates over it.
- Tools are basic python functions, which takes input arguments provided by the LLM and outputs the results.
- Ideally in previous attempts the LLM generates its own set of tools to create a skill library. Was too expensive!!!
- Given a list of “tools” llm decides which tool is most suited and which input will give the desired results.
- The tool initially requires the printer's IP and endpoint as input. While it can accept multiple arguments, a bug in LangChain with GPT-4.0 causes invalid content to be generated.
- Always give LLM an identity: You are an expert in \_\_\_\_ or Imagine you are a robot with extruder for hands.

## Information Gathering Executor Agent:

```
You are tasked with querying a 3D printer running the Moonraker API for requested information.\n\nYou have access to the following tools:\n\n{tools}\n\nHere is the information known to you {information_known}.\n\nNOTE: The current maximum print speed is (speed*speed_factor) found in gcode_move.\n\nHere are the printer and Gcode objects you can query {printer_obj}, use your experience in determining which object to call to get current information.\n\nYou can write additional python code to interpret the information or process information from the printer.\n\nIf you dont find the required info try semantically similar words.\n\nUse the following format:\n\nQuestion: the input question you must answer\nThought: you should always think about what to do\nAction: the action to take, should be one of [{tool_names}]\nAction Input: the input to the action\nObservation: the result of the action\n...\n(this Thought/Action/Action Input/Observation can repeat N times)\nThought: I now know the final answer\nFinal Answer: the final answer to the original input question\n\nBegin!\n\nQuestion: {input}\nThought: {agent_scratchpad}
```

## Solution Executor Agent:

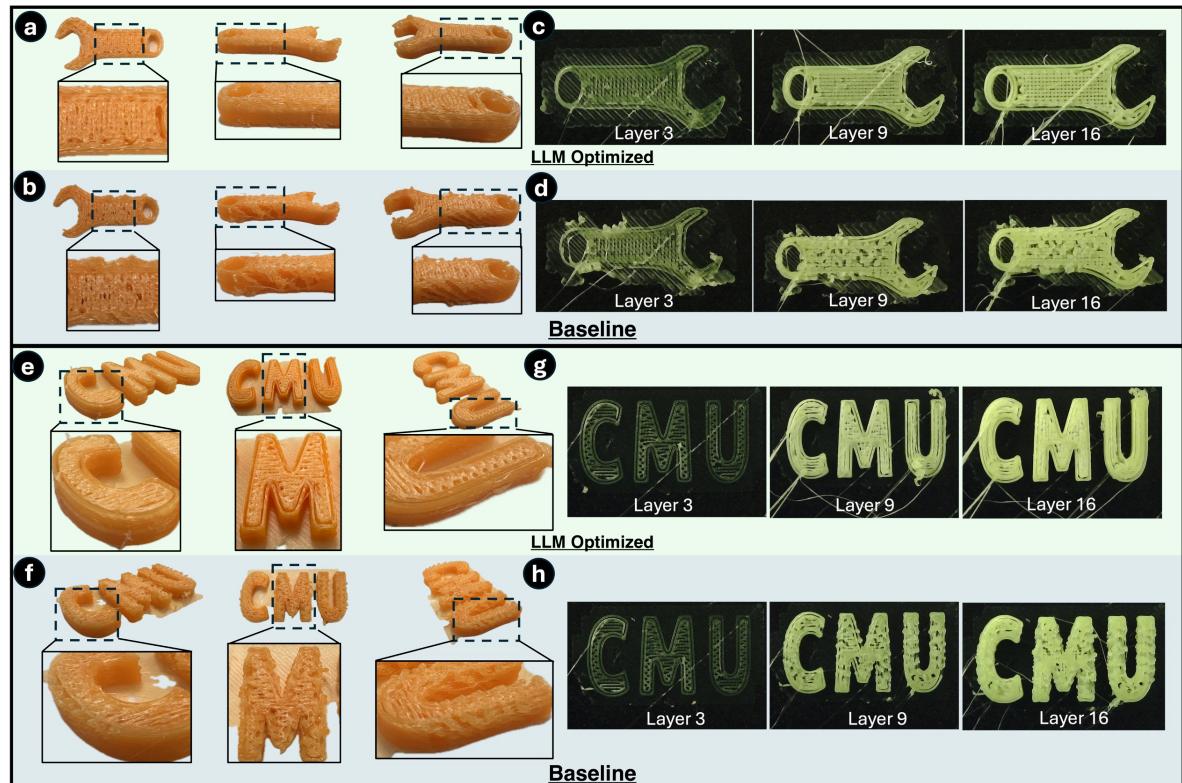
```
You are tasked with querying a 3D printer running the Moonraker API for requested information.\n\nYou have access to the following tools:\n\n{tools}\n\nHere is the information known to you {information_known}.\n\nHere are the printer parameters you can modify with gcode commands: {gcode_commands}, use your experience in determining which command to execute to implement the solution plan.\n\nIf you dont find the required command to run try semantically similar words.\n\nUse the following format:\n\nQuestion: the input question you must answer\nThought: you should always think about what to do\nAction: the action to take, should be one of [{tool_names}]\nAction Input: the input to the action\nObservation: the result of the action\n...\n(this Thought/Action/Action Input/Observation can repeat N times)\nThought: I now know the final answer\nFinal Answer: the final answer to the original input question\n\nBegin!\n\nQuestion: {input}\nThought: {agent_scratchpad}
```

```
@tool\ndef query_printer(object_name:str) -> dict:\n    """\n        Query the Moonraker API to get state of object_name and return the response data.\n\n    Args:\n        input: object_name.\n\n    Returns:\n        dict: The data returned by the Moonraker API.\n    """\n    url = f'{printer_url}/printer/objects/query?{object_name}'\n    url = f'http://172.26.79.51/printer/objects/query?{object_name}'\n    # Check if the request was successful\n    try:\n        response = requests.get(url)\n        # response.raise_for_status()\n        out = response.json()\n        print("Output ", str(out))\n        return out\n    except requests.exceptions.HTTPError as e:\n        return {"error": str(e)}
```

