



SCHOOL OF
COMPUTING &
DATA SCIENCE
The University of Hong Kong



香港大學自然語言處理實驗室
Natural Language Processing Group, The University of Hong Kong

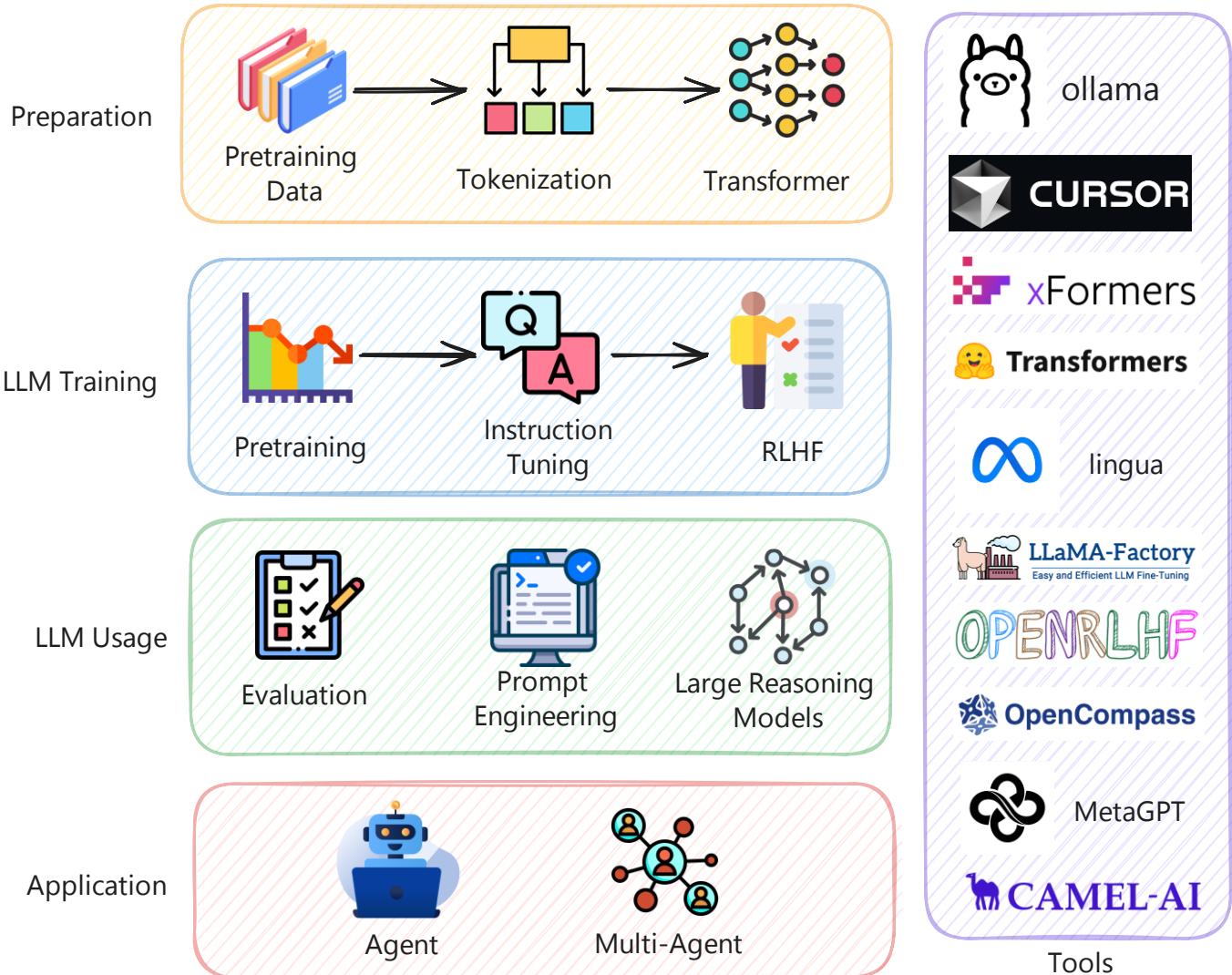
Hands-on Large Language Models

Xiachong Feng, Lingpeng Kong

20/03/2025

Outline

- Pretraining Data
- Tokenization
- Transformer
- Pre-training
- Instruction-tuning
- Reinforcement Learning from Human Feedback
- Evaluation
- Prompt Engineering
- Large Reasoning Models
- Agent
- Multi-agents
- Conclusion



Pretraining Data

1. Common Crawl
2. FineWeb
3. Code Pretraining Data
4. Math Pretraining Data



Common Crawl

- Common Crawl is a non-profit founded in 2007.
- Common crawl is an open repository of web crawl data that can be accessed and analyzed by researchers, data scientists, and developers.
 - Over 250 billion pages spanning 18 years.
 - 3–5 billion new pages added each month.

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

Table 2.2: Datasets used to train GPT-3 “Weight in training mix” refers to the fraction of examples during training that are drawn from a given dataset, which we intentionally do not make proportional to the size of the dataset. As a result, when we train for 300 billion tokens, some datasets are seen up to 3.4 times during training while other datasets are seen less than once.



Common Crawl

Accessing the Data

Crawl data is free to access by anyone from anywhere.

The data is hosted by [Amazon Web Services' Open Data Sets Sponsorships](#) program on the bucket `s3://commoncrawl/`, located in the `US-East-1` (Northern Virginia) AWS Region.

You may process the data in the AWS cloud or download it for free over HTTP(S) with a good Internet connection.

Choose a crawl...

CC-MAIN-2025-08

CC-MAIN-2025-05

CC-MAIN-2024-51

... using the URL schemes `http://commoncrawl.org/` and `https://commoncrawl.org/`.

<https://commoncrawl.org/get-started>



Common Crawl February 2025 Crawl Archive (CC-MAIN-2025-08)

The February 2025 crawl archive contains 2.67 billion pages, see the [announcement](#) for details.

Data Size and File Listings

Data Type	File List	#Files	Total Size Compressed (TiB)
Segments	segment.paths.gz	100	
WARC	warc.paths.gz	90000	82.17
WAT	wat.paths.gz	90000	18.98
WET	wet.paths.gz	90000	7.40
Robots.txt files	robotstxt.paths.gz	90000	0.15
Non-200 responses	non200responses.paths.gz	90000	3.09
URL index files	cc-index.paths.gz	302	0.20
Columnar URL index files	cc-index-table.paths.gz	900	0.23

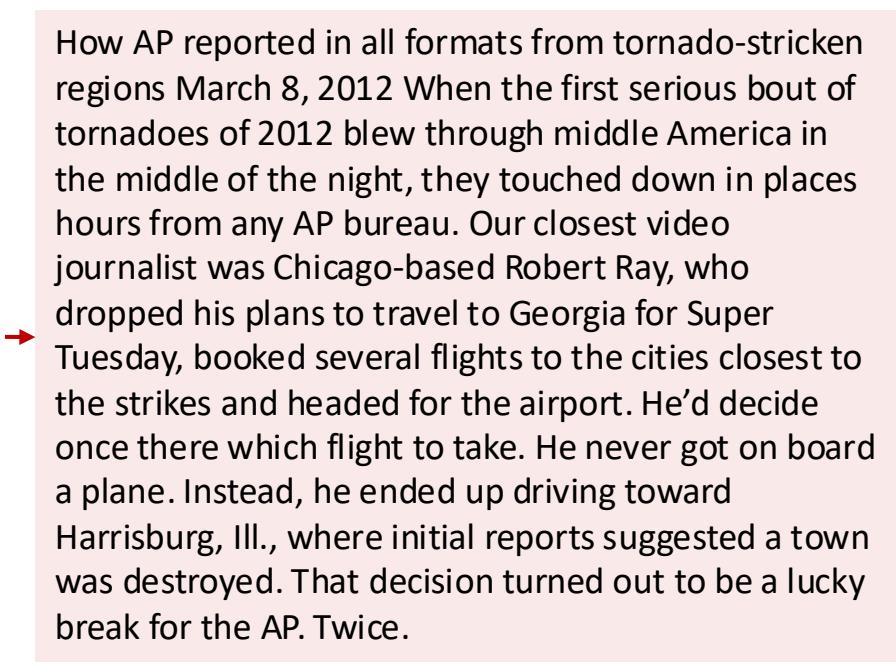
<https://data.commoncrawl.org/crawl-data/CC-MAIN-2025-08/index.html>

CC-MAIN-2025-08
↳ Main File ↳ Year ↳ Week

- **WARC** files which store the raw crawl data
- **WAT** files which store computed metadata for the data stored in the WARC
- **WET** files which store extracted plaintext from the data stored in the WARC

FineWeb

- FineWeb, a new, large-scale (**15-trillion tokens, 44TB disk space**) dataset for LLM pretraining.
- FineWeb is derived from **96 CommonCrawl** snapshots and produces better-performing LLMs than other open pretraining datasets.



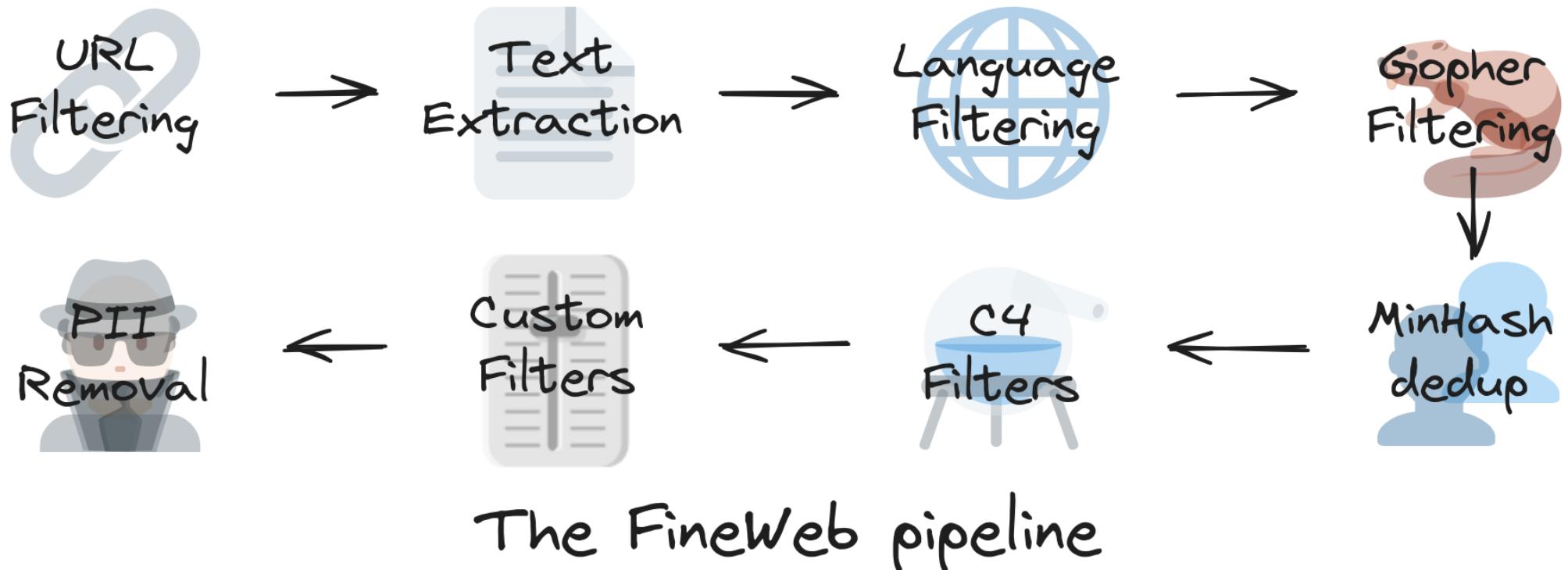
The full dataset viewer is not available (click to read why). Only showing a preview of the rows.

text string	id string	dump string	url string	date string	file_path string	language string
How AP reported in all formats from tornado-stricken regions March 8, 2012 When the first...	curn:uuid:d66bc6fe-8477-daf4-h43b...	CC-MAIN-...	http://%20jwashington@ap.org/Content/Press-Release/2012/How-AP-reported-in-all-formats-from-...	2013-05-18T05:48:54Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
Did you know you have two little yellow, nine-volt-battery-sized adrenal glands in your body. Just...	curn:uuid:803e14c3-dc2e-43d6-b75d-...	CC-MAIN-...	http://100awesomethings.com/2012/09/24/934-adrenaline/	2013-05-18T08:11:45Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
Car Wash for Clara! Now is your chance to help! 2 year old Clara Woodward has Cancer! Clara can't sa...	curn:uuid:ac1bbfff-9519-4967-9c64-...	CC-MAIN-...	http://1027kord.com/car-wash-for-clara/	2013-05-18T06:49:55Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
Listeners Get Sky-high View of Missoula From Hot Air Balloons On Friday, June 1, during the...	curn:uuid:c1445c58-b111-4c4e-badd-...	CC-MAIN-...	http://1075zoofm.com/listeners-get-sky-high-view-of-missoula-from-hot-air-balloons/	2013-05-18T06:25:20Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
Log In Please enter your ECode to log in. Forgotten your eCode? If you created your login but do not...	curn:uuid:e5829f7d-b944-4468-9573-...	CC-MAIN-...	http://1105govinfoevents.com/enterprisearchitectur eeevent/public/MyBriefcasef671.html?...	2013-05-18T05:27:01Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
spotlight provide a convenient rechargeable LED light for work play and everyday life. choose from...	curn:uuid:6bfcfa20f-ea07-41ba-b995-...	CC-MAIN-...	http://12vspotlight.com/	2013-05-18T06:49:17Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
K-State put themselves in sole position of first place in the Big 12 with their 79-70 over Iowa...	curn:uuid:dc09fd8-5a21-4ab0-bbb2-...	CC-MAIN-...	http://1350kman.com/k-state-now-in-top-10/	2013-05-18T07:19:46Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
Five Reasons I Love Boston 1. The water. The Atlantic Ocean, as deep and true as denim, so blue...	curn:uuid:64f968bf-14bc-48bd-a1bb-...	CC-MAIN-...	http://17andbaking.com/2012/09/30/five-reasons-i-love-boston/?...	2013-05-18T07:25:34Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
TRIBE CHIEF TRIS DAZZLES AT DISH, FLUBS IN FIELD IN SIXTH STRAIGHT TIGER WIN By Calvin J. Butterworth...	curn:uuid:2c08e1d4-9706-41d8-84dc-...	CC-MAIN-...	http://1924andyouarethere.blogspot.com/2009_07_01_archive.html	2013-05-18T05:54:12Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
[Tommy Pi - Trance Experience] Written by Paul Tommy Pi started DJing at small private parties at...	curn:uuid:7e6216ca-0a01-498d-8517-...	CC-MAIN-...	http://1mix.co.uk/trance-shows/tommy-pi-trance-experience.html	2013-05-18T05:54:17Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
When I found out we would be getting a PopATot for review I was excited! I knew before we even had it...	curn:uuid:8868921d-8323-4a3d-b012-...	CC-MAIN-...	http://1plus1plus1qualsireviews.blogspot.com/2009_10/grand-finale-4-popatot.html?...	2013-05-18T06:19:38Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en
2012 Indy Info It seems we can't find what you're looking for. Perhaps searching can help. giving it...	curn:uuid:b7319126-5fdb-4ae0-a17b-...	CC-MAIN-...	http://2012indyinfo.com/category/sfhs/	2013-05-18T08:07:40Z	s3://commoncrawl/crawl-data/CC-MAIN-2013-...	en