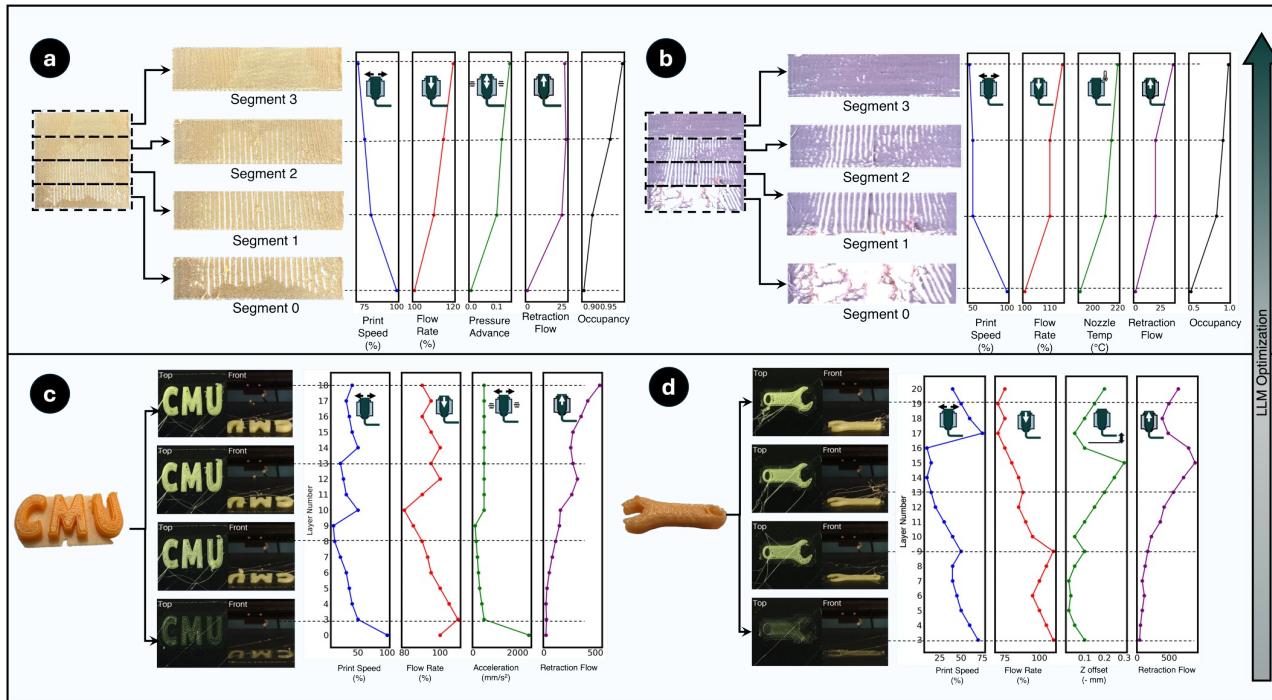
```
@tool\ndef query_gCode(gcode_command:str) -> dict:\n    """\n        Query the Moonraker API to get current retraction rate.\n\n    Args:\n        input: Dict containing the printer_url and object_name.\n\n    Returns:\n        dict: The data returned by the Moonraker API.\n    """\n    endpoint = f'http://172.26.79.51/printer/gcode/script'\n\n    # Data payload for the POST request\n    data = {\n        "script": gcode_command\n    }\n\n    # Check if the request was successful\n    response = requests.post(endpoint, json=data)\n    time.sleep(3)\n    get_url=f"http://172.26.79.51/server/gcode_store?count=1"\n    out = requests.get(get_url)\n    out=out.json()\n    # print(out)\n\n    return {"message": str(out)}
```

# Results: Multi Layer prints

- Randomly selected STLs were sliced in PRUSA slicer. Only change was adding PAUSE macro after each layer.
- No default parameters for 1mm nozzle, so random value was selected within the generally used range of parameters.
- Multi layer prints with PLA show significant qualitative improvement with LLM optimizations.
- Each layer is better than the previous.
- Smaller defects such as inconsistent extrusions is solved, also other minor surface defects are solved before it shows.
- A purge tower is required to remove oozing and priming the nozzle.



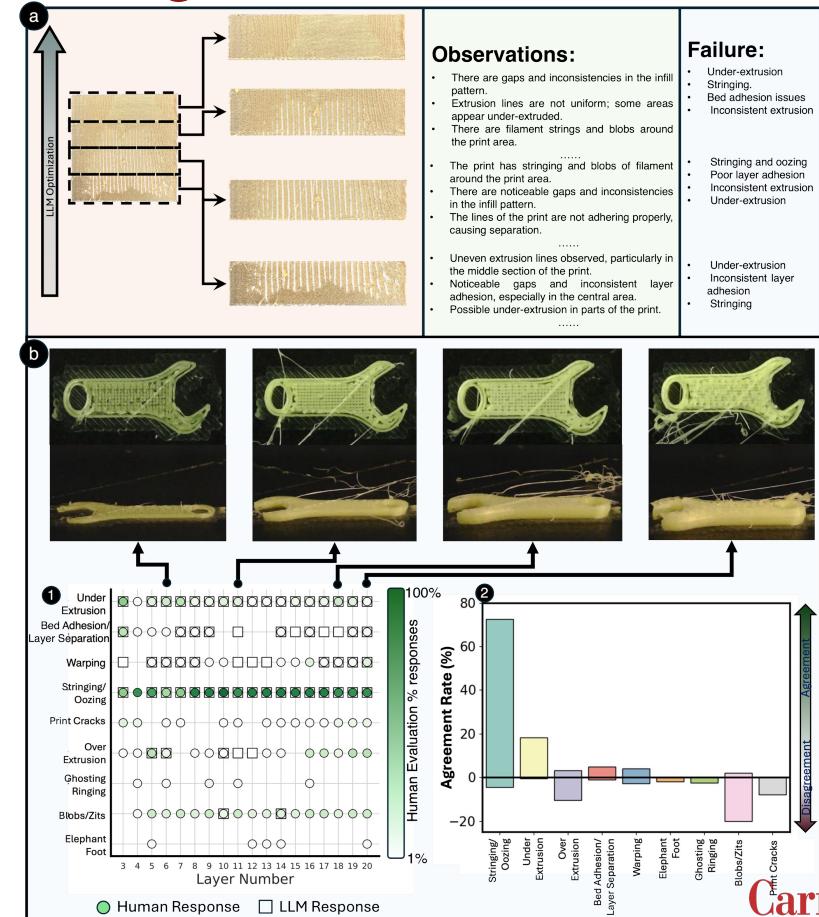
# Results:

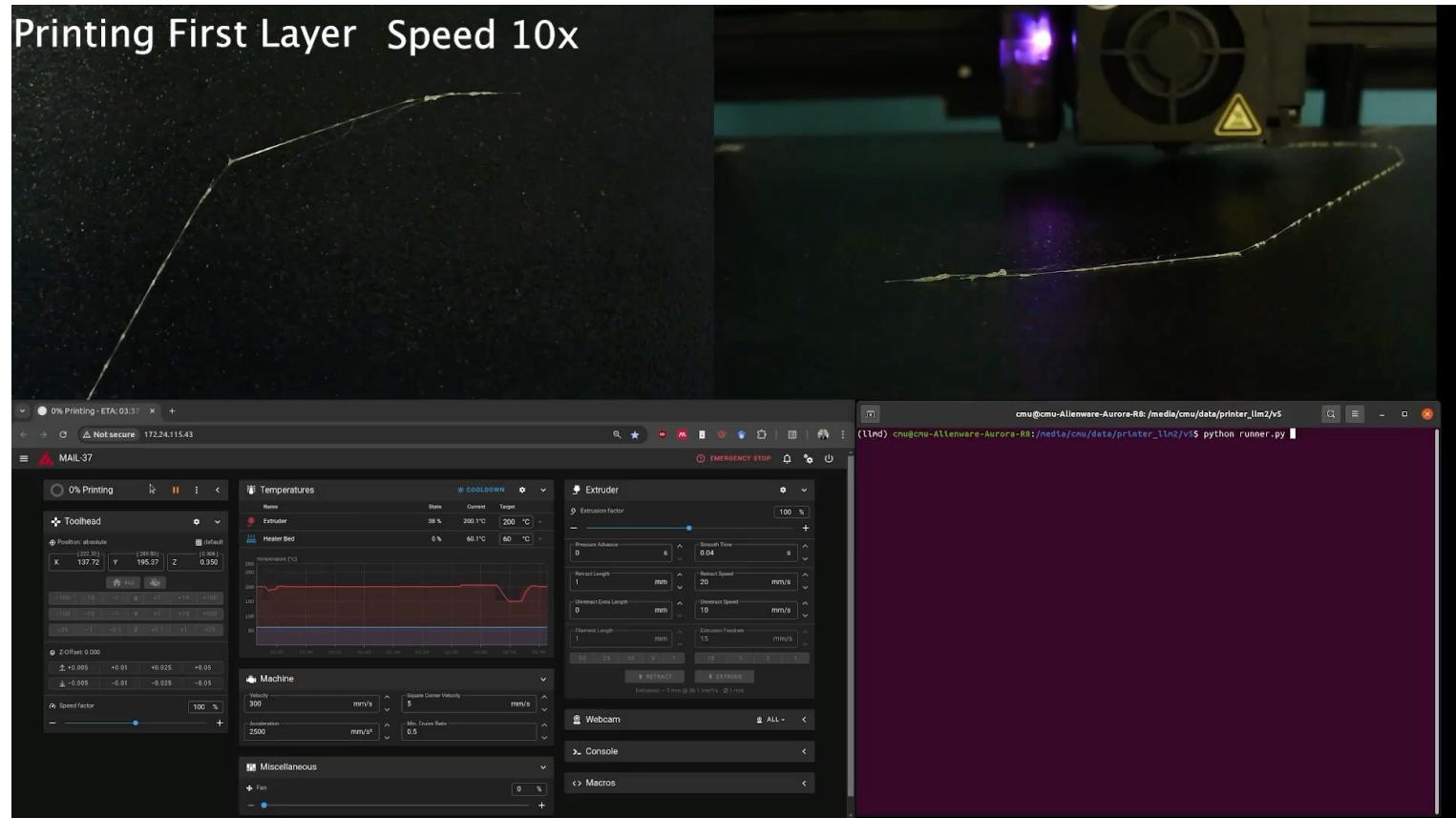


- We show that our framework is material and design agnostic, and works on multi part prints.
- The LLM shows optimizing behaviour on selected parameters.
- It optimizes other parameters such as fan speed bed temperature too, but most of them are just iterated over a couple of layers.
- For single layer prints we used occupancy metric. As observed it reach up to 99% occupancy withing 3 iterations.

# Results: Can it beat Engineers?

- Based on human evaluation of the print layers done by engineers (most of you), it can.
- The errors introduced were:
  - Stringing
  - Inconsistent extrusion
  - Blobs and zits.
- Most of the evaluations were on point with some suggesting unseen defects such as elephant foot and print cracks.
- It has a good memory and localization to ignore defects in previous layers.
- Agreement metric shows how much humans and LLM agree on for a specific defect.