<https://huggingface.co/datasets/HuggingFaceFW/fineweb>

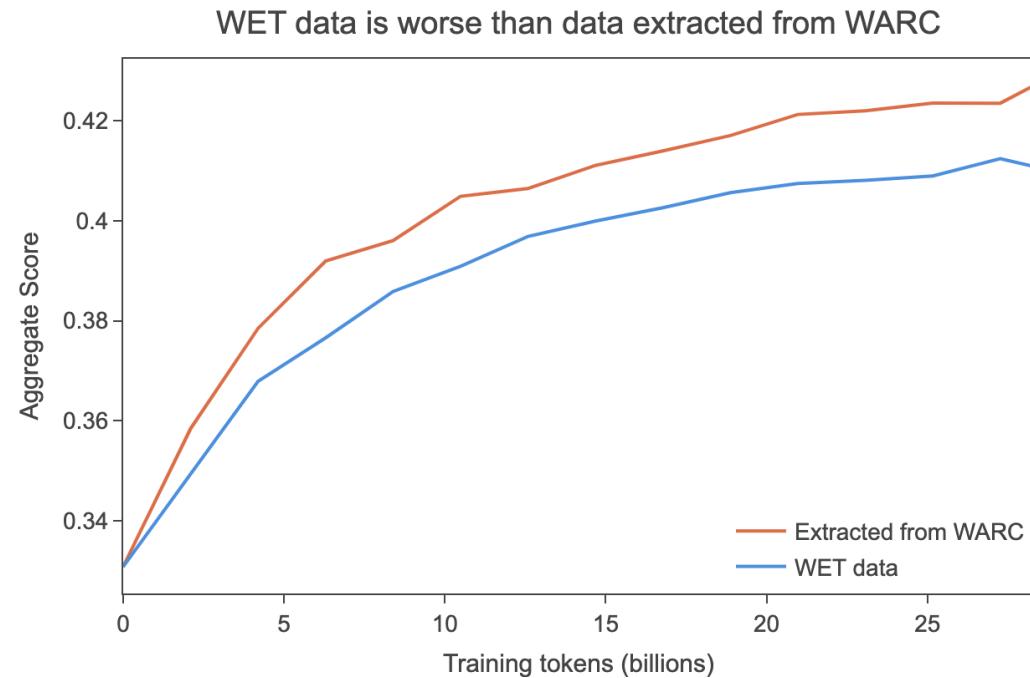
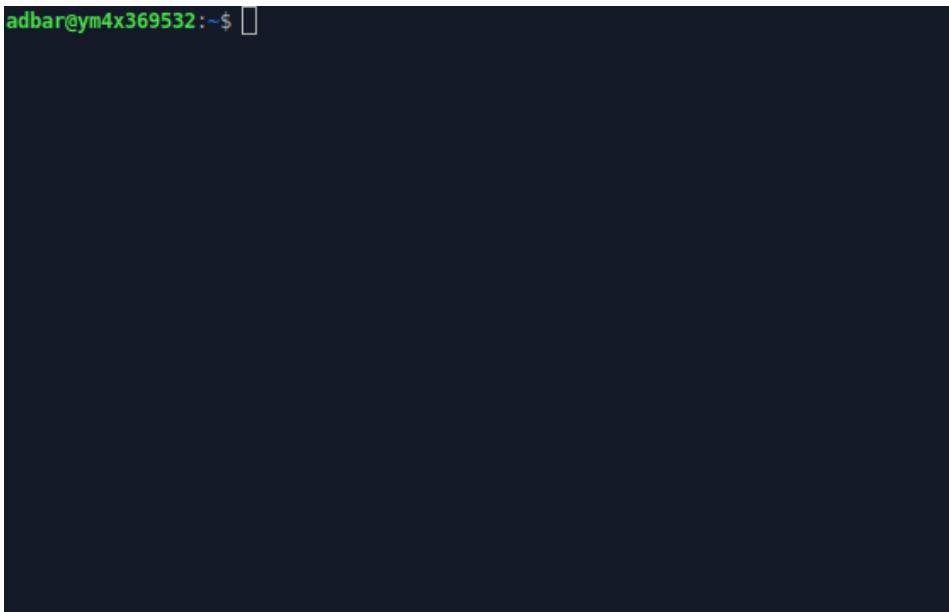
How AP reported in all formats from tornado-stricken regions March 8, 2012 When the first serious bout of tornadoes of 2012 blew through middle America in the middle of the night, they touched down in places hours from any AP bureau. Our closest video journalist was Chicago-based Robert Ray, who dropped his plans to travel to Georgia for Super Tuesday, booked several flights to the cities closest to the strikes and headed for the airport. He'd decide once there which flight to take. He never got on board a plane. Instead, he ended up driving toward Harrisburg, Ill., where initial reports suggested a town was destroyed. That decision turned out to be a lucky break for the AP. Twice.

FineWeb Recipe



FineWeb Recipe: Text Extraction

- **WARC (Web ARChive format)** files contain the raw data from the crawl, including the full page HTML and request metadata.
- **WET (WARC Encapsulated Text)** files provide a text only version of those websites.
- Extract the text content from the WARC files using the **trafilatura library**



FineWeb Recipe: Base filtering

- Filtering is an important part of the curation process.
- It consists in removing part of the data (be it words, lines, or even full documents) that lowers the performance of the model
 - Applied URL filtering using a **blocklist** (<https://dsi.ut-capitole.fr/blacklists/>) to remove adult content
 - Applied a **fastText language classifier** to keep only English text with a score ≥ 0.65
 - Applied quality and repetition filters from **MassiveText**

```
● ● ●

import fasttext
from huggingface_hub import hf_hub_download

model_path = hf_hub_download(repo_id="facebook/fasttext-
language-identification", filename="model.bin")
model = fasttext.load_model(model_path)
model.predict("Hello, world!")
# ('__label__eng_Latn',), array([0.81148803])

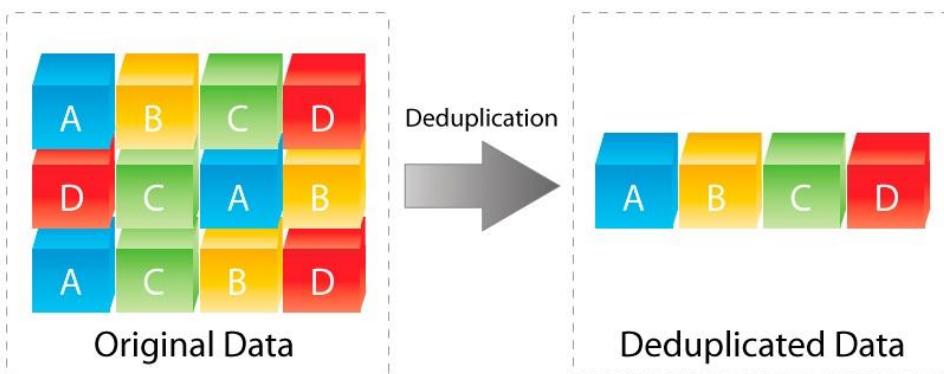
model.predict("Hello, world!", k=5)
# ('__label__eng_Latn', '__label__vie_Latn',
 '__label__nld_Latn', '__label__pol_Latn',
 '__label__deu_Latn'),
# array([0.61224753, 0.21323682, 0.09696738,
 0.01359863, 0.01319415])
```

Quality Filtering (MassiveWeb only) The vast majority of text found on the web is of insufficient quality to be useful for language model training. For example, many web pages contain primarily automatically generated content, or text that is not intended for human consumption (such as keywords for search-engine optimisation). Much of the web also comprises social media content, which can variously lack context, coherence, or substance. To remove low-quality data while minimising potential for bias, we apply a number of simple, easily understood heuristic filters: we remove any document that does not contain between 50 and 100,000 words, or whose mean word length is outside the range of 3 to 10 characters; we remove any document with a symbol-to-word ratio greater than 0.1 for either the hash symbol or the ellipsis; and we remove any document with more than 90% of lines starting with a bullet point, or more than 30% ending with an ellipsis. We also require that 80% of words in a document contain at least one alphabetic character, and apply a "stop word" filter, to remove documents that do not contain at least two of the following English words: *the, be, to, of, and, that, have, with*; this adequately deals with ostensibly English documents that contain no coherent English text.

Repetition Removal (MassiveWeb only) Another indicator of poor quality data is excessive repetition of certain words or phrases within a document. Qualitatively we observe that excessive repetition is often linked with uninformative content. Furthermore a well-studied failure mode of current language models is to repeat themselves during sampling (Holtzman et al., 2019) which may be partially attributed to repetitious training data.

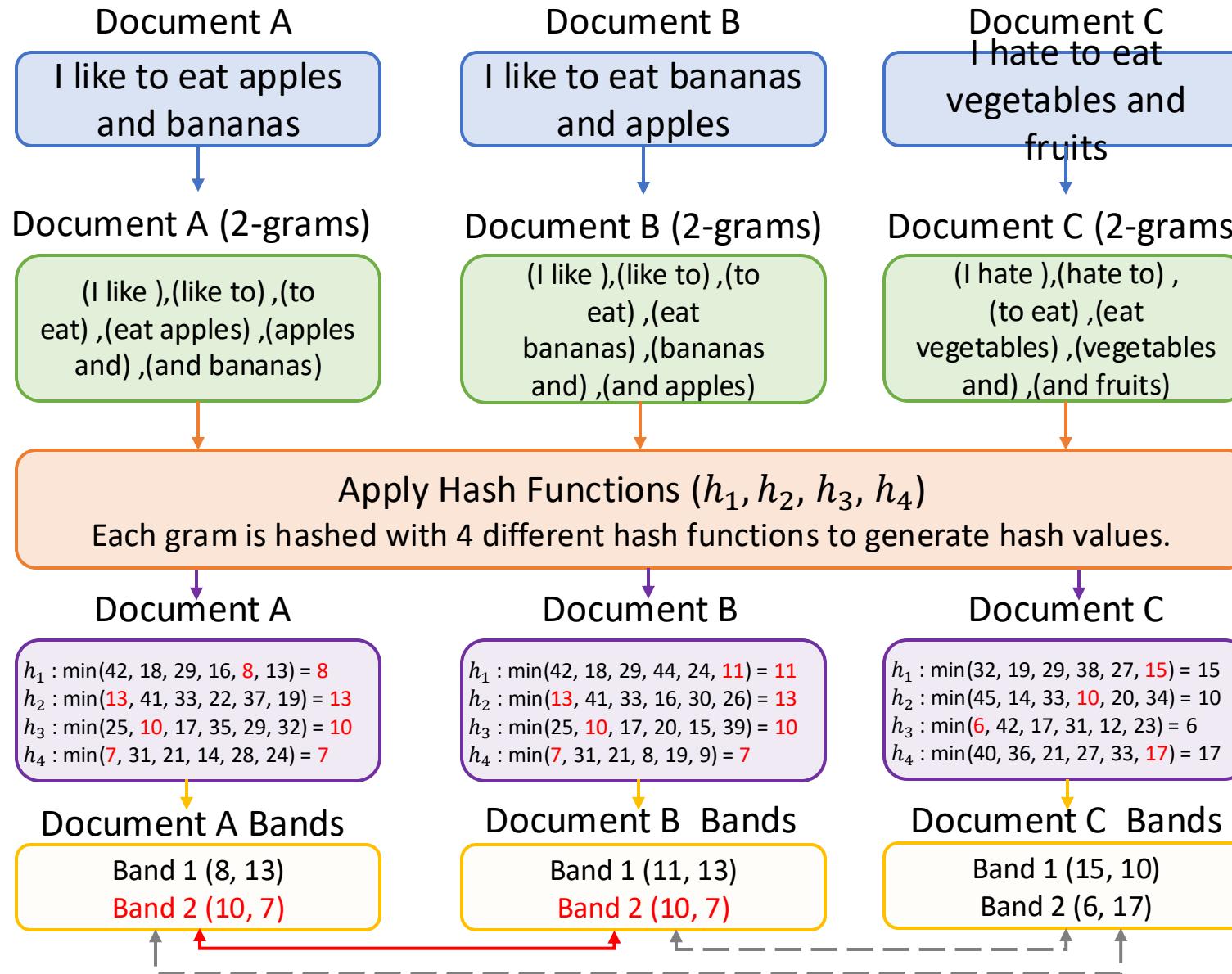
FineWeb Recipe: Deduplication

- Methods to deduplicate datasets attempt to identify and remove redundant/repeated data from the dataset.
- Removing these duplicates (deduplicating) has been correlated with improvements in model performance^[1] and a reduction in memorization of pretraining data^[2], which might allow for better generalization.
- Additionally, the performance uplift obtained through deduplication can be equated to increased training efficiency: by removing duplicated content, a model can reach the same performance level with fewer training iterations – or equivalently, for a given number of training tokens, a model will have seen more diverse data.^{[3][4]}



- [1] Deduplicating Training Data Makes Language Models Better
- [2] Quantifying Memorization Across Neural Language Models
- [3] Scaling Data-Constrained Language Models
- [4] Scaling Laws and Interpretability of Learning from Repeated Data

FineWeb Recipe: MinHash Deduplication



Shingle	h_1	h_2	h_3	h_4
"I like"	42	13	25	7
"like to"	18	41	10	31
"to eat"	29	33	17	21
"eat apples"	16	22	35	14
"apples and"	8	37	29	28
"and bananas"	13	19	32	24
"eat bananas"	44	16	20	8
"bananas and"	24	30	15	19
"and apples"	11	26	39	9
"I hate"	32	45	6	40
"hate to"	19	14	42	36
"eat vegetables"	38	10	31	27
"vegetables and"	27	20	12	33
"and fruits"	15	34	23	17

Pretraining Data: Code

Qwen2.5-Coder

Qwen2.5-Coder collected public repositories from GitHub created before February 2024, spanning 92 programming languages. Similar to StarCoder2 (Lozhkov et al., 2024) and DS-Coder (Guo et al., 2024a), we applied a series of rule-based filtering methods. In addition to raw code, we also collected data from Pull Requests, Commits, Jupyter Notebooks, and Kaggle datasets, all of which were subjected to similar rule-based cleaning techniques.

Github
92 Programming Language

Other Formats

DeepSeek-Coder

We collect public repositories created before February 2023 on GitHub and retain only 87 programming languages, as listed in Table 1. To reduce the amount of data to be processed, we apply filtering rules similar to those used in the StarCoder project (Li et al., 2023) to preliminarily filter out lower-quality code. By applying these filtering rules, we reduce the total amount of data to only 32.8% of its original size. To make the paper self-contained, we briefly describe the filter rules used in the StarCoder Data project:

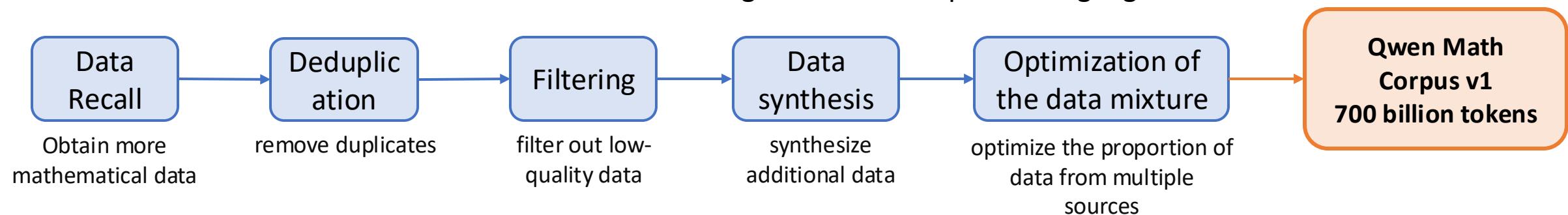
Firstly, we filter out files with an average line length exceeding 100 characters or a maximum line length surpassing 1000 characters. Additionally, we remove files with fewer than 25% alphabetic characters. Except for the XSLT programming language, we further filter out files where the string "<?xml version=" appeared in the first 100 characters. For HTML files, we consider the ratio of visible text to HTML code. We retain files where the visible text constitutes at least 20% of the code and is no less than 100 characters. For JSON and YAML files, which typically contain more data, we only keep files that have a character count ranging from 50 to 5000 characters. This effectively removes most data-heavy files.

Github
87 Programming Language

Code-specific Rule-based Filtering

Pretraining Data: Math (*Qwen Math Corpus v1*)

- **Data recall**
 - Recall of mathematical data from web sources, such as Common Crawl, to escalate the quantity of data
 - Train a FastText classifier to classify mathematical texts.
 - Leverage meta-information, such as URLs, to expand the data pool for mathematical data retrieval.
- **Deduplication**
 - MinHash is employed to filter out similar mathematical documents.
- **Filtering**
 - Language-model-based filtering technique to further curate the dataset.
- **Data synthesis**
 - Employ the Qwen2-72B-Instruct model to synthesize a large amount of mathematical pre-training corpus
- **Optimization of the data mixture**
 - Conduct ablation studies on data mixture using a small math-specific language model.

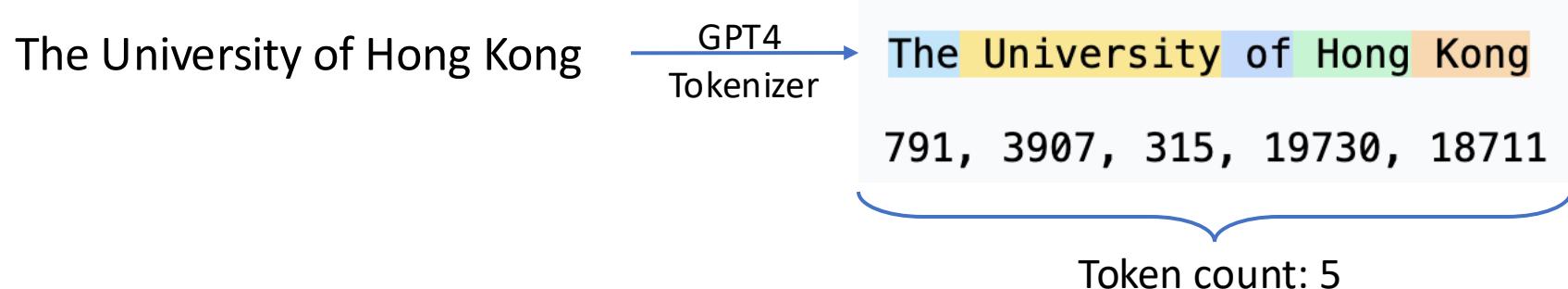


Tokenization

1. What is tokenization?
2. Word-level Tokenization
3. Byte Pair Encoding (BPE)
4. Quirks of LLM Tokenization
5. Byte-Level Large Language Model

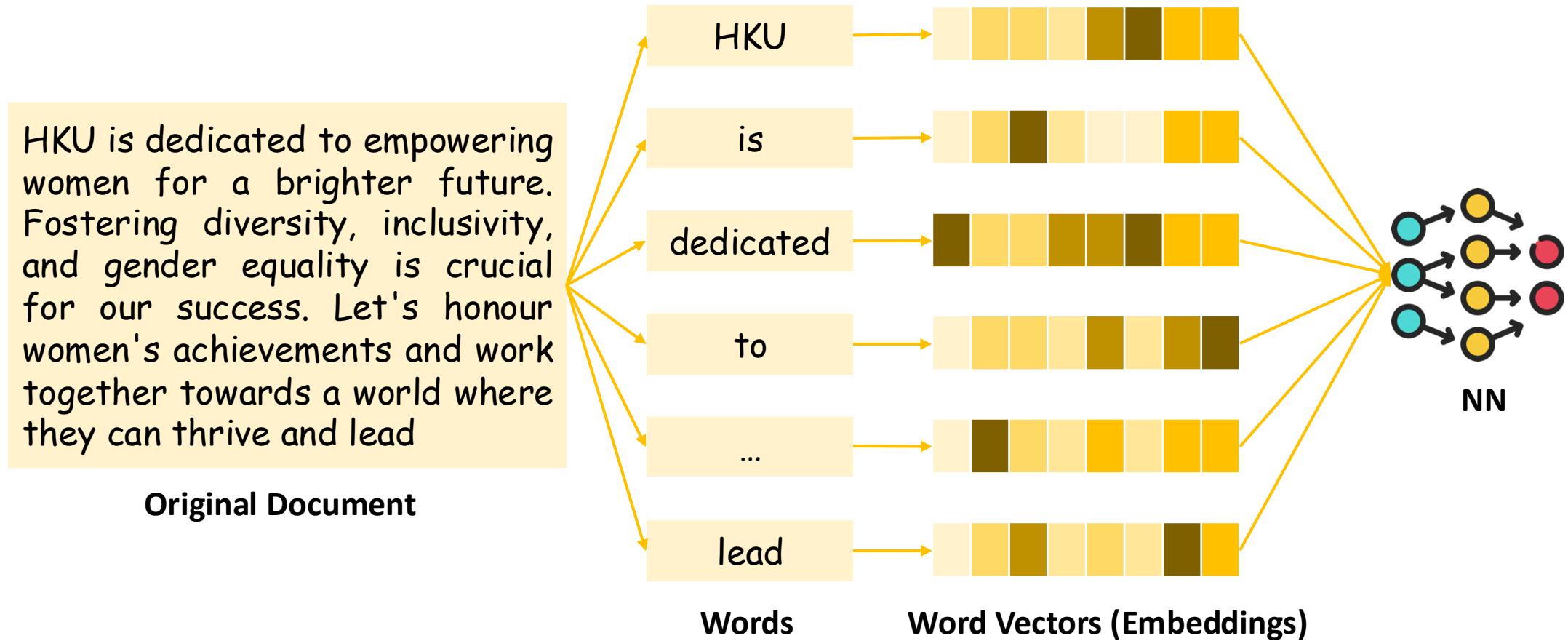
Tokenization

Tokenization is the process of splitting text into smaller units (tokens) to enable efficient processing and analysis in Natural Language Processing.



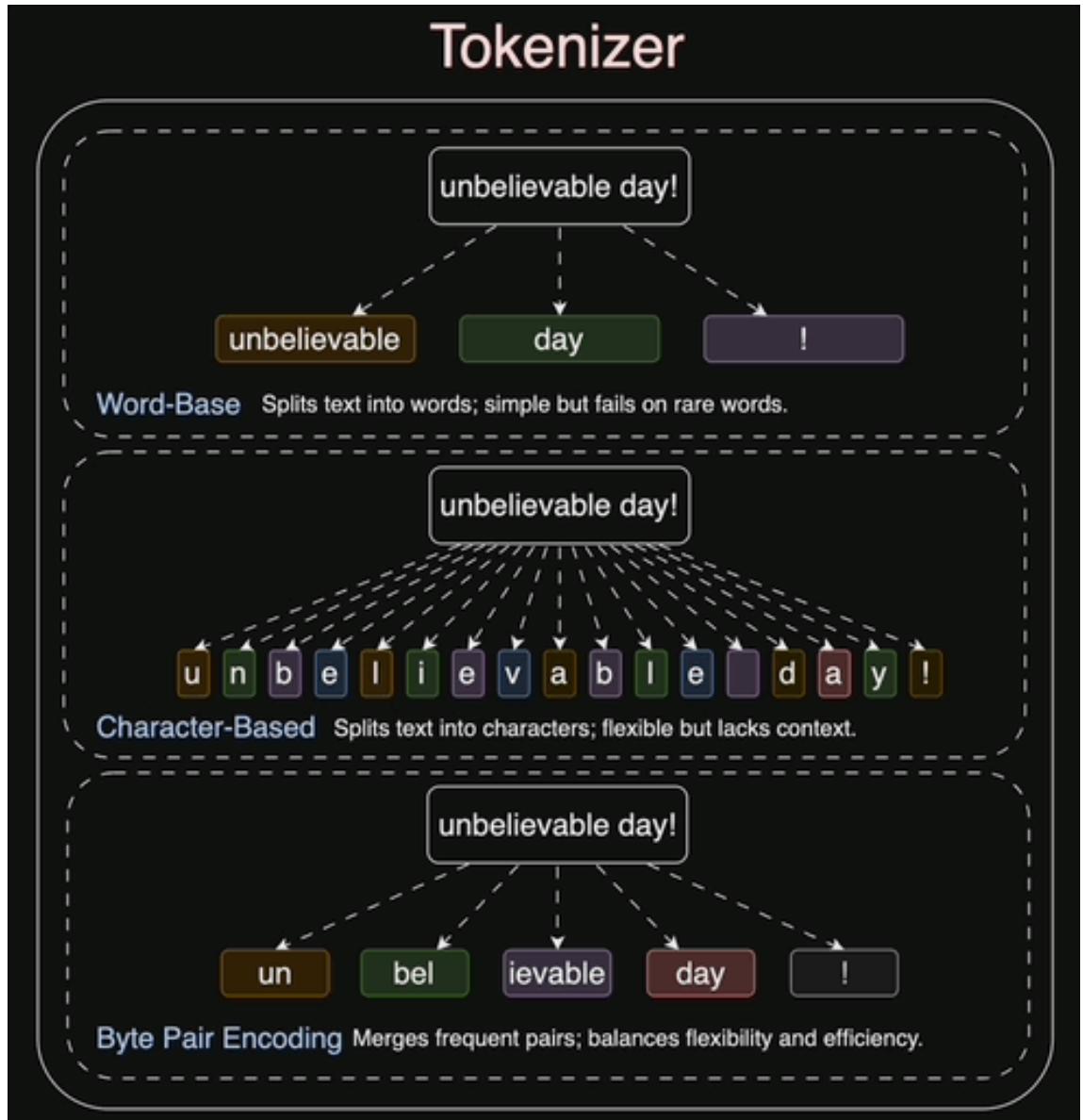
Basic: Word-Level Tokenization

- Word-level tokenization splits text into individual words based on spaces and punctuation, making it suitable for languages with clear word boundaries like English.



Byte Pair Encoding (BPE)

- Byte Pair Encoding (BPE) is a subword tokenization algorithm that iteratively merges the most frequent adjacent character pairs to create a vocabulary of subwords, helping models handle rare words and out-of-vocabulary (OOV) words efficiently.



Byte Pair Encoding (BPE)

- Two words in the dataset: *low* and *lowest*
- **Initialization**
 - Start with each character in the corpus as a separate token
 - *Vocabulary:* {*l, o, w, e, s, t, </w>*}
- **Tokenization at Character Level:**
 - “*low*” → *l o w </w>*
 - “*lowest*” → *l o w e s t </w>*
- **Counting Pair Frequencies**
 - Pair *l o* appears **twice**;
 - Pair *o w* appears **twice**;
 - Pair *w </w>* appears **once**;
 - Pair *e s* appears **once**;
 - Pair *s t* appears **once**: in “*lowest*”.
- **Merge the Most Frequent Pair:**
 - “*low*” → *l **ow** </w>*
 - “*lowest*” → *l **ow** e s t </w>*
 - *Vocabulary:* {*l, o, w, e, s, t, </w>, ow*}

- **Recount Pair Frequencies:**
 - Pair *l **ow*** appears **twice**;
 - Pair *e s* appears **once**;
 - Pair *s t* appears **once**.
- **Merge the Next Frequent Pair:**
 - “*low*” → *low </w>*
 - “*lowest*” → *low e s t </w>*
 - *Vocabulary:* {*l, o, w, e, s, t, </w>, ow, low*}
- **Merge the Next Frequent Pair:**
 - “*lowest*” → *low es t </w>*
 - *Vocabulary:* {*l, o, w, e, s, t, </w>, ow, low, es*}
- **Merge the Final Pair:**
 - “*lowest*” → *low est </w>*
 - *Vocabulary:* {*l, o, w, e, s, t, </w>, ow, low, es, est*}
- **Final Vocabulary:**
 - *Vocabulary:* {*l, o, w, e, s, t, </w>, ow, low, es, est*}
 - “*low*” → *low </w>*
 - “*lowest*” → *low est </w>*

Merge Rules	<i>o w → ow</i> <i>l o w → low</i> <i>e s → es</i> <i>es t → est</i>
--------------------	---

Vocab	{ <i>l, o, w, e, s, t, </w>, ow, low, es, est</i> }
--------------	---

Apply BPE for New Word

lower ← new word
↓
l o w e r </w> ← initialization
↓
low e r </w> ← merge
↓
low e unk </w> ← final

Note1: Byte-level Byte Pair Encoding (BPE)

- A base vocabulary that includes all possible base characters can be quite large if *e.g.* all unicode characters are considered as base characters.
- To have a better base vocabulary, GPT-2 uses bytes as the base vocabulary, which is a clever trick to force the base vocabulary to be of size **256** while ensuring that every base character is included in the vocabulary.



```
>>> list("香港大学 🙌 (The University of Hong Kong!)".encode("utf-8"))
[233, 166, 153, 230, 184, 175, 229, 164, 167, 229, 173,
 166, 32, 240, 159, 145, 139, 32, 40, 84, 104, 101, 32,
 85, 110, 105, 118, 101, 114, 115, 105, 116, 121, 32,
 111, 102, 32, 72, 111, 110, 103, 32, 75, 111, 110, 103,
 33, 41]
```

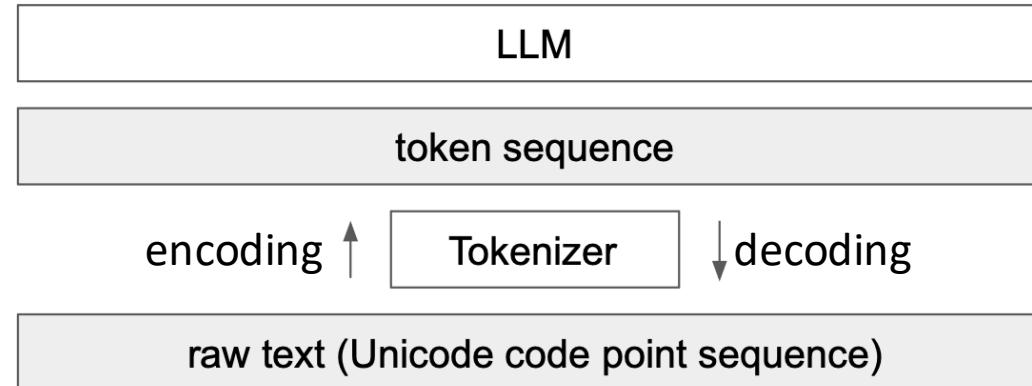


```
>>> list("香港大学".encode("utf-8"))
[233, 166, 153, 230, 184, 175, 229,
 164, 167, 229, 173, 166]
```

Byte sequence

Note2: Independent Module Separate from LLM

- Tokenizer is a completely separate, independent module from the LLM. It has its own training dataset of text (which could be different from that of the LLM), on which you train the vocabulary using the Byte Pair Encoding (BPE) algorithm. It then translates back and forth between raw text and sequences of tokens.
- The LLM later only ever sees the tokens and never directly deals with any text.



Note3: Visualization Tool for Tokenization

Tiktokenizer

The University of Hong Kong

香港大学

```
def decode(ids):
    # given ids (list of integers), return Python string
    text_bytes = b"".join(vocab[idx] for idx in ids)
    text = text_bytes.decode("utf-8", errors="replace")
    return text
```

今日天氣唔錯。

hello world

helloworld

hello. world

cl100k_base

Token count

75

The University of Hong Kong

香港大学

```
def decode(ids):
    # given ids (list of integers), return Python string
    text_bytes = b"".join(vocab[idx] for idx in ids)
    text = text_bytes.decode("utf-8", errors="replace")
    return text
```

今日天氣唔錯。

791, 3907, 315, 19730, 18711, 271, 46065, 247, 35086,
107, 27384, 48864, 271, 755, 17322, 44548, 997, 262, 6
74, 2728, 14483, 320, 1638, 315, 26864, 705, 471, 1332
5, 925, 198, 262, 1495, 12807, 284, 293, 70455, 6115,
3832, 21135, 11858, 60, 369, 7335, 304, 14483, 340, 26
2, 1495, 284, 1495, 12807, 16301, 446, 4867, 12, 23, 4
98, 6103, 429, 8319, 1158, 262, 471, 1495, 271, 37271,
9080, 36827, 30320, 96, 84150, 242, 91779, 107, 1811

Minimal Implementation

```
def get_stats(ids, counts=None):
    """
    Given a list of integers, return a dictionary of counts of consecutive pairs
    Example: [1, 2, 3, 1, 2] -> {(1, 2): 2, (2, 3): 1, (3, 1): 1}
    Optionally allows to update an existing dictionary of counts
    """
    counts = {} if counts is None else counts
    for pair in zip(ids, ids[1:]): # iterate consecutive elements
        counts[pair] = counts.get(pair, 0) + 1
    return counts
```

```
def merge(ids, pair, idx):
    """
    In the list of integers (ids), replace all consecutive occurrences
    of pair with the new integer token idx
    Example: ids=[1, 2, 3, 1, 2], pair=(1, 2), idx=4 -> [4, 3, 4]
    """
    newids = []
    i = 0
    while i < len(ids):
        # if not at the very last position AND the pair matches, replace it
        if ids[i] == pair[0] and i < len(ids) - 1 and ids[i+1] == pair[1]:
            newids.append(idx)
            i += 2
        else:
            newids.append(ids[i])
            i += 1
    return newids
```

```
def train(text, vocab_size, verbose=False):
    assert vocab_size >= 256
    num_merges = vocab_size - 256

    # input text preprocessing
    text_bytes = text.encode("utf-8") # raw bytes
    ids = list(text_bytes) # list of integers in range 0..255

    # iteratively merge the most common pairs to create new tokens
    merges = {} # (int, int) -> int
    vocab = {idx: bytes([idx]) for idx in range(256)} # int -> bytes
    for i in range(num_merges):
        # count up the number of times every consecutive pair appears
        stats = get_stats(ids)
        # find the pair with the highest count
        pair = max(stats, key=stats.get)
        # mint a new token: assign it the next available id
        idx = 256 + i
        # replace all occurrences of pair in ids with idx
        ids = merge(ids, pair, idx)
        # save the merge
        merges[pair] = idx
        vocab[idx] = vocab[pair[0]] + vocab[pair[1]]
        print(f"merge {i+1}/{num_merges}: {pair} -> {idx}\n{vocab[idx]} had {stats[pair]} occurrences")
```

Additional: CURSOR

① Select the code

```
def train(self, text, vocab_size, verbose=False):
    assert vocab_size >= 256
    num_merges = vocab_size - 256

    # input text preprocessing
    text_bytes = text.encode("utf-8") # raw bytes
    ids = list(text_bytes) # list of integers in range 0..255

    # iteratively merge the most common pairs to create new tokens
    merges = {} # (int, int) -> int
    vocab = {idx: bytes([idx]) for idx in range(256)} # int -> bytes

    for i in range(num_merges):
        # count up the number of times every consecutive pair appears
        stats = get_stats(ids)
        # find the pair with the highest count
        pair = max(stats, key=stats.get)
        # mint a new token: assign it the next available id
        idx = 256 + i
        # replace all occurrences of pair in ids with idx
        ids = merge(ids, pair, idx)
        # save the merge
        merges[pair] = idx
        vocab[idx] = vocab[pair[0]] + vocab[pair[1]]
        # prints
        if verbose:
            print(
                f"merge {i+1}/{num_merges}: {pair} -> {idx} ({vocab[idx]}) had {stats[pair]} occurrences")

    # save class variables
    self.merges = merges # used in encode()
```

② Ask the question

bpe.py bpe.py (188-188)

explain this code.

Let me explain this line of code in English.

The line `vocab[idx] = vocab[pair[0]] + vocab[pair[1]]` is creating a new vocabulary entry by concatenating two existing vocabulary items. This is a key part of the BPE (Byte Pair Encoding) algorithm.

Let's break it down:

1. `vocab` is a dictionary that maps token IDs to their byte representations. For example:

python

```
vocab = {
    97: b'a', # byte representation of 'a'
    98: b'b', # byte representation of 'b'
    256: b'aa' # a merged token
}
```

2. `idx` is the new token ID being created (usually starting from 256, as 0-255 are reserved for basic bytes)

3. `pair[0]` and `pair[1]` are the IDs of the two tokens being merged

4. `vocab[pair[0]]` gets the byte representation of the first token

`vocab[pair[1]]` gets the byte representation of the second token

5. The `+` operator concatenates these byte sequences

For example, if:

- `pair = (97, 97)` (representing two 'a' characters)

- `idx = 256`

- `vocab[97] = b'a'`

Then `vocab[256] = b'a' + b'a'` will result in `vocab[256] = b'aa'`

This is how the BPE algorithm builds up its vocabulary of larger tokens from smaller ones. Each new token is created by concatenating the byte sequences of two existing tokens that frequently appear together in the training text.