## Questions?

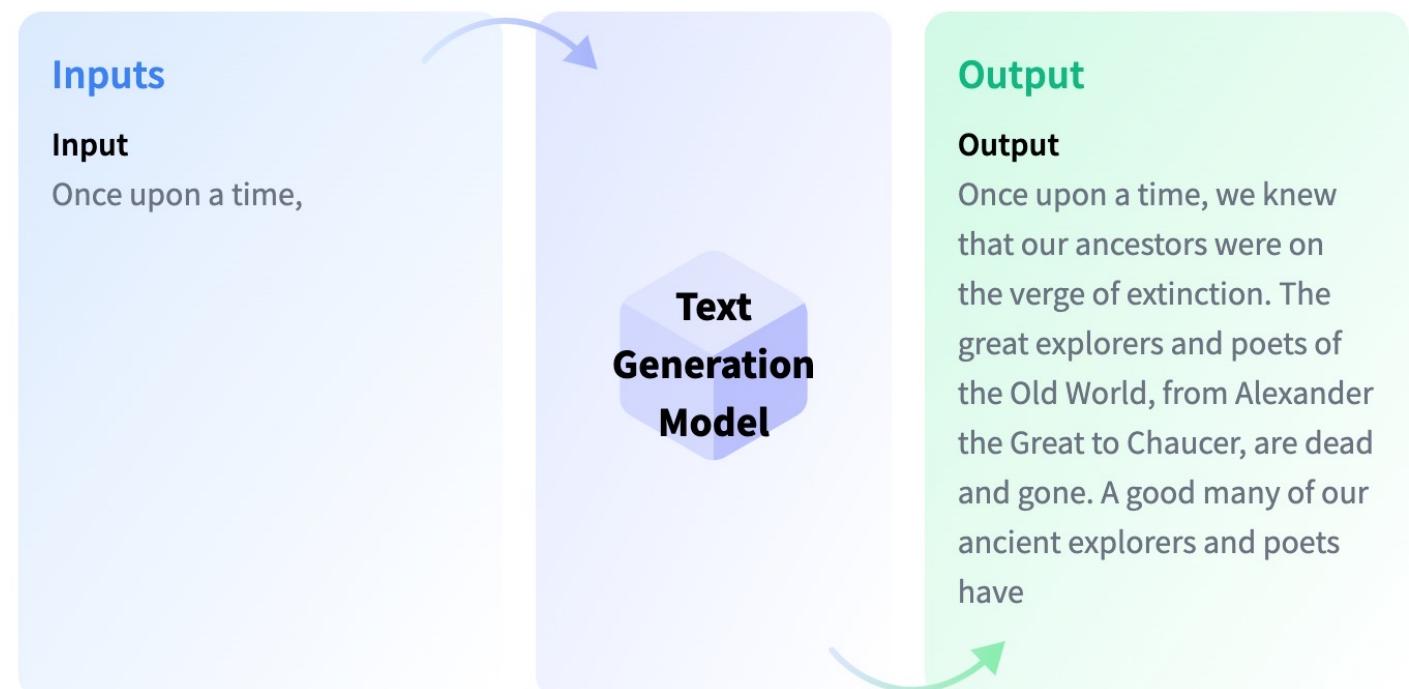
# Text Generation

---

## Natural Language Generation (NLG)

- Language modeling
- Conditional language modeling

Next word prediction



# Text Generation

⚡ **Text Generation demo**  
using gpt2

Text Generation

Examples

Once upon a time,

Compute

Computation time on cpu: 1.1964 s

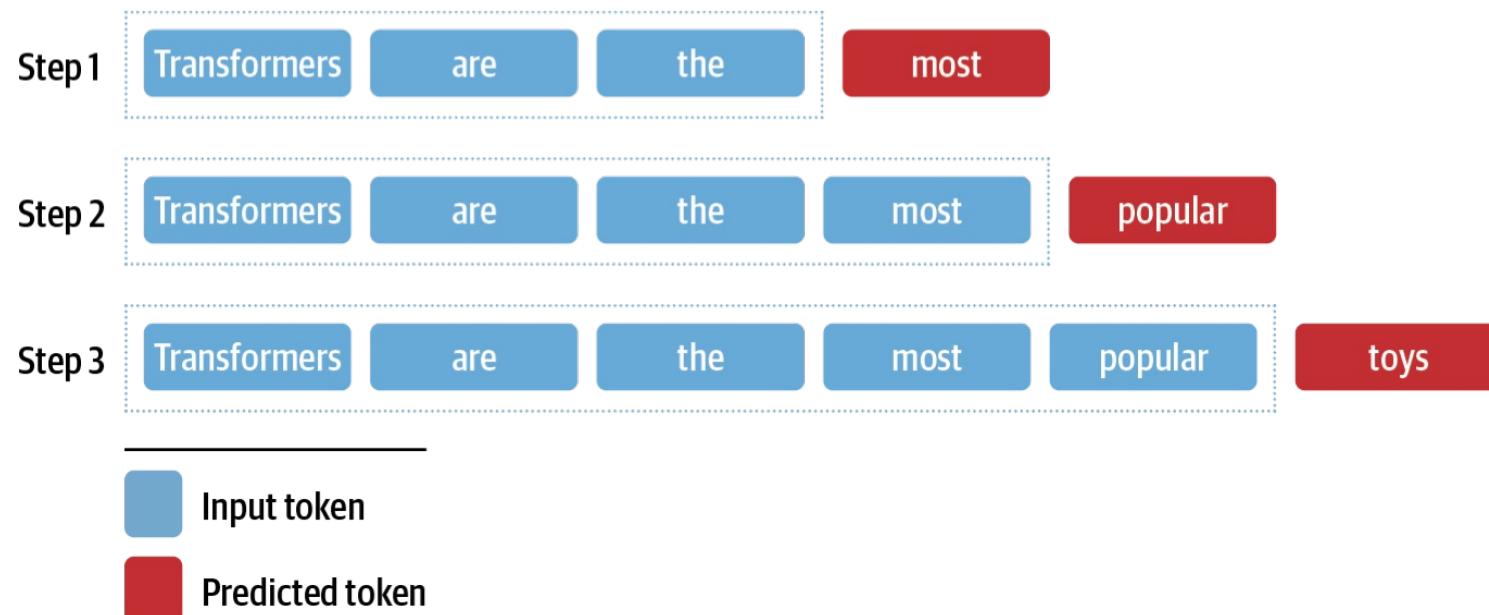
Once upon a time, there was no such thing as a real-world version. The original Japanese version of Samurai Master, for instance, showed the main character as a giant insect with two eyes that could be seen directly out of the perspective lens,