③ Explain and Answer

Minimal Implementation

```
def decode(ids):
    # given ids (list of integers), return Python string
    text_bytes = b"".join(vocab[idx] for idx in ids)
    text = text_bytes.decode("utf-8", errors="replace")
    return text

def encode(text):
    # given a string text, return the token ids
    text_bytes = text.encode("utf-8") # raw bytes
    ids = list(text_bytes) # list of integers in range 0..255
    while len(ids) >= 2:
        # find the pair with the lowest merge index
        stats = get_stats(ids)
        # This is different from the training phase where we chose
        # the most frequent pairs. During encoding, we want to apply the merges
        # in the same order they were learned, so we choose the pair with the
        # lowest merge index.
        # The use of float("inf") ensures that pairs that weren't in
        # the original training merges will never be selected, as they'll always
        # have an "infinite" merge index.
        pair = min(stats, key=lambda p: merges.get(p, float("inf")))
        # subtle: if there are no more merges available, the key will
        # result in an inf for every single pair, and the min will be
        # just the first pair in the list, arbitrarily
        # we can detect this terminating case by a membership check
        if pair not in merges:
            break # nothing else can be merged anymore
        # otherwise let's merge the best pair (lowest merge index)
        idx = merges[pair]
        ids = merge(ids, pair, idx)
    return ids
```

explain this code

Let me explain this line of code in English.

The line `pair = min(stats, key=lambda p: self.merges.get(p, float("inf")))` is finding the pair of tokens that has the lowest merge index in the previously learned merges. This is used during the encoding process of BPE.

Let's break it down:

1. `stats` is a dictionary containing pairs of tokens and their frequencies in the current text being encoded
2. `min()` is finding the pair with the smallest value according to the key function
3. `lambda p: self.merges.get(p, float("inf"))` is the key function that determines how to compare pairs:
 - `p` is each pair from the stats dictionary
 - `self.merges.get(p, float("inf"))` looks up the merge index for the pair
 - If the pair isn't in `self.merges`, it returns `float("inf")` (infinity)

For example, if we have:

```
python
stats = {(1, 2): 3, (2, 3): 2} # Current pairs and their frequencies
self.merges = {
    (1, 2): 256, # This pair was merged first
    (2, 3): 257  # This pair was merged second
}
```

The line will return `(1, 2)` because:

- For pair `(1, 2)`: `self.merges.get((1, 2))` returns `256`
- For pair `(2, 3)`: `self.merges.get((2, 3))` returns `257`
- `256` is less than `257`, so `(1, 2)` is chosen

This is different from the training phase where we chose the most frequent pairs. During encoding, we want to apply the merges in the same order they were learned, so we choose the pair with the lowest merge index.

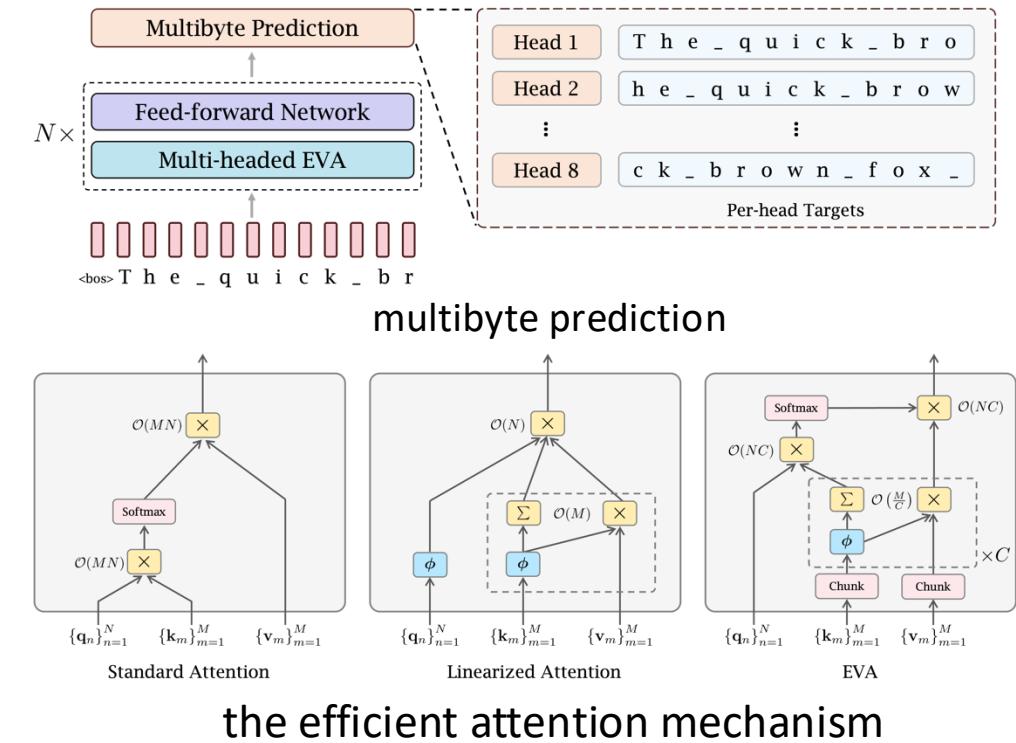
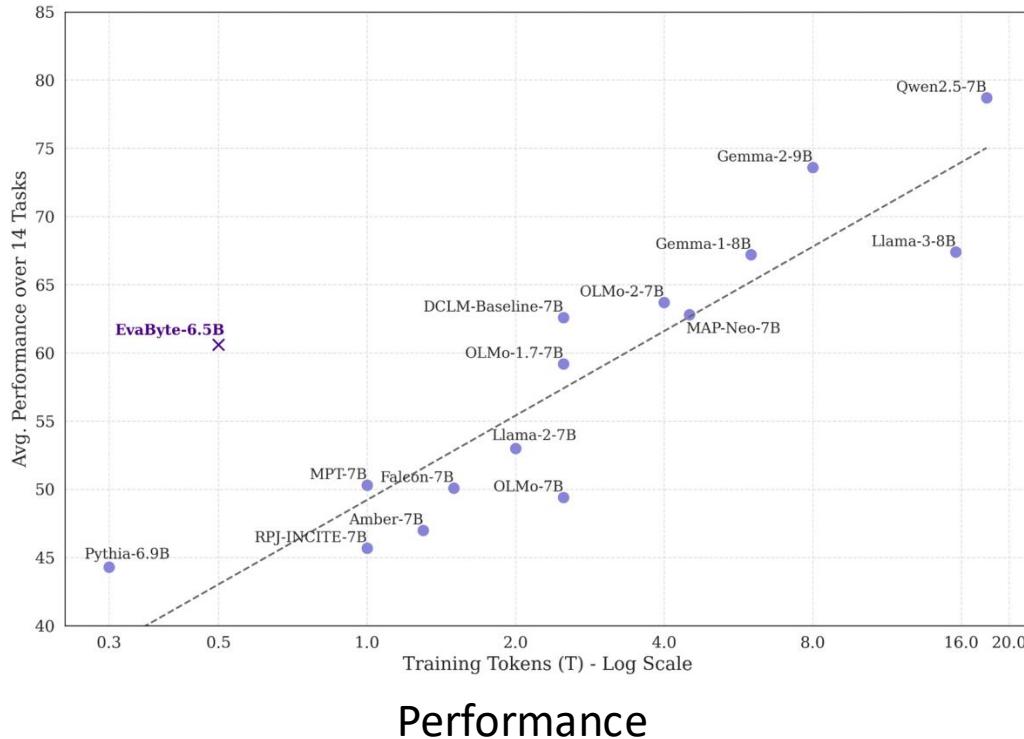
The use of `float("inf")` ensures that pairs that weren't in the original training merges will never be selected, as they'll always have an "infinite" merge index.

Quirks of LLM Tokenization

- Why can't LLM spell words? **Tokenization**.
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization**.
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization**.
- Why is LLM bad at simple arithmetic? **Tokenization**.
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization**.
- Why did my LLM abruptly halt when it sees the string "<| endoftext |>"? **Tokenization**.
- What is this weird warning I get about a "trailing whitespace"? **Tokenization**.
- Why does the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization**.
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization**.
- Why is LLM not actually end-to-end language modeling? **Tokenization**.
- What is the real root of suffering? **Tokenization**.

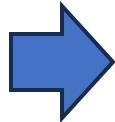
Byte-level LLM: EvaByte

- EvaByte is the first open-source byte-level model without tokenization that yet matches the performance of modern tokenizer-based LMs.



More Topics

- Regex patterns
- Tiktoker library
- Special tokens



Exercise

- <https://github.com/karpathy/minbpe>
- <https://github.com/karpathy/minbpe/blob/master/exercise.md>

The screenshot shows a Jupyter Notebook interface in a browser window. The title bar says 'Tokenization - Jupyter Notebook'. The notebook has three cells:

- Cell 348:

```
In [348]: import sentencepiece as spm
```
- Cell 349:

```
In [349]: # write a toy.txt file with some random text
with open("toy.txt", "w", encoding="utf-8") as f:
    f.write("SentencePiece is an unsupervised text tokenizer and detokenizer mainly for Neural Networks. It tokenizes text into subword units using the BPE algorithm. It can handle any language and is trained on a large corpus of text. The resulting tokens are small integers, which are then converted back into text using a vocabulary file. SentencePiece is designed to be fast and memory efficient, making it suitable for large-scale NLP applications. It is used in both Llama and Mistral series. SentencePiece runs BPE on the Unicode code points directly! It then has an option character_coverage for what to do with very rare codepoints that appear very few times, and it either maps them onto an UNK token, or if byte_fallback is turned on, it encodes them with utf-8 and then encodes the raw bytes instead.
```
- Cell 350:

```
Docs for sentencepiece options:
  • markdown
  • protobuf
```

To the right of the notebook, there is a video player showing a man with a beard and dark hair, wearing a black t-shirt, speaking. The video player has a play button and a progress bar.

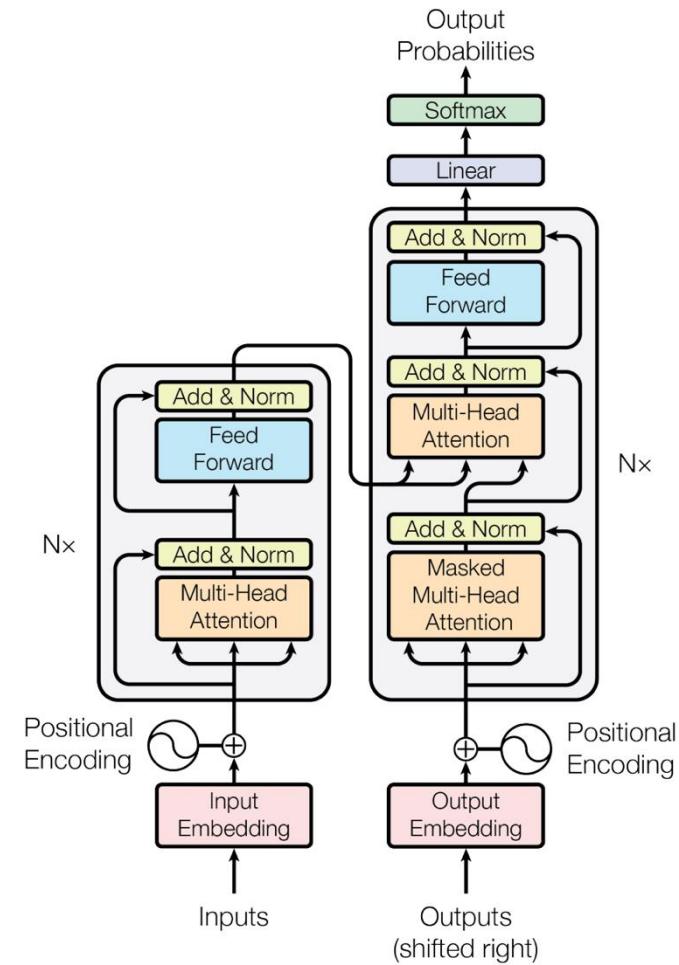
<https://www.youtube.com/watch?v=zduSFxRajkE&t=5399s>

Transformer

- Transformer
- Attention Mechanism
- Multi-Head Attention
- Visualization
- Python Library
- HuggingFace Transformers
- Different Architectures

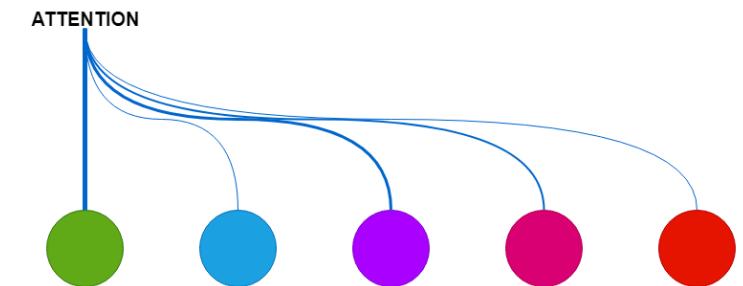
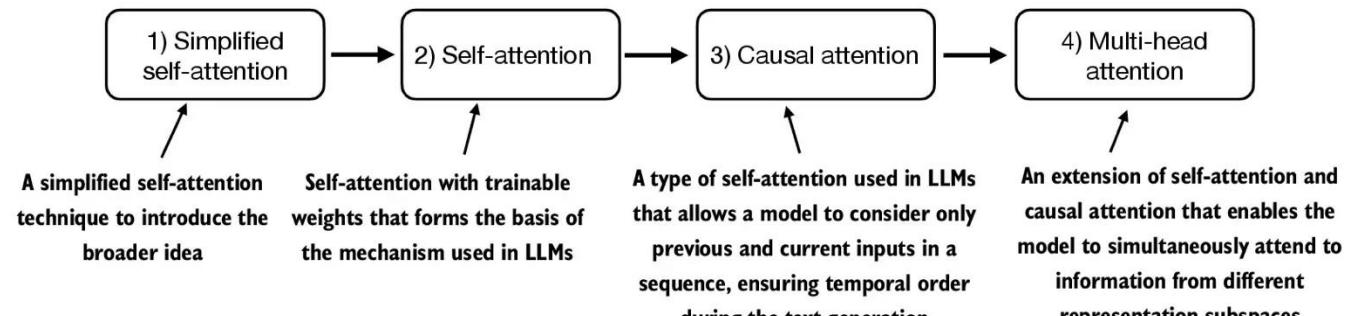
Transformer

- **Transformer Architecture** – Introduced a model relying entirely on self-attention, removing recurrence (RNNs) and convolution.
- **Self-Attention Mechanism** – Enabled parallel processing of words by attending to all positions in a sequence.
- **Multi-Head Attention** – Allowed the model to capture multiple relationships between words simultaneously.
- **Positional Encoding** – Injected sequence order information into the model, since Transformers lack recurrence.
- **State-of-the-Art Results** – Achieved breakthrough performance in machine translation and laid the foundation for models like BERT & GPT.



Attention Mechanism

- Attention in Transformers is a mechanism that enables the model to focus on relevant parts of the input sequence by dynamically weighting token relationships, improving context understanding and long-range dependencies.