<https://huggingface.co/tasks/text-generation>

Carnegie  
Mellon  
University

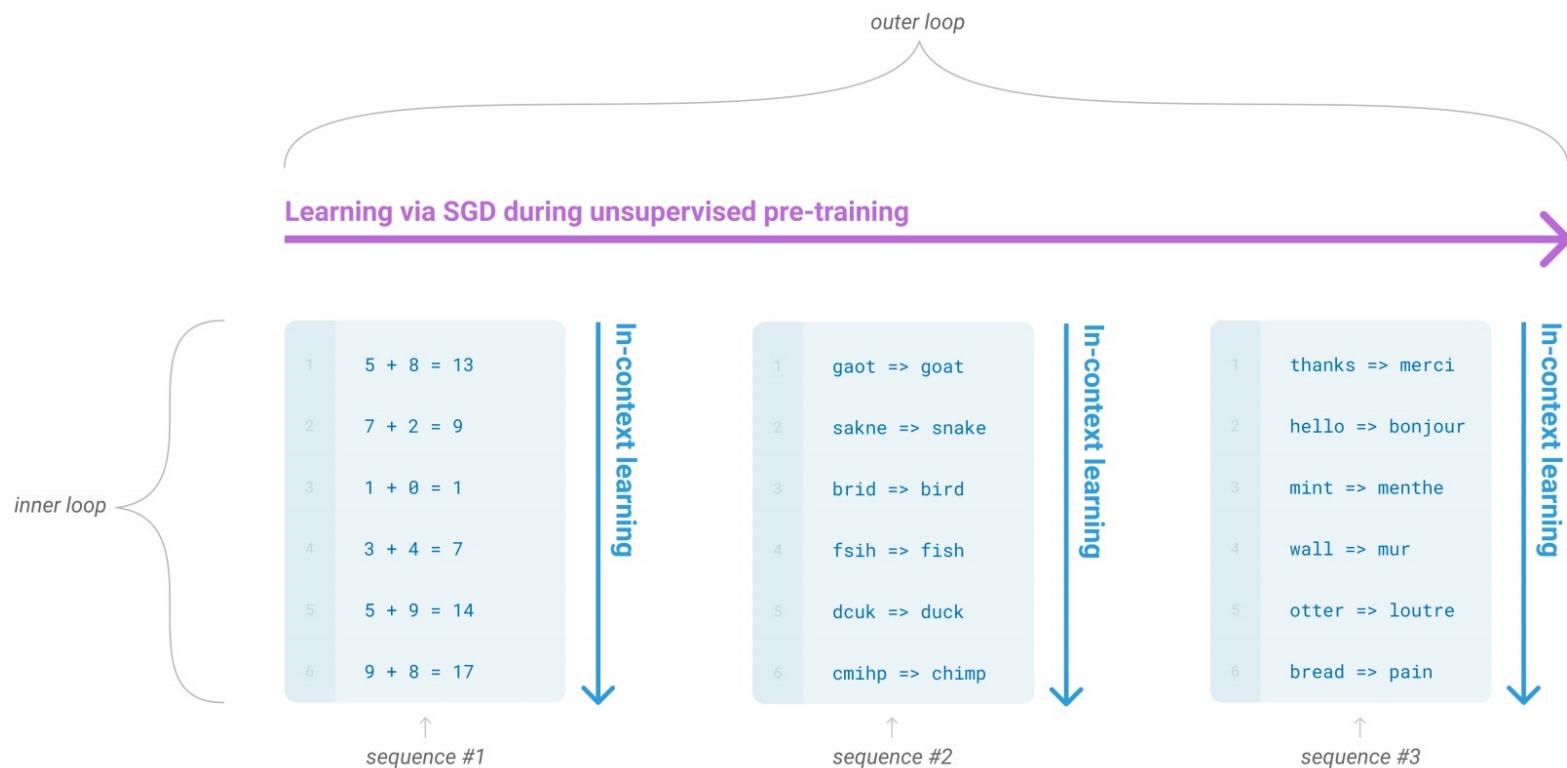
# Natural Language Generation (NLG)

The Challenge with Generating Coherent Text: Next Word Prediction



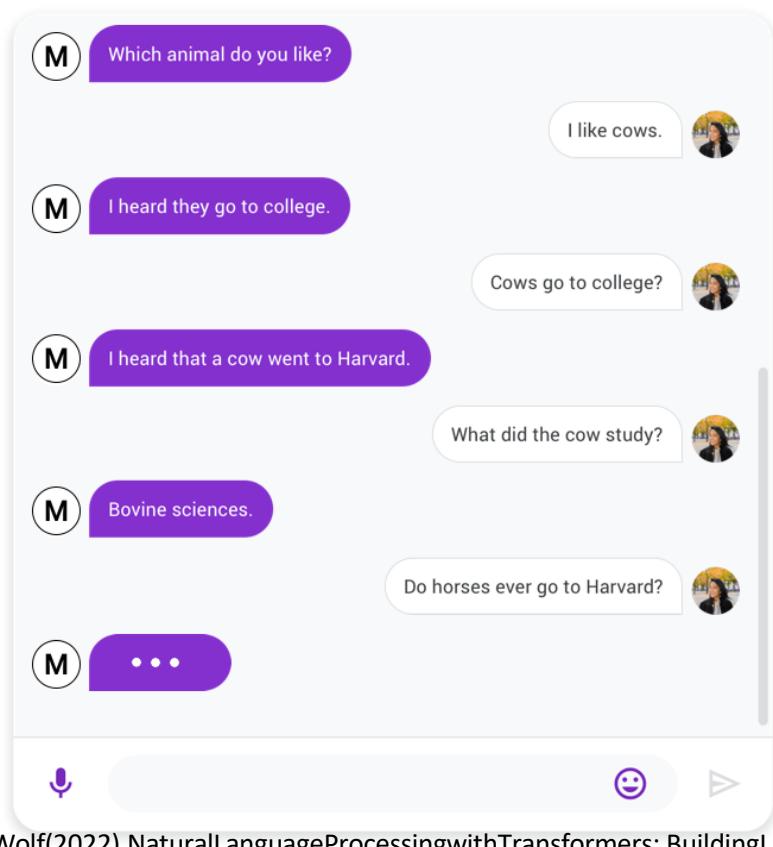
Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

# Text Generation



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

# Text Generation



Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

# Text Generation: Decoding Algorithm

---

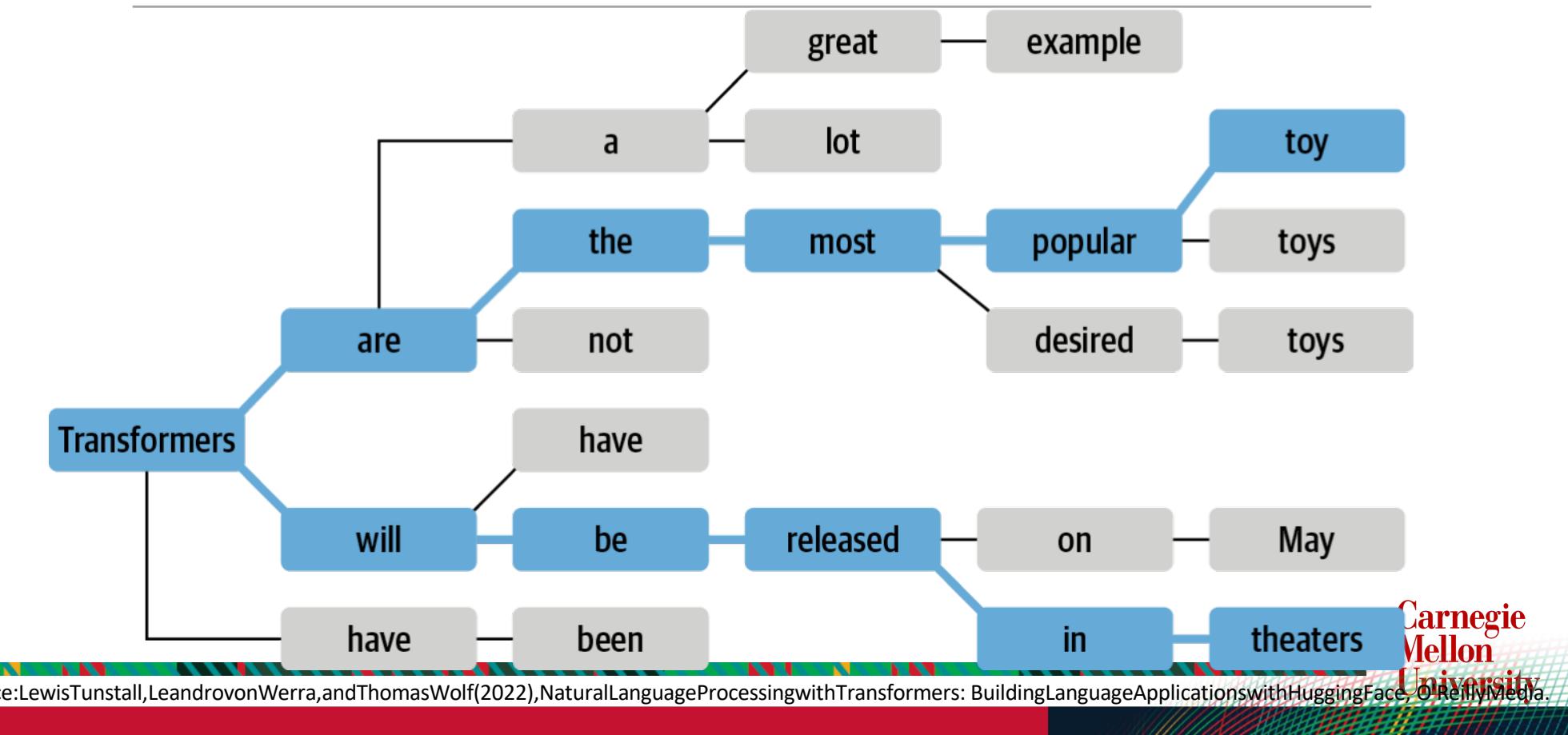
- Greedy Search Decoding
- Beam Search Decoding
- Sampling Methods
- Top-k and Nucleus Sampling

# Text Generation: Greedy Search Decoding

Input	Choice 1	Choice 2	Choice 3	Choice 4	Choice 5
0 Transformers are the	most (8.53%)	only (4.96%)	best (4.65%)	Transformers (4.37%)	ultimate (2.16%)
1 Transformers are the most	popular (16.78%)	powerful (5.37%)	common (4.96%)	famous (3.72%)	successful (3.20%)
2 Transformers are the most popular	toy (10.63%)	toys (7.23%)	Transformers (6.60%)	of (5.46%)	and (3.76%)
3 Transformers are the most popular toy	line (34.38%)	in (18.20%)	of (11.71%)	brand (6.10%)	line (2.69%)
4 Transformers are the most popular toy line	in (46.28%)	of (15.09%)	, (4.94%)	on (4.40%)	ever (2.72%)
5 Transformers are the most popular toy line in	the (65.99%)	history (12.42%)	America (6.91%)	Japan (2.44%)	North (1.40%)
6 Transformers are the most popular toy line in the	world (69.26%)	United (4.55%)	history (4.29%)	US (4.23%)	U (2.30%)
7 Transformers are the most popular toy line in the world	, (39.73%)	. (30.64%)	and (9.87%)	with (2.32%)	today (1.74%)

Source: Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.