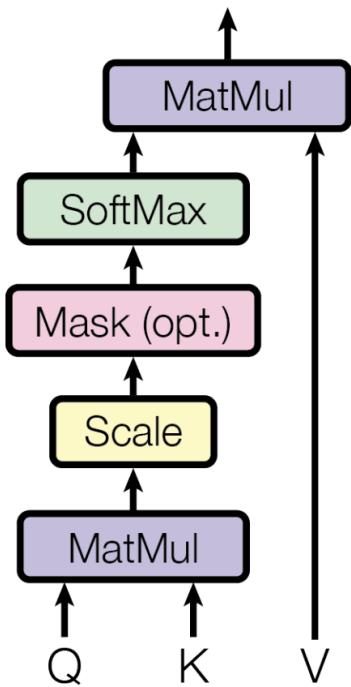
© 2024 Sebastian Raschka

<https://github.com/rasbt/LLMs-from-scratch>

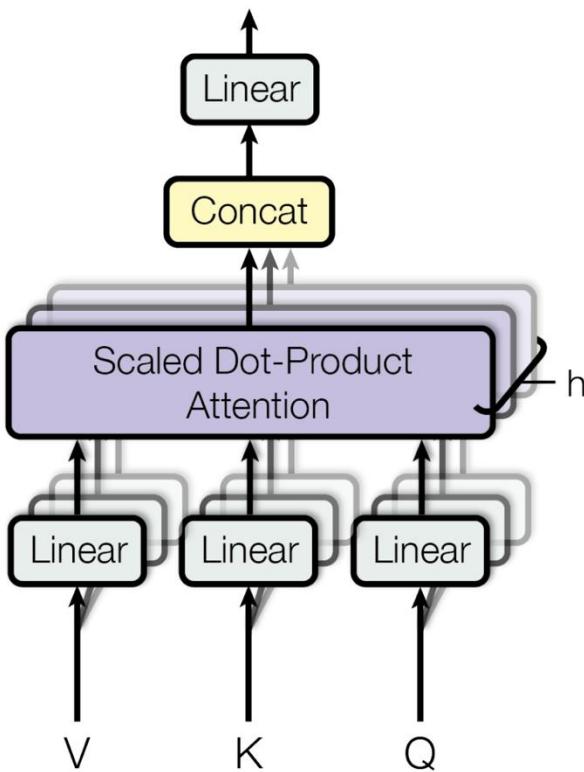
<https://lih-verma.medium.com/query-key-and-value-in-attention-mechanism-3c3c6a2d4085>

Multi-Head Attention

Scaled Dot-Product Attention



Multi-Head Attention



```
import torch
import torch.nn as nn

class MultiHeadSelfAttention(nn.Module):
    def __init__(self, embed_dim, num_heads):
        super().__init__()
        assert embed_dim % num_heads == 0, "Embedding dim must be divisible by number of heads"

        self.num_heads = num_heads
        self.head_dim = embed_dim // num_heads

        self.W_q = nn.Linear(embed_dim, embed_dim, bias=False)
        self.W_k = nn.Linear(embed_dim, embed_dim, bias=False)
        self.W_v = nn.Linear(embed_dim, embed_dim, bias=False)
        self.W_out = nn.Linear(embed_dim, embed_dim, bias=False)

    def forward(self, x):
        batch_size, seq_len, embed_dim = x.shape

        # Compute Q, K, V
        Q = self.W_q(x).view(batch_size, seq_len, self.num_heads,
                             self.head_dim).transpose(1, 2)
        K = self.W_k(x).view(batch_size, seq_len, self.num_heads,
                             self.head_dim).transpose(1, 2)
        V = self.W_v(x).view(batch_size, seq_len, self.num_heads,
                             self.head_dim).transpose(1, 2)

        # Scaled Dot-Product Attention
        scores = torch.matmul(Q, K.transpose(-2, -1)) / torch.sqrt(torch.tensor(self.head_dim, dtype=torch.float32))
        attention_weights = torch.softmax(scores, dim=-1)
        out = torch.matmul(attention_weights, V)

        # Merge heads and apply final linear layer
        out = out.transpose(1, 2).contiguous().view(batch_size,
                                                    seq_len, embed_dim)
        return self.W_out(out)
```

Visualization

The screenshot shows a web-based visualization tool for Large Language Models (LLMs). On the left, a "Chapter: Overview" panel displays a flowchart of the LLM architecture. It starts with input tokens, which are combined with position embeddings and passed through a series of layers: token embeddings, transformer blocks (each containing multi-head causal self-attention and feed-forward networks), layer normalization, and finally a linear softmax layer. A callout box highlights the "GPT-2 (small)" model, which has 124,439,808 parameters. Below the diagram, a welcome message for the "nano-gpt" model (85,584 parameters) is shown, along with a sequence of letters (C, B, A, B, B, C) and a sorting instruction. Navigation buttons for "Continue" and "Skip" are at the bottom. On the right, a large 3D visualization shows a dense grid of colored points representing the parameter space of different LLM models, with "GPT-2" and "nano-gpt" clearly labeled.

LLM Visualization

Chapter: Overview

Table of Contents

- Intro
- Introduction**
- Preliminaries
- Components
- Embedding
- Layer Norm
- Self Attention
- Projection
- MLP
- Transformer
- Softmax
- Output

GPT-2 (small)
n_params = 124,439,808

nano-gpt
n_params = 85,584

Welcome to the walkthrough of the GPT large language model! Here we'll explore the model *nano-gpt*, with a mere 85,000 parameters.

Its goal is a simple one: take a sequence of six letters:

C B A B B C

and sort them in alphabetical order, i.e. to "ABBBC".

Continue Skip

Python Library

- There are already numerous tools available for quickly building various Transformer architectures.

The screenshot shows the homepage of the `transformers` library on GitHub. It features a large yellow emoji of a smiling face with hands clasped. The title "Transformers" is displayed next to it. Below the title are several status badges: "build failing", "license Apache-2.0", "website online", "release v4.49.0", "Contributor Covenant v2.0 adopted", and a DOI link "DOI 10.5281/zenodo.7391177". A horizontal bar below the badges lists links in multiple languages: English, 简体中文, 繁體中文, 한국어, Español, 日本語, हिन्दी, Русский, Português, ଶେଷାଳ୍, Français, Deutsch, Tiếng Việt, and العربية. A subtitle reads "State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow". A yellow banner at the bottom left contains a graduation cap emoji and the text "Part of the Hugging Face course!". Below the banner, a paragraph explains that the library provides thousands of pretrained models for tasks like text, vision, and audio. Another section lists the types of models that can be applied on, including text, images, and audio.

🤗 Transformers provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio.

These models can be applied on:

- 📄 Text, for tasks like text classification, information extraction, question answering, summarization, translation, and text generation, in over 100 languages.
- 🖼 Images, for tasks like image classification, object detection, and segmentation.
- 🎧 Audio, for tasks like speech recognition and audio classification.

<https://github.com/huggingface/transformers>



xFormers - Toolbox to Accelerate Research on Transformers

xFormers is:

- **Customizable building blocks:** Independent/customizable building blocks that can be used without boilerplate code. The components are domain-agnostic and xFormers is used by researchers in vision, NLP and more.
- **Research first:** xFormers contains bleeding-edge components, that are not yet available in mainstream libraries like PyTorch.
- **Built with efficiency in mind:** Because speed of iteration matters, components are as fast and memory-efficient as possible. xFormers contains its own CUDA kernels, but dispatches to other libraries when relevant.

<https://github.com/facebookresearch/xformers>

HuggingFace Transformers

The screenshot shows the Hugging Face website interface. At the top, there's a navigation bar with links for Models, Datasets, Spaces, Posts, Docs, Enterprise, and Pricing. Below the navigation is a search bar and a user profile icon. The main content area displays a model card for "Qwen2.5-7B-Instruct". The card includes a "Model card" tab (selected), "Files and versions", and "Community" sections. It features a "Train", "Deploy", and "Use this model" button. A chart shows "Downloads last month" at 1,588,483. Below the chart is a "Safetensors" section with "Model size" (7.62B params) and "Tensor type" (BF16). The "Inference Providers" section shows a "Together AI" provider and a "Text Generation" example in Python. The example code is:

```
def bubble_sort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n-i-1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
```

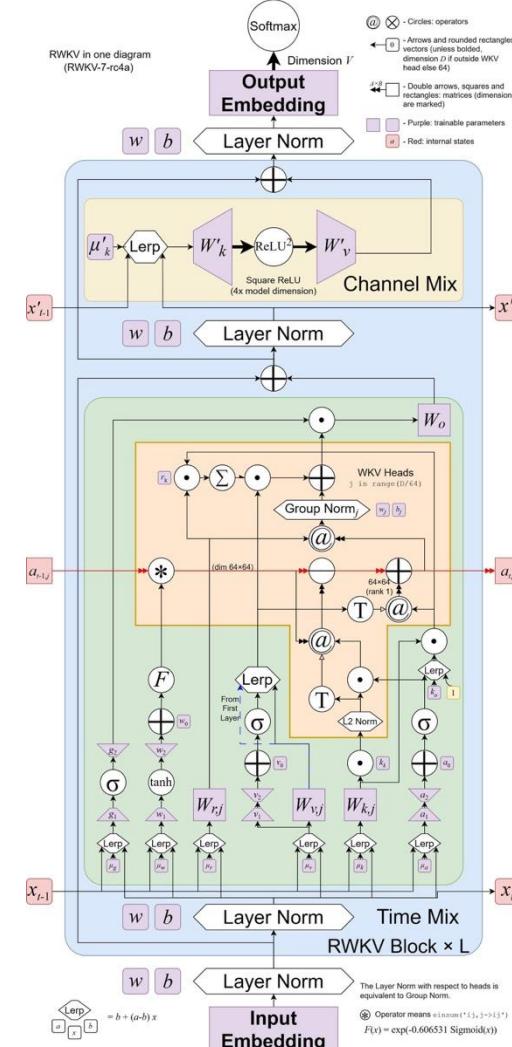
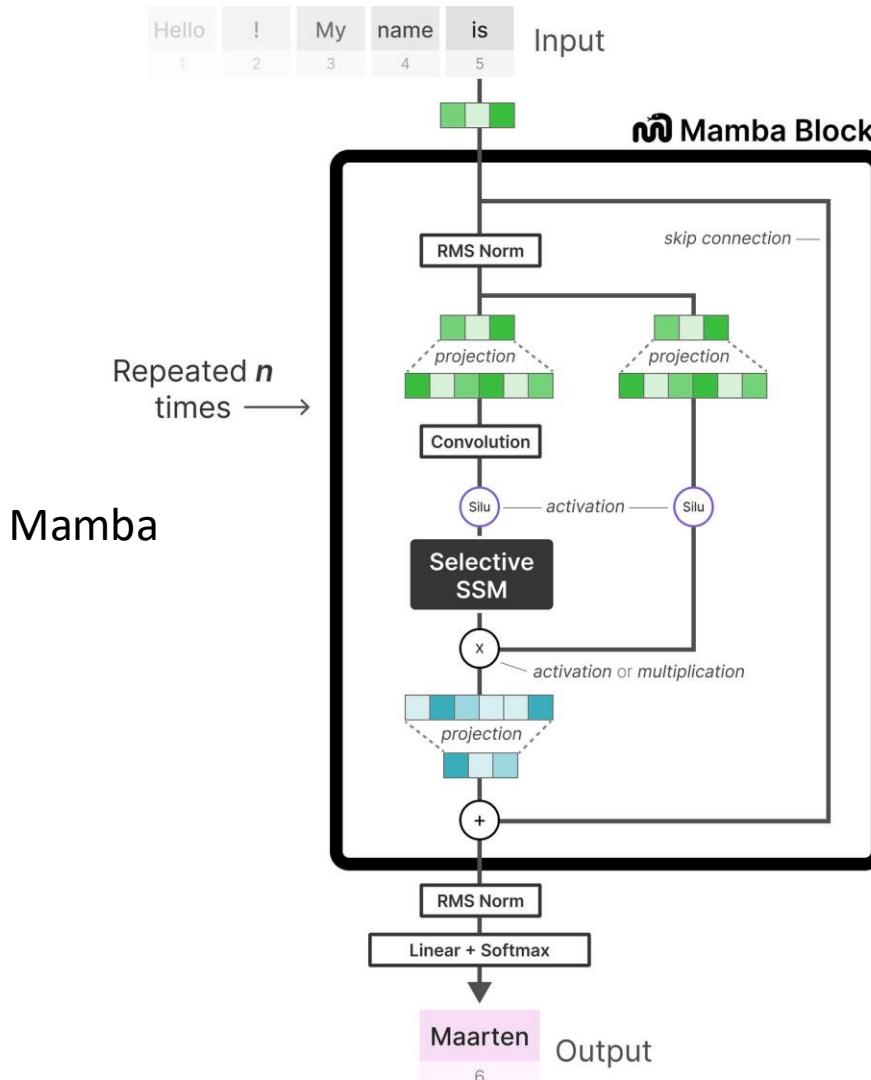
```
from transformers import AutoModelForCausalLM, AutoTokenizer
model_name = "Qwen/Qwen2.5-7B-Instruct"
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype="auto",
    device_map="auto"
)
tokenizer = AutoTokenizer.from_pretrained(model_name)

prompt = "Give me a short introduction to large language model."
messages = [
    {"role": "system", "content": "You are Qwen, created by Alibaba Cloud. You are a helpful assistant."},
    {"role": "user", "content": prompt}
]
text = tokenizer.apply_chat_template(
    messages,
    tokenize=False,
    add_generation_prompt=True
)
model_inputs = tokenizer([text], return_tensors="pt").to(model.device)

generated_ids = model.generate(
    **model_inputs,
    max_new_tokens=512
)
generated_ids = [
    output_ids[len(input_ids):] for input_ids, output_ids in
    zip(model_inputs.input_ids, generated_ids)
]

response = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

Different Architectures

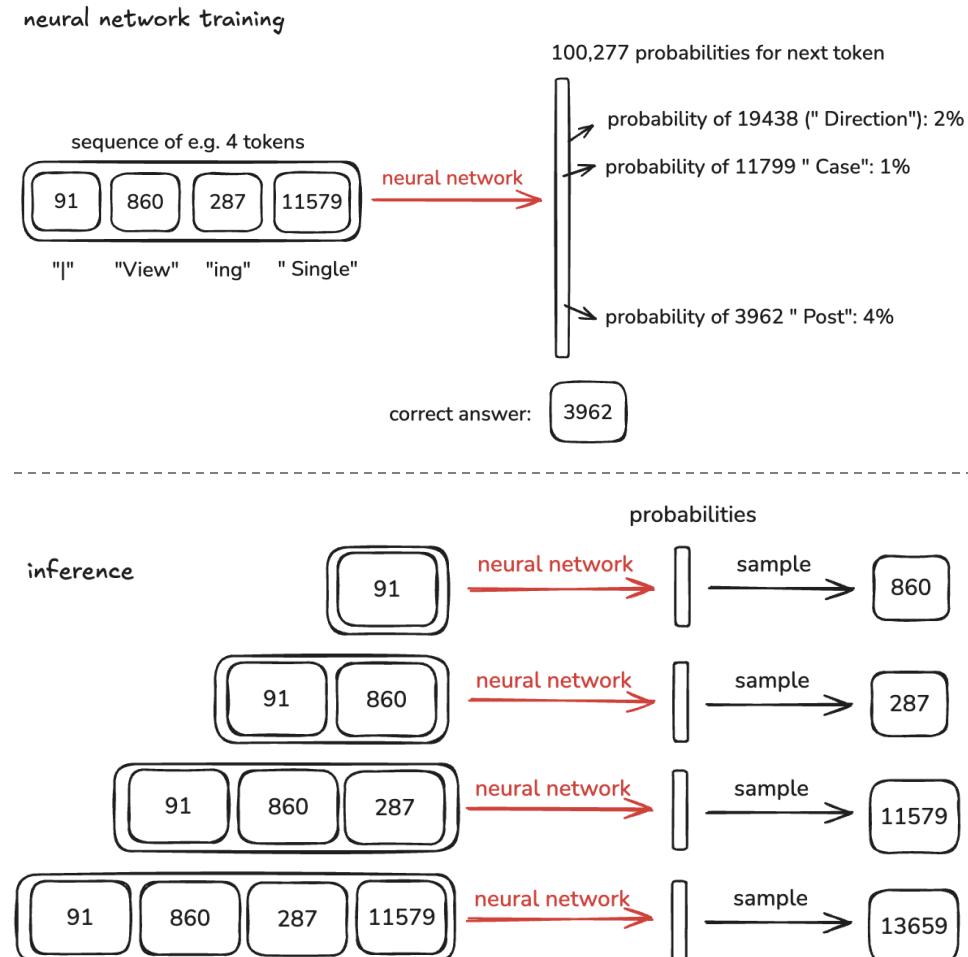


Pretraining

- Training a Neural Network
- Traditional Training vs Pretraining
- Core Implementation
- NVIDIA GPU
- Memory Usage
- Pretraining Optimization
- Lingua

Training a Neural Network

- **Prepare Data** – Collect, preprocess, and split data into training, validation, and test sets.
- **Define Model** – Choose a neural network architecture (e.g., MLP, CNN, Transformer) and initialize parameters.
- **Select Loss Function & Optimizer** – Define a loss function (e.g., Cross-Entropy, MSE) and an optimizer (e.g., SGD, Adam).
- **Train the Model** – Iterate over multiple epochs, performing forward propagation, loss computation, and backpropagation to update weights.
- **Evaluate** – Assess model performance on validation/test data.



Deep Dive into LLMs like ChatGPT, Andrej Karpathy

Traditional Training vs Pretraining

Feature	Traditional Training	Pretraining
Objective	Learn task-specific patterns from scratch	Learn general representations or features
Dataset	Labeled task-specific dataset	Large, diverse, unlabeled (self-supervised)
Training Time	Shorter (depends on dataset size)	Longer (days to weeks to months)
Generalization	Low (trained for a specific task only)	High (can be adapted to many tasks)
Fine-tuning?	No fine-tuning, trained end-to-end	Yes, adapted to downstream tasks No, Zero-shot prompting

Core Implementation

Define Model

```
import torch
import torch.nn as nn
from transformers import (
    AutoConfig,
    AutoModelForCausalLM,
    AutoTokenizer,
    Trainer,
    TrainingArguments,
    DataCollatorForLanguageModeling
)
from datasets import load_dataset

# 1. Define model configuration (e.g., GPT-2 from scratch)
model_name = "gpt2" # Base model name (used for tokenizer)
config = AutoConfig.from_pretrained(model_name) # Load GPT-2
config (modify if needed)
model = AutoModelForCausalLM(config) # Initialize model with
random weights
```

Define Data

```
# 3. Load dataset (WikiText-2 for language modeling)
dataset = load_dataset("wikitext", "wikitext-2-raw-v1")
```

Define
Training Args

```
# 5. Define training arguments
training_args = TrainingArguments(
    output_dir="./llm_pretraining_scratch",
    per_device_train_batch_size=8,
```

Start Training

```
num_train_epochs=5, # Increase for full pretraining
save_steps=1000,
logging_steps=500,
evaluation_strategy="epoch",
save_total_limit=2,
learning_rate=3e-4, # Typically higher when training from
scratch
weight_decay=0.01,
report_to="none"
)
```

```
# 6. Data Collator for Causal Language Modeling
data_collator = DataCollatorForLanguageModeling(
    tokenizer=tokenizer,
    mlm=False # Causal LM (for autoregressive models like GPT)
)
```