# Text Generation: Beam Search Decoding



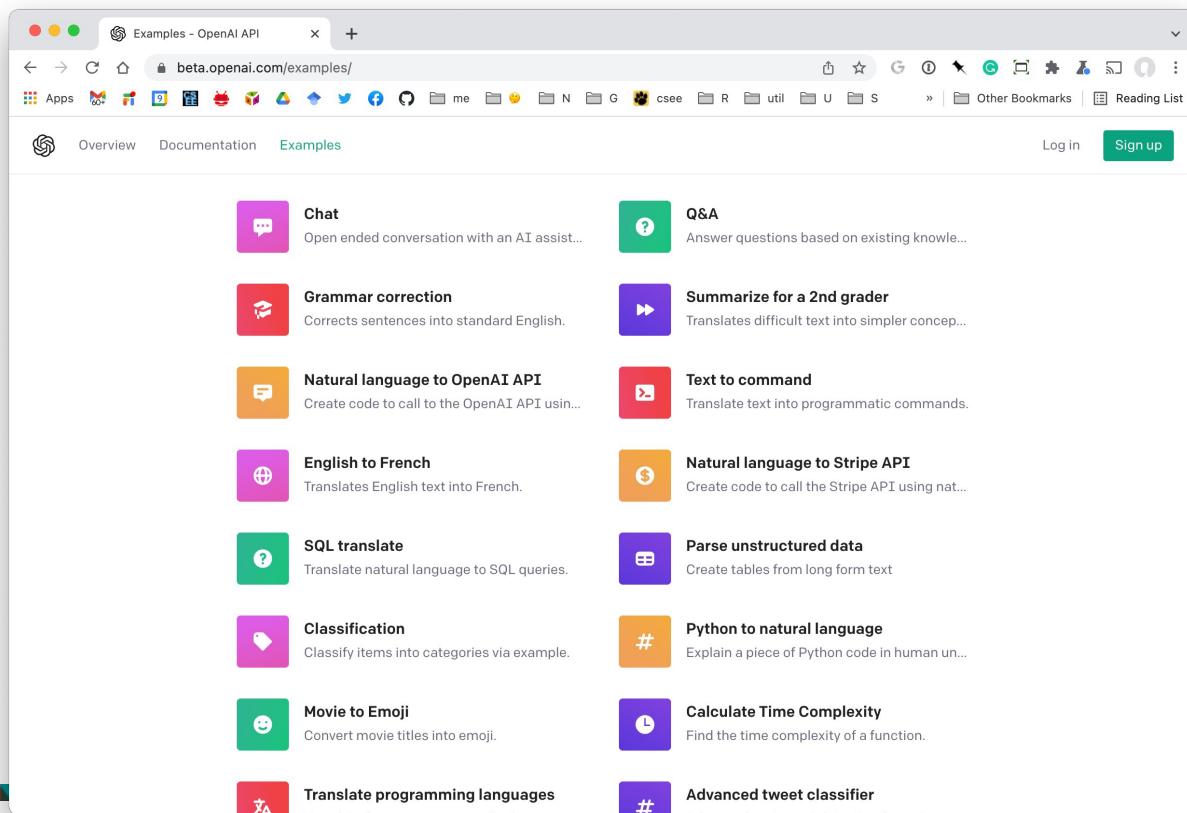
# Hugging Face models

The screenshot shows a web browser window displaying the Hugging Face website at [huggingface.co/models](https://huggingface.co/models). The page features a navigation bar with links for Models, Datasets, Spaces, Resources, Solutions, Pricing, Log In, and Sign Up. On the left, there's a sidebar with sections for Tasks (Fill-Mask, Question Answering, Summarization, Table Question Answering, Text Classification, Text Generation, Text2Text Generation, Token Classification, Translation, Zero-Shot Classification, Sentence Similarity) and Libraries (PyTorch, TensorFlow, JAX). The main content area shows a list of models, each with a thumbnail, name, task, updated date, file size, and download count. The first few models listed are:

- bert-base-uncased
- sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2
- roberta-base
- distilbert-base-uncased
- gpt2

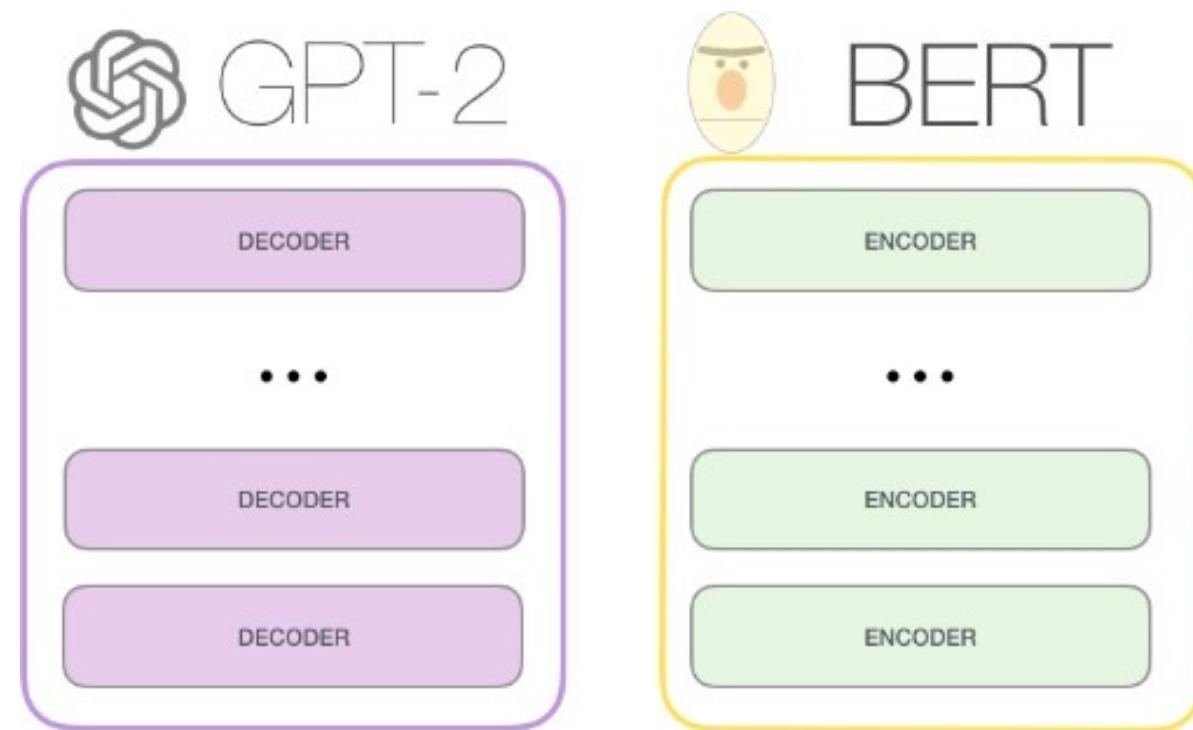
A search bar labeled "Search Models" and a sorting option "Sort: Most Downloads" are also visible.