```
# 7. Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    tokenizer=tokenizer,
    data_collator=data_collator
)
```

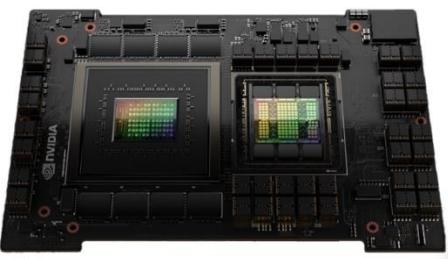
```
# 8. Train the model from scratch
trainer.train()
```

```
trainer.save_model("./llm_pretrained_from_scratch")
tokenizer.save_pretrained("./llm_pretrained_from_scratch")

print("LLM Pretraining from Scratch Complete!")
```



- NVIDIA GPUs are high-performance graphics processing units designed for computing tasks such as gaming, AI, deep learning, and scientific simulations, offering powerful parallel processing capabilities.



H100

NVIDIA A100 TENSOR CORE GPU SPECIFICATIONS (SXM4 AND PCIe FORM FACTORS)

	A100 40GB PCIe	A100 80GB PCIe	A100 40GB SXM	A100 80GB SXM
FP64		9.7 TFLOPS		
FP64 Tensor Core		19.5 TFLOPS		
FP32		19.5 TFLOPS		
Tensor Float 32 (TF32)		156 TFLOPS 312 TFLOPS*		
BFLOAT16 Tensor Core		312 TFLOPS 624 TFLOPS*		
FP16 Tensor Core		312 TFLOPS 624 TFLOPS*		
INT8 Tensor Core		624 TOPS 1248 TOPS*		
GPU Memory	40GB HBM2	80GB HBM2e	40GB HBM2	80GB HBM2e
GPU Memory Bandwidth	1,555GB/s	1,935GB/s	1,555GB/s	2,039GB/s
Max Thermal Design Power (TDP)	250W	300W	400W	400W
Multi-Instance GPU	Up to 7 MIGs @ 5GB	Up to 7 MIGs @ 10GB	Up to 7 MIGs @ 5GB	Up to 7 MIGs @ 10GB
Form Factor	PCIe		SXM	
Interconnect	NVIDIA® NVLink® Bridge for 2 GPUs: 600GB/s ** PCIe Gen4: 64GB/s		NVLink: 600GB/s PCIe Gen4: 64GB/s	
Server Options	Partner and NVIDIA-Certified Systems™ with 1-8 GPUs		NVIDIA HGX™ A100-Partner and NVIDIA-Certified Systems with 4,8, or 16 GPUs NVIDIA DGX™ A100 with 8 GPUs	

* With sparsity

** SXM4 GPUs via HGX A100 server boards; PCIe GPUs via NVLink Bridge for up to two GPUs

SPECIFICATIONS

	H100 SXM	H100 PCIe
FP64	34 TFLOPS	26 TFLOPS
FP64 Tensor Core	67 TFLOPS	51 TFLOPS
FP32	67 TFLOPS	51 TFLOPS
TF32 Tensor Core	989 TFLOPS*	756 TFLOPS*
BFLOAT16 Tensor Core	1,979 TFLOPS*	1,513 TFLOPS*
FP16 Tensor Core	1,979 TFLOPS*	1,513 TFLOPS*
FP8 Tensor Core	3,958 TFLOPS*	3,026 TFLOPS*
INT8 Tensor Core	3,958 TOPS*	3,026 TOPS*
GPU memory	80GB	80GB
GPU memory bandwidth	3.35TB/s	2TB/s
Decoders	7 NVDEC 7 JPEG	7 NVDEC 7 JPEG
Max thermal design power (TDP)	Up to 700W (configurable)	300-350W (configurable)
Multi-Instance GPUs	Up to 7 MIGs @ 10GB each	
Form factor	SXM	PCIe dual-slot air-cooled
Interconnect	NVLink: 900GB/s PCIe Gen5: 128GB/s	NVLink: 600GB/s PCIe Gen5: 128GB/s
Server options	NVIDIA HGX™ H100 partner and NVIDIA-Certified Systems™ with 4 or 8 GPUs NVIDIA DGX™ H100 with 8 GPUs	Partner and NVIDIA-Certified Systems with 1-8 GPUs
NVIDIA AI Enterprise	Add-on	Included

* Shown with sparsity. Specifications 1/2 lower without sparsity.

https://www.megware.com/fileadmin/user_upload/LandingPage%20NVIDIA/nvidia-h100-datasheet.pdf

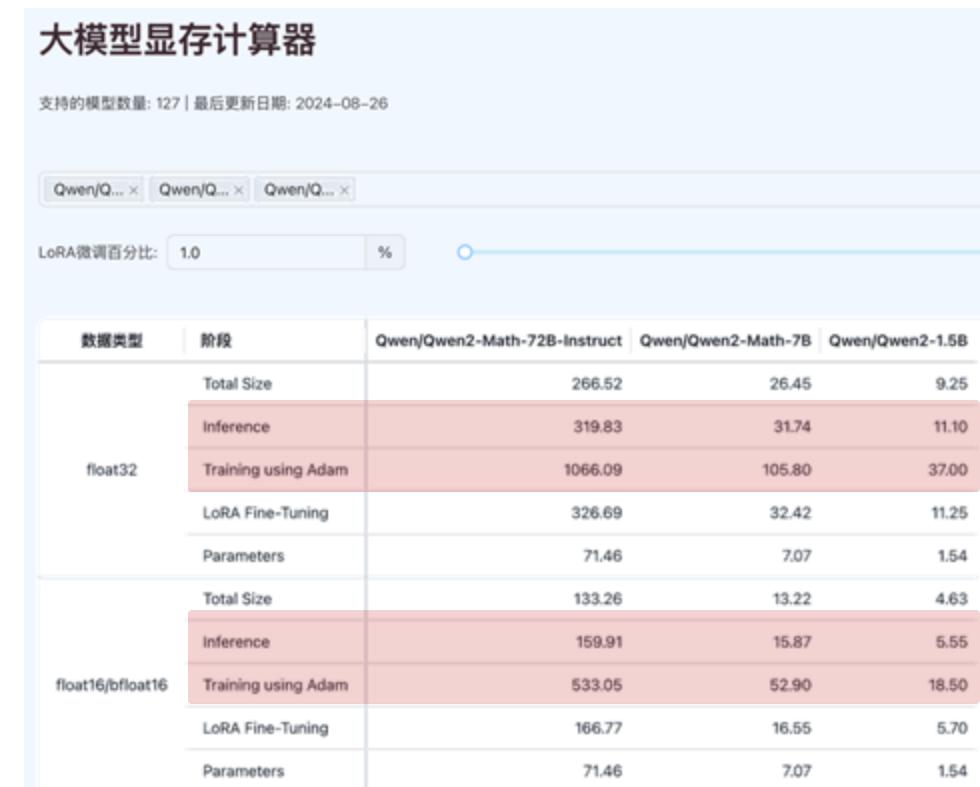
<https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf>

Memory Usage

- The table shows the GPU memory required for inference, while the memory required for training is approximately four times that of inference.

	FP32	FP16	INT8
BERT-base-uncased-0.1B	0.5 GB	0.24 GB	0.12 GB
GPT-2-0.1B	0.56 GB	0.28 GB	0.14 GB
TinyLlama-1.1B-Chat	4.6 GB	2.3 GB	1.1 GB
Microsoft Phi-2	11.8 GB	6 GB	3 GB
Mistral-7B-v0.2	33 GB	16.5 GB	8.25 GB
Llama-3-8B-Instruct	33 GB	17 GB	8.5 GB
Llama-3-70B-Instruct	311 GB	155 GB	77 GB
Qwen2-72B-Instruct	322 GB	161 GB	80 GB
Mistral-8x22B-v0.1	631 GB	316 GB	158 GB
Bloom-176B	787 GB	394 GB	200 GB

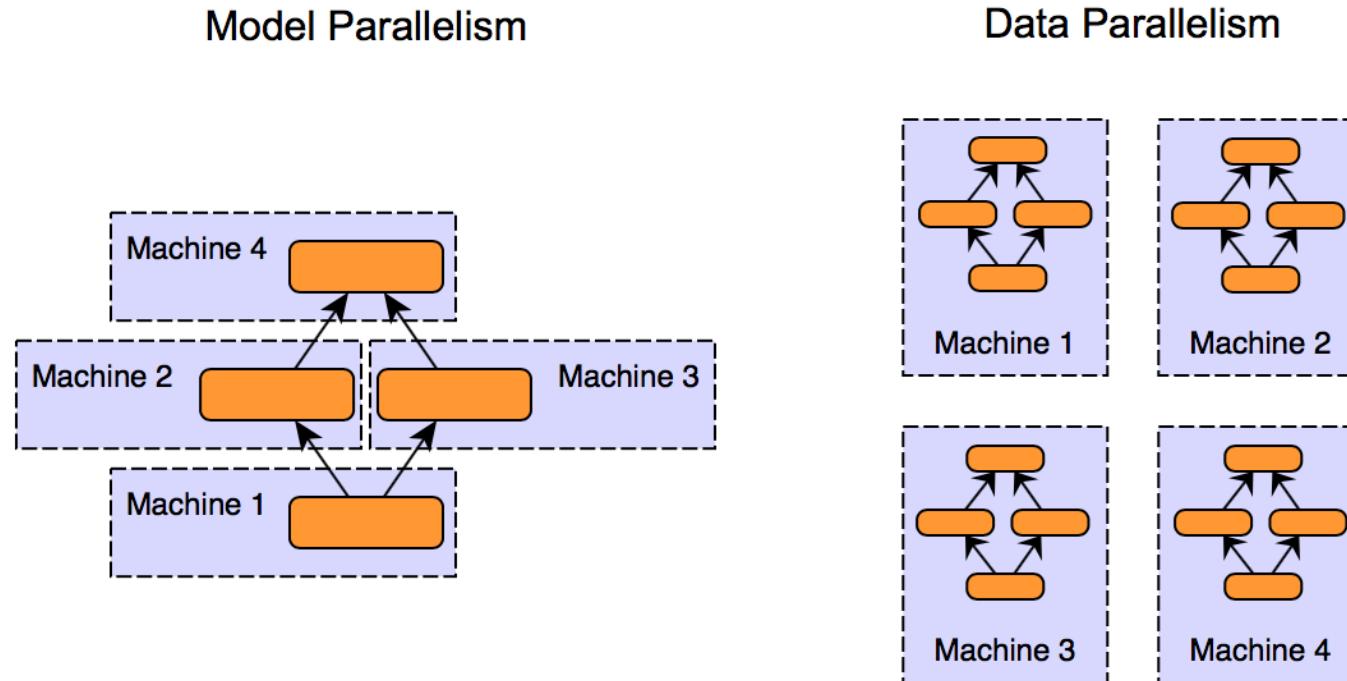
Assumption: Text length is 512, batch size is 8, and the number of GPUs is 1.



<https://www.llamafactory.cn/gpu-memory-estimation.html>

Pretraining Optimization

- **Model Parallelism** – Splits a large model across multiple GPUs, distributing different layers or partitioning individual layers (e.g., tensor parallelism) to handle memory constraints.
- **Data Parallelism** – Duplicates the entire model across multiple GPUs, but each GPU processes a different mini-batch of data, synchronizing gradients after each step.



<https://medium.com/@minhahanh.dongnguyen/megatron-lm-how-model-parallelism-is-pushing-language-models-to-new-heights-c21a5343e06a>

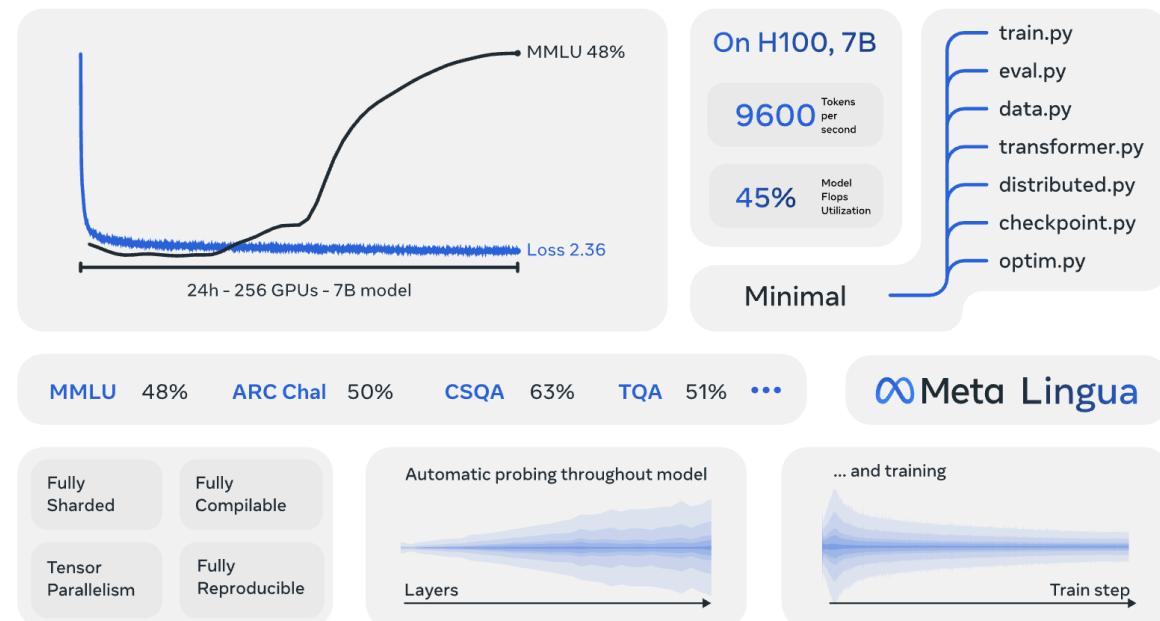
Lingua

- Meta Lingua is a lightweight, efficient codebase developed by Meta AI for training and inference of large language models (LLMs).
- Designed with research in mind, it utilizes modular PyTorch components, enabling researchers to experiment with new architectures, loss functions, and datasets with ease.
- This self-contained platform facilitates rapid prototyping and scalability, making it accessible for those with limited computational resources.

Meta Lingua

Mathurin Videau*, Badr Youbi Idrissi*, Daniel Haziza, Luca Wehrstedt, Jade Copet, Olivier Teytaud, David Lopez-Paz. *Equal and main contribution

Meta Lingua is a minimal and fast LLM training and inference library designed for research. Meta Lingua uses easy-to-modify PyTorch components in order to try new architectures, losses, data, etc. We aim for this code to enable end to end training, inference and evaluation as well as provide tools to better understand speed and stability. While Meta Lingua is currently under development, we provide you with multiple `apps` to showcase how to use this codebase.



<https://github.com/facebookresearch/lingua>

Instruction Tuning

- Instruction Tuning
- The Role of Instruction Tuning
- Instruction Data
 - Text
 - Math
 - Code
 - Multi-modal
- Instruction Tuning Loss
- Llama-factory

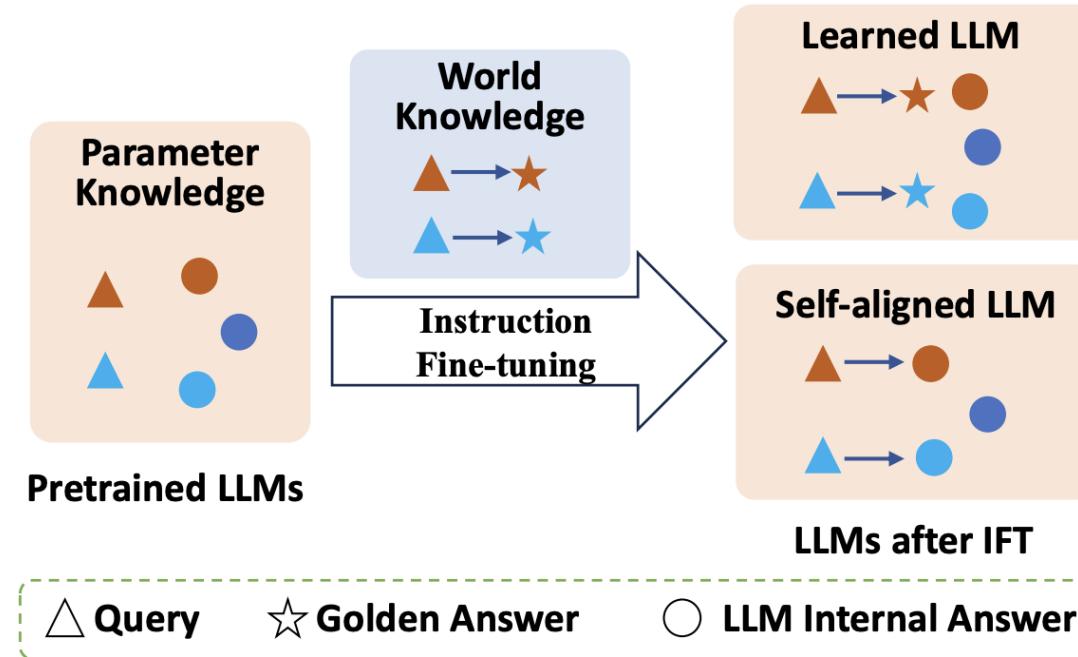
Instruction Tuning or Supervised Fine-tuning (SFT)

- **Instruction Tuning:** Fine-tunes a pretrained model using instruction-based datasets to improve task-following abilities.

Feature	Pretraining	Instruction Tuning
Objective	Train the model on a large-scale corpus to learn general language patterns	Make the model better at following human instructions across multiple tasks
Training Data	Large, unlabeled, diverse text (e.g., books, Wikipedia, web data)	Instruction-based datasets with input-output pairs (e.g., “Summarize this text”)
Learning Type	Self-supervised learning (e.g., Masked LM, Causal LM)	Supervised fine-tuning on human-written instructions
Generalization	Learns broad language representations	Adapts to task-following behaviors and improves zero-shot/few-shot abilities
Task-Specific?	No, learns from raw text without explicit tasks	Yes, explicitly trained to understand instructions across multiple domains
Example	Training GPT from scratch on massive web data	Fine-tuning GPT on an instruction dataset like FLAN, Super-NaturalInstructions
Example Models	GPT-3, BERT, T5 (trained from scratch)	InstructGPT, ChatGPT, FLAN-T5, GPT-4

The Role of Instruction Tuning

- Instruction tuning aligns queries with knowledge already in LLMs with similar behavioral norms



Instruction Data (Text)

Input

Explain tradeoffs between fiscal and monetary policy as tools in a nation's economic toolkit. Provide examples of past instances when each were utilized, the economic conditions that led to them being deployed, their intended effects, and an evaluation of their relative efficacy and consequences.

Output

Fiscal and monetary policy are the two main tools that governments have to influence economic activity. They each have benefits and drawbacks. Fiscal policy refers to government spending and taxation decisions. Examples of fiscal policy include:

- During the Great Recession, the U.S. government implemented a fiscal stimulus through the American Recovery and Reinvestment Act of 2009. This included increased spending on infrastructure, tax cuts, and expanded unemployment benefits. The intention was to boost aggregate demand and stimulate economic activity. Studies have found that the stimulus had a positive but modest impact on economic growth.
- The Trump administration implemented tax cuts in 2017 with the goal of increasing business investment and growth. However, the effect on growth has been debated, and the tax cuts significantly increased the federal budget deficit.

Monetary policy refers to decisions by a central bank, like the Federal Reserve, about interest rates and the money supply. Examples of monetary policy include:

- After the 2008 financial crisis, the Fed lowered interest rates to near zero and implemented quantitative easing programs to increase liquidity. The intention was to boost lending, investment, and consumer spending. These actions are credited with helping the economy recover.
- In the late 1990s, the Fed raised interest rates to reduce inflationary pressures from economic growth. Higher rates contributed to slowing the economy and avoiding a spike in prices.

The key tradeoffs are:.....

Instruction Data (Math)

Input

I have 5 marbles numbered 1 through 5 in a bag. Suppose I take out two different marbles at random. What is the expected value of the sum of the numbers on the marbles?

Output

There are $\binom{5}{2} = 10$ different pairs of marbles can be drawn, and the expected value of the sum is the average of the sums of each pair. This is
$$\frac{1}{10}((1+2)+(1+3)+(1+4)+(1+5)+(2+3)+(2+4)+(2+5)+(3+4)+(3+5)+(4+5)) = \frac{60}{10} = 6.$$

Input

A standard deck of 52 cards has 13 ranks (Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King) and 4 suits (\spadesuit , \heartsuit , \diamondsuit , and \clubsuit), such that there is exactly one card for any given rank and suit. Two of the suits (\spadesuit and \clubsuit) are black and the other two suits (\heartsuit and \diamondsuit) are red. The deck is randomly arranged. What is the probability that the top three cards are all \spadesuit s?

Output

There are 13 ways to choose the first card to be a \spadesuit , then 12 ways to choose the second card to be another \spadesuit , then 11 ways to choose the third card to be a \spadesuit . There are $52 \times 51 \times 50$ ways to choose any three cards. So the probability is $\frac{13 \times 12 \times 11}{52 \times 51 \times 50} = \boxed{\frac{11}{850}}.$

Instruction Data (Code)

Input

Write me Python code is a program to read, parse, and display thermal data from the Topdon TC001 Thermal camera. It includes detailed information about the program, such as its author, date of creation, and purpose. The code also includes a list of key bindings for controlling the program, such as increasing/decreasing blur, changing the interpolated scale, and toggling the HUD. The code also includes a function to detect if the program is running on a Raspberry Pi, as OpenCV behaves differently on different builds. The code then initializes the video capture device and sets the appropriate properties, such as not converting the video to RGB to preserve the temperature data. The code then sets the initial values for various parameters, such as the window size, scale, and contrast. It also creates a named window for displaying the thermal data and sets up the font and color map for the display. The main loop of the program then begins, which includes code for reading the frames from the video capture device, parsing the thermal data, and displaying the data on the screen. The code also includes code for handling key presses, such as changing the blur radius, contrast, and color map, and for recording and taking snapshots of the thermal data.

Output

```
#!/usr/bin/env python3\n\""\nLes Wright 21 June 2023\nhttps://youtube.com/leslaboratory\nA Python\nprogram to read, parse and display thermal data from the Topdon TC001 Thermal camera!\n\""\nprint('Les\nWright 21 June 2023')\nprint('https://youtube.com/leslaboratory')\nprint('A Python program to read, parse\nand display thermal data from the Topdon TC001 Thermal camera!')\nprint("")\nprint('Tested on Debian all\nfeatures are working correctly')\nprint('This will work on the Pi However a number of workarounds are\nimplemented!')\nprint('Seemingly there are bugs in the compiled version of cv2 that ships with the Pi...')
```

Instruction Data (Multi-modal)

Input



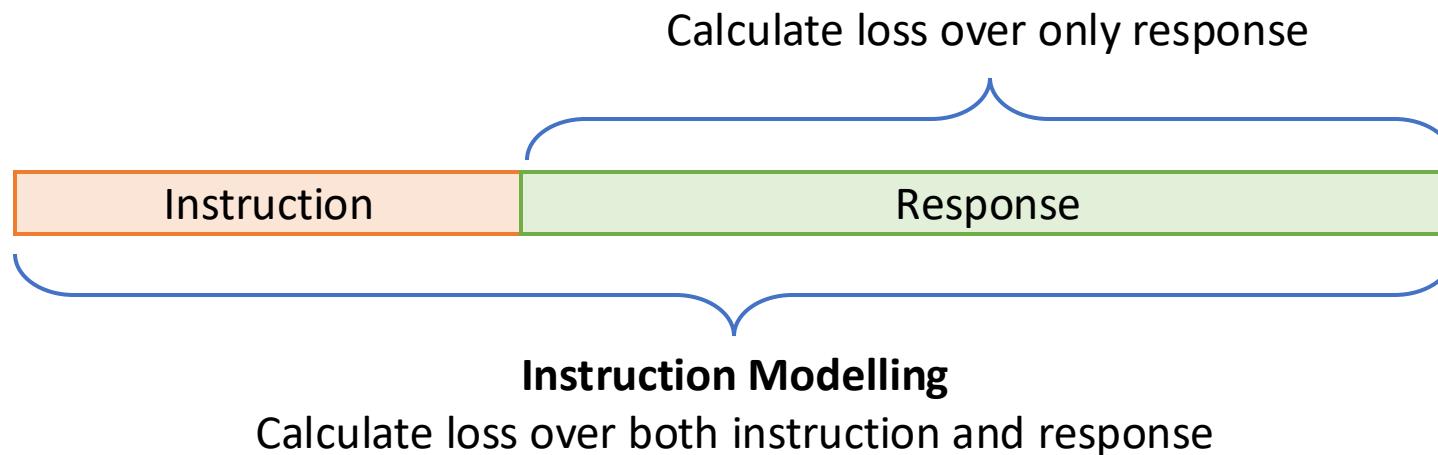
What sport is this?

Output

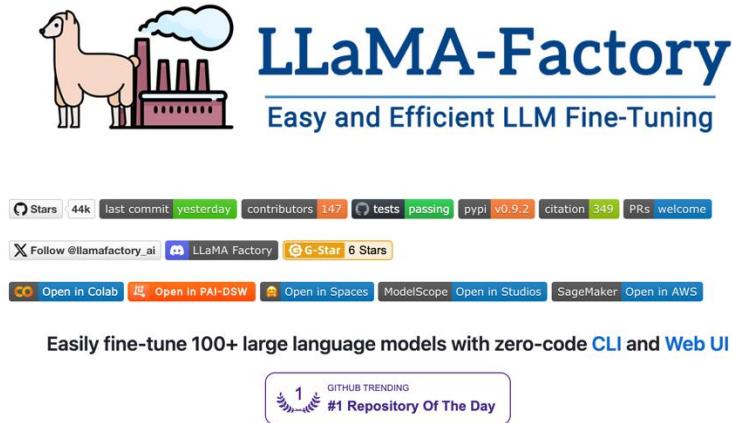
a baseball game

Instruction Tuning Loss

- Two key factors affecting the effectiveness of Instruction Tuning (IM):
 - The ratio between instruction length and output length in the training data. IM is particularly effective when instructions are long while outputs are short.
 - The number of training samples. IM performs better when the number of training samples is small.



Llama-factory



Supported Training Approaches

Approach	Full-tuning	Freeze-tuning	LoRA	QLoRA
Pre-Training	✓	✓	✓	✓
Supervised Fine-Tuning	✓	✓	✓	✓
Reward Modeling	✓	✓	✓	✓
PPO Training	✓	✓	✓	✓
DPO Training	✓	✓	✓	✓
KTO Training	✓	✓	✓	✓
ORPO Training	✓	✓	✓	✓
SimPO Training	✓	✓	✓	✓

Llama3-SFT-Lora

```
### model
model_name_or_path: meta-llama/Meta-Llama-3-8B-Instruct
trust_remote_code: true

### method
stage: sft
do_train: true
finetuning_type: lora
lora_rank: 8
lora_target: all

### dataset
dataset: identity,alpaca_en_demo
template: llama3
cutoff_len: 2048
max_samples: 1000
overwrite_cache: true
preprocessing_num_workers: 16
dataloader_num_workers: 4

### output
output_dir: saves/llama3-8b/lora/sft
logging_steps: 10
save_steps: 500
plot_loss: true
overwrite_output_dir: true
save_only_model: false

### train
per_device_train_batch_size: 1
gradient_accumulation_steps: 8
learning_rate: 1.0e-4
num_train_epochs: 3.0
lr_scheduler_type: cosine
warmup_ratio: 0.1
bf16: true
ddp_timeout: 180000000
resume_from_checkpoint: null
```

Define source model

Define finetune method

Define dataset

```
llamafactory-cli train llama3_lora_sft.yaml
```

Define tuning args

Reinforcement Learning from Human Feedback

- AI Safety
- Reinforcement Learning from Human Feedback
- PPO and DPO
- DPO Implementation
- OpenRLHF
- Post-Training

AI Safety

- AI safety is crucial to ensure that artificial intelligence systems operate reliably, ethically, and without unintended harmful consequences, protecting both individuals and society as AI becomes more powerful and autonomous.

INSIGHTS

Taylor Swift highlights harmful AI use: a tale of two “deepfakes”

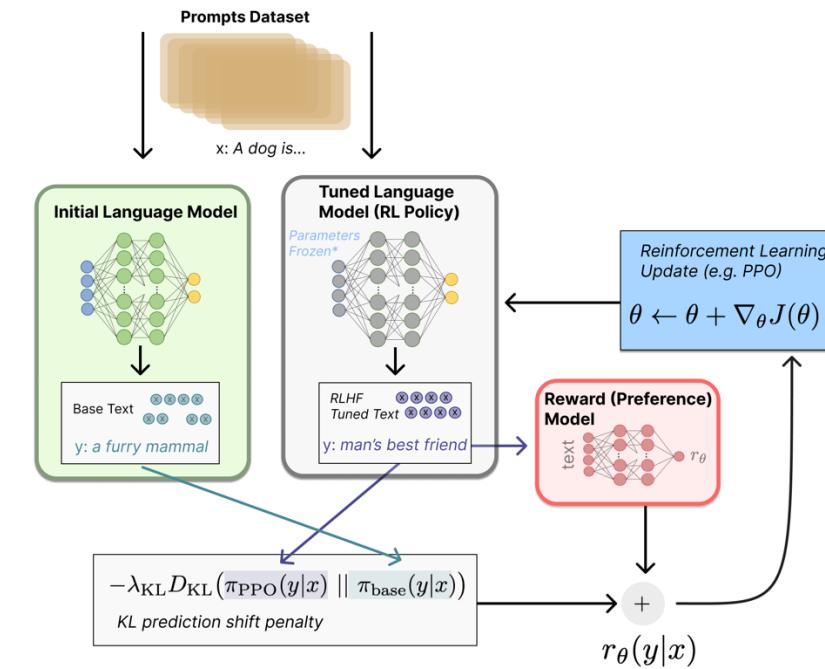
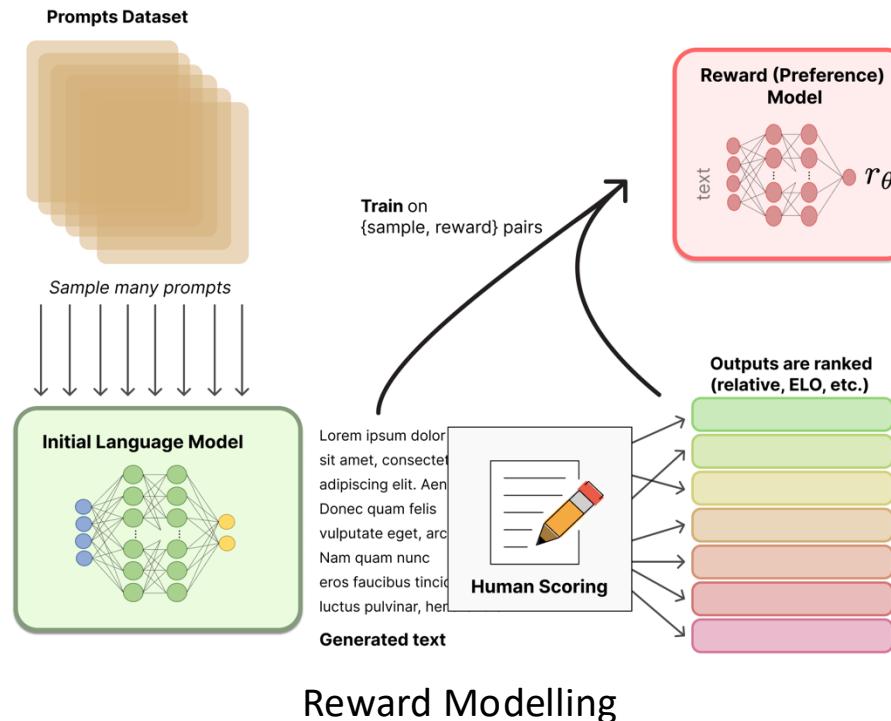
DATE PUBLISHED
13 February 2024

READ TIME
8 mins

The recent incident involving fake explicit images of singer Taylor Swift has brought the challenge of ‘deepfakes’ once again into the mainstream. The incident, along with a second, lesser-known “deepfake” debacle following Swift’s recent win at the Grammy Awards, provides apt (albeit unfortunate) grounds to explore the evolving meaning of deepfakes and the regulatory responses to their harmful distribution online.

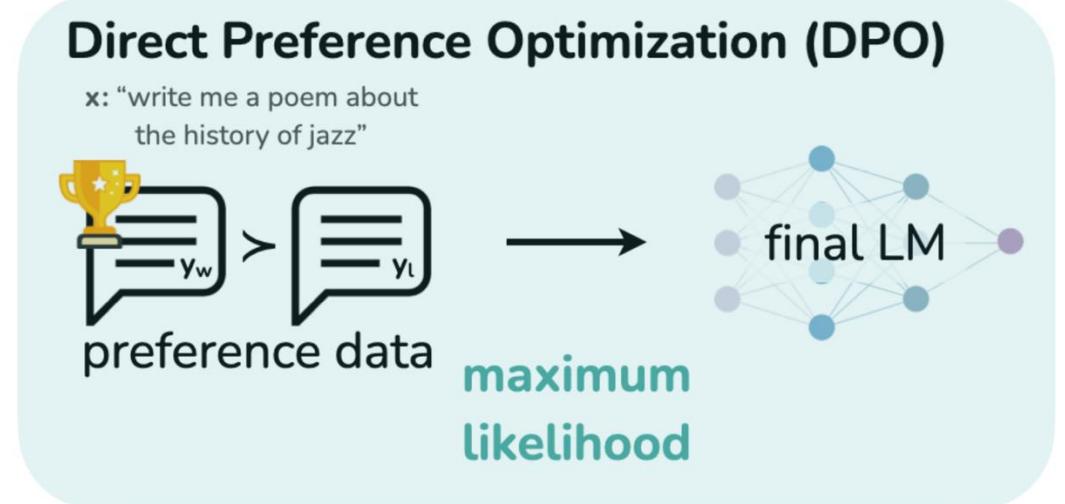
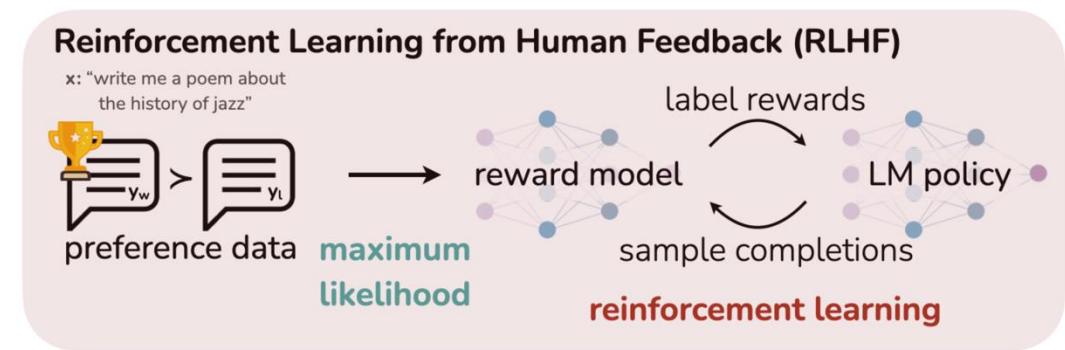
Reinforcement Learning from Human Feedback (RLHF)

- Reinforcement Learning from Human Feedback (RLHF) is a technique that fine-tunes AI models by incorporating **human preferences** to **improve alignment with human values and expectations**.



PPO and DPO

- Direct Preference Optimization (DPO) is an alternative to Reinforcement Learning from Human Feedback (RLHF) that fine-tunes language models based on human preferences **without requiring a reward model or reinforcement learning**.
- **Collect human preference data** – Given two model responses, humans choose the preferred one.
- **Optimize the model directly** – Instead of training a reward model + PPO. DPO directly optimizes the model to increase the probability of preferred responses while decreasing the probability of rejected ones.
- **More stable training** – DPO avoids the instabilities and high compute costs of RL-based fine-tuning.



DPO Implementation