# OpenAI applications



# GPT-2, BERT

---

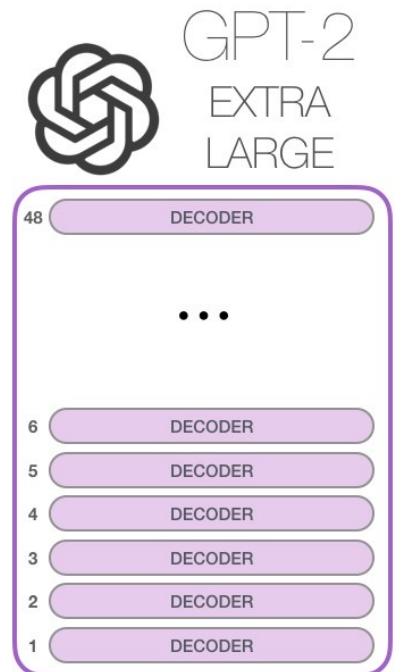
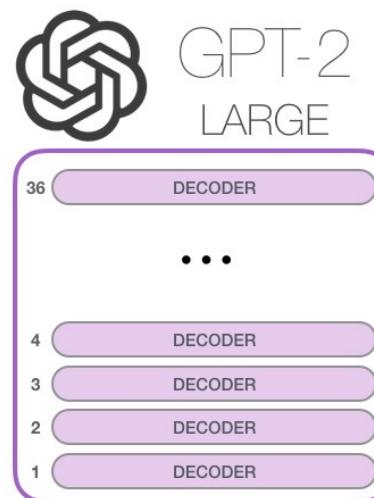
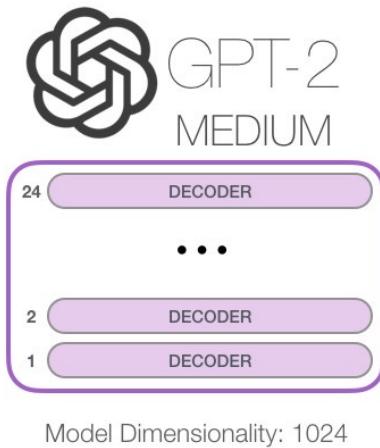
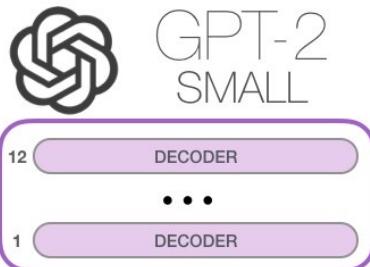


# Development of GPT

GPT released June 2018

GPT-2 released Nov. 2019 with 1.5B parameters

GPT-3 released in 2020 with 175B parameters



# Text Generation

---

```
!pip install transformers
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
generator("Hello, I'm a language model", max_length = 30,
num_return_sequences=3)
```

[{'generated\_text': "Hello, I'm a language model. It's like looking at it, where is each word of the sentence? That's what I mean. Like"},  
 {'generated\_text': "Hello, I'm a language modeler. I'm using this for two purposes: I'm having a lot fewer bugs and faster performance. If I"},  
 {'generated\_text': "Hello, I'm a language model, and I was born to code.\n\nNow, I am thinking about this from a different perspective with a"}]

# Text Generation

---

```
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
outputs = generator("Once upon a time", max_length = 30)
print(outputs[0]['generated_text'])
```

Once upon a time, every person who ever saw Jesus, knew that He was Christ. And even though he might not have known Him, He was

# Text Generation

---

```
from transformers import pipeline
generator = pipeline('text-generation', model = 'gpt2')
outputs = generator("Once upon a time", max_length = 100)
print(outputs[0]['generated_text'])
```

Once upon a time we should be able to speak to people who have lost children, so we try to take those that have lost the children to our institutions – but the first time is very hard for us because of our institutions. To me, it's important to acknowledge that in an institution of faith and love they are not children. And that there are many people who are still hurting the child and there are many in need help, if not a system. So I'm very curious.

# Text2Text Generation

---

```
from transformers import pipeline
text2text_generator = pipeline("text2text-generation", model = 't5-base')
outputs = text2text_generator("translate from English to
French: I am a student")
print(outputs[0]['generated_text'])
```

I am a student

Je suis un étudiant

# Text2Text Generation

---

```
from transformers import pipeline
text2text_generator = pipeline("text2text-generation")
text2text_generator("question: What is 42 ? context: 42 is
the answer to life, the universe and everything")
```

```
[{'generated_text': 'the answer to life, the universe and everything'}]
```