```
● ● ●  
from datasets import load_dataset  
from trl import DPOConfig, DPOTrainer  
from transformers import AutoModelForCausalLM, AutoTokenizer  
  
model = AutoModelForCausalLM.from_pretrained("Qwen/Qwen2-0.5B-Instruct")  
tokenizer = AutoTokenizer.from_pretrained("Qwen/Qwen2-0.5B-Instruct")  
train_dataset = load_dataset("trl-lib/ultrafeedback_binarized",  
    split="train")  
  
training_args = DPOConfig(output_dir="Qwen2-0.5B-DPO", logging_steps=10)  
trainer = DPOTrainer(model=model, args=training_args,  
    processing_class=tokenizer, train_dataset=train_dataset  
    trainer.train()
```

← DPO Training

https://huggingface.co/docs/trl/main/dpo_trainer

OpenRLHF

- OpenRLHF is a high-performance RLHF framework built on Ray, DeepSpeed and HF Transformers.

- **Ray** helps distribute RL training across multiple GPUs/CPUs.
- **DeepSpeed** optimizes large-scale model training, making RLHF more memory-efficient.
- **HF Transformers** provide the LLM backbone for fine-tuning.



contributors 58 issues 170 open discussions 11 total pull requests 13 open Stars 5.6k

Open-source / Comprehensive / Lightweight / Easy-to-use

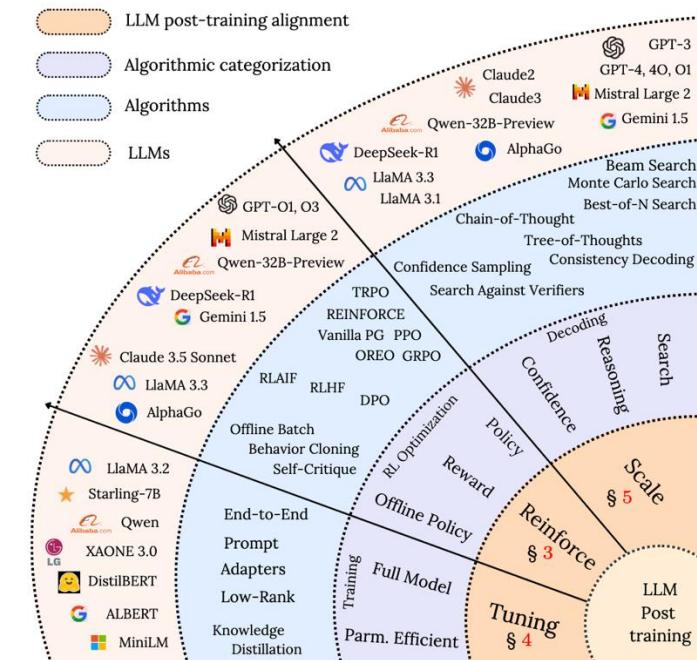
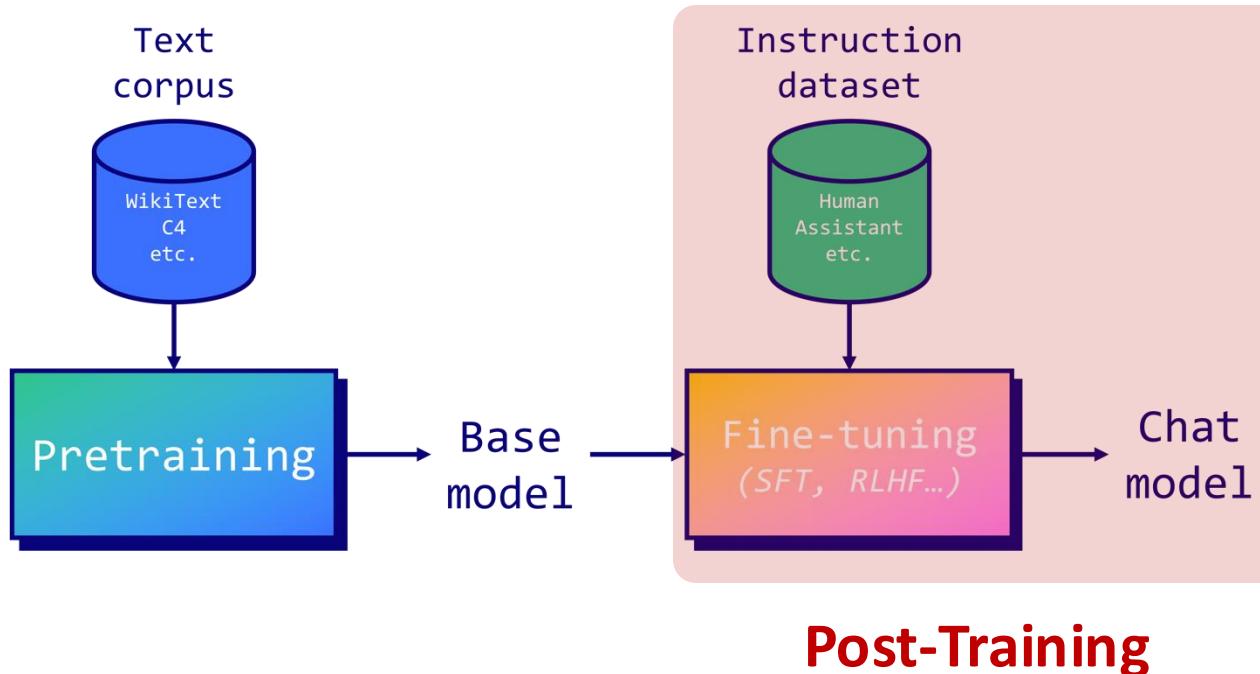
[English | [中文](#) | [日本語](#)]

OpenRLHF is a high-performance RLHF framework built on Ray, DeepSpeed and HF Transformers:

- **Simple and easy to use:** OpenRLHF is one of the simplest high-performance RLHF libraries currently available, and seamlessly compatible with Huggingface models and datasets.
- **High performance:** RLHF training spends 80% of the time on the sample generation stage. Thanks to the ability to use a large inference batch size with Ray and Packing Samples and vLLM generation acceleration, the performance of OpenRLHF 3~4x+ that of Optimized DeepSpeedChat with Hybrid Engine.
- **Distributed RLHF:** OpenRLHF distribute the Actor, Reward, Reference, and Critic models onto separate GPUs using Ray, while placing the Adam optimizer on the CPU. This enables full-scale fine-tuning of 70B+ models with multiple A100 80G GPUs and vLLM and 7B models across multiple 24GB RTX 4090 GPUs.
- **Hybrid Engine:** OpenRLHF also supports the hybrid engine, allowing all models and vLLM engines to share the GPUs to avoid GPU idling.
- **PPO Implementation Optimization:** We integrated the implementation tricks for PPO to improve the training stability, referencing [Zhihu](#) and [Advanced Tricks for Training Large Language Models with Proximal Policy Optimization](#).

Post-Training

- Post-training of LLMs refines pretrained models through techniques like supervised fine-tuning, instruction tuning, RLHF, DPO, and quantization to enhance alignment, efficiency, and task performance.



Evaluation

- Benchmarks
- Cantonese Benchmark
- GSM-PLUS
- TMGBench
- VL-RewardBench
- Opencompass

Benchmarks

Benchmark	Focus Area	Description	Example Metrics
MMLU (Massive Multitask Language Understanding)	General knowledge & reasoning	57 subjects covering STEM, humanities, social sciences, etc.	Accuracy (%)
HellaSwag	Commonsense reasoning	Tests everyday scenario understanding	Accuracy (%)
ARC (AI2 Reasoning Challenge)	Logical reasoning	Grade-school-level science and reasoning questions	Accuracy (%)
GSM8K (Grade School Math 8K)	Math reasoning	Solving elementary school-level math problems	Accuracy (%)
MATH	Advanced math	Tests high-school & olympiad-level math ability	Accuracy (%)
BBH (BigBench Hard)	Complex reasoning	Harder subset of BigBench tasks, including ethics, social dynamics	Accuracy (%)
TruthfulQA	Truthfulness	Measures resistance to misinformation and factual consistency	Truthfulness Score (%)
MT-Bench	Multi-turn chat	Evaluates LLMs in a conversational multi-turn dialogue setting	Score (1-10)
HumanEval	Code generation	Tests LLMs' ability to write functional code	Pass@1 (%)
MBPP (Multi-turn Python Benchmark)	Python programming	Evaluates Python code generation for multi-turn problem-solving	Pass@1 (%)
OpenAI's Chatbot Arena	Overall LLM ranking	Human preference ranking of chatbot responses	Elo Score
AGIEval	Human-like intelligence	Measures model performance on human exams (SAT, GRE, LSAT, etc.)	Score (%)
SuperGLUE	NLP general tasks	Evaluates performance across a variety of NLP tasks	Accuracy (%)
TydiQA	Multilingual QA	Tests question-answering ability in multiple languages	F1 Score

Cantonese Benchmark

Designed to evaluate LLM performance in factual generation, mathematical logic, complex reasoning, and general knowledge in Cantonese, which aim to advance open-source Cantonese LLM technology.

question string · lengths	answer string · lengths
 Janet每日经常有16枚蛋。渠每朝早食三只作早餐，同埋每日用四只喺磨丁饼制作俾佢慨朋友。每日渠会将余下慨蛋以每只\$2慨价钱喺农货市场出售。渠每日喺农货市场赚几多钱？	 Janet每日卖 $16 - 3 - 4 = 9$ 只鸭蛋。渠每日系农夫市场赚 $\$9 \times 2 = \18 。 #####18
一个长袍需要2卷蓝色纤维同半卷白色纤维。总共需要几卷布？	要制作一件白色的布料，需要 $2/2=1$ 条白色纤维，所以总共需要 $2+1=3$ 条布料。 #####3
Josh决定装修一间屋。渠买咗一间屋，价钱系\$80,000，再加上\$50,000慨维修费用。呢个令到呢间屋慨价值增加咗150%。渠最终赚咗几多钱？	间屋慨成本同埋维修费系 $80000 + 50000 = 130000$ ，渠增加咗间屋慨价值系 $80000 \times 1.5 = 120000$ ，所以间屋慨新价值系 $120000 + 80000 = 200000$ ，所以渠赚咗 $\$200000 - 130000 = 70000$ 。 #####70000
James决定每周跑3次短跑，每次跑60米。渠每星期总共跑咗几多米？	渠一个星期跑三次，所以渠跑 $3 \times 60 = 180$ 米。 #####180
Wendi每日喂饲渠慨鸡，每只鸡都要食三杯混合慨鸡饲料，入面有籽、虫虫同菜籽，帮助渠哋保持健康。渠一日分三次喂啖鸡食。早上渠会喂渠慨鸡15杯慨饲料。下午又系25杯。如果Wendi慨鸡群有20只，最后一餐渠要喂几多杯慨饲料俾佢哋？	如果每只鸡每日吃3杯饲料，咁20只鸡每日就要用 $3 \times 20 = 60$ 杯饲料。如果渠早上喂15杯，而下午喂25杯，咁最后一餐就需要 $60 - 15 - 25 = 20$ 杯鸡饲料。 #####20
Kylar去到商店买杯系渠慨新公寓慨。一个杯系\$5，但每第二个杯只系原价慨60%。 Kylar想买16个杯。渠要支付多少钱？	一杯嘅打折后价钱系 $60 / 100 \times 5 = \$3$ 。如果每第二杯嘅平啲，咁就系Kylar会买 $16 / 2 = 8$ 杯平啲慨杯。所以系平啲慨杯，Kylar就要俾 $8 \times 3 = \$24$ 。而系原价慨杯，Kylar就要俾 $8 \times 5 = \$40$ 。所以总共Kylar需要俾 $24 + 40 = \$64$ 。 #####64
如果西雅图有20只羊，查尔斯顿有西雅图的4倍，图卢兹有查尔斯顿的两倍。所以图卢茵、查尔斯顿和西雅图总共有多少只羊？	如果西雅图有20只羊，查尔斯顿就有 $20 \times 4 = 80$ 只羊，图卢兹有比查尔斯顿多两倍的羊，即为 $2 \times 80 = 160$ 只羊，总共三个地方有 $20 + 160 + 80 = 260$ 只羊。 #####260
Carla而家要download一个200GB慨档案。通常渠每分钟可以download 2GB，但系download到40%慨时候，Windows突然叫渠重新启动去安装更新，要用咗20分钟。之后Carla就要由零重新开始download。最终整个download慨过程需要几多时间？	先搵出40%档案有几个GB： $200GB \times 40\% = 80GB$ ，然后将呢个数字除以下载速率，搵出Windows重新启动慨时间： $80GB / 2GB/\text{分钟} = 40$ 分钟，之后搵出重新启动后完整下载档案慨时间： $200GB / 2GB/\text{分钟} = 100$ 分钟，再加上下载40%档案慨时间，完整下载档案慨时间，同埋...
John开车行驶3个钟头，每小时60英里的速度，但突然发现有啲好重要慨事系屋企，所以掉头返去。渠试图系4个钟头内返到屋企，但起初2个钟头都系坐喺堵塞慨交通度过。之后行车半个钟头，速度为30英里每小时，然后先得机会以每小时80英里慨速度行驶余下时间慨4...	渠转身时，渠离屋企有180英里远，喺最初慨四小时度，渠只可以开2个钟，系 $4 - 2 = 2$ 个钟，半个钟后，渠行15英里，之后再开1.5个钟，喺呢段时间，渠行咗120英里，所以渠总共行咗135英里，所以渠而家离屋企有45英里远。 #####45
Eliza每个星期工作慨前40个钟每小时收费系\$10。渠亦会收到加班费，加班费系渠正常小时收费慨1.2倍。如果Eliza呢个星期工作咗45个钟，囉周渠慨收入系几多？	Eliza有 $45 - 40 = 5$ 个小时的加班费。渠慨加班时薪系 $\$10 \times 1.2 = \12 。咁Eliza将会收到 $\$12 \times 5 = \60 个加班费。渠正常慨每周收入系 $\$10 \times 40 = \400 。所以，Eliza将会获得总共 $\$400 + \$60 = \$460$ 呢个星期慨工资。 #####460
呢个新慨程式喺第一个月慨下载量系60次。第二个月慨下载量系第一个月慨3倍，但系第三个月就减少咗30%。呢个程式喺三个月内总共冇几次下载？	第二个月慨下载次数增加到 $3 \times 60 = 180$ ，喺头两个月慨时候，呢个程式慨总下载次数系 $180 + 60 = 240$ ，第三个月慨时候，呢个程式慨下载次数 $30 / 100 \times 180 = 54$ ，第三个月慨下载次数系 $180 - 54 = 126$ ，三个月慨总下载次数系 $126 + 240 = 366$ #####366

GSM-PLUS

- Adversarial grade school math (GSM-PLUS) dataset, an extension of GSM8K augmented with various mathematical perturbations.

Seed Question: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Solution: Janet sells $16 - 3 - 4 = 9$ duck eggs a day. She makes $9 * 2 = 18$ every day at the farmer's market. Answer: 18

Perturbation Category	Question Variation
Numerical Variation	Num. Sub. $16 \rightarrow 20$ three → five four → six 2 → 3
	Digit Exp. $16 \rightarrow 1600$ four → 400
	IDF Conv. three → 1/4 2 → 2.5
Arithmetic Variation	Add. Op. Janet's ducks lay ... every day with four. She also uses two eggs to make a homemade hair mask every day. She sells ... make every day at the farmers' market?
	Rev. Op. Janet's ducks lay 16 eggs per day. She eats three ... with four. She sells the remainder at the farmers' market daily for a certain amount per fresh duck egg. She makes \$18 every day at the farmers' market. How much does each duck egg cost?
Problem Understanding	Janet's ducks lay 16 eggs daily. She eats three for breakfast and uses four to bake muffins for her friends. She sells the remaining eggs at the local farmers' market for \$2 per fresh duck egg. How much money does she make each day by selling eggs at the farmers' market?
Distractor Insertion	Janet's ducks ... with four. She also uses two eggs to feed her pet parrot, but her neighbor gives her two eggs from his own ducks to replace them. She sells ... at the farmers' market?
Critical Thinking	Janet's ducks lay eggs per day. She eats three for breakfast every morning and ... How much in dollars does she make every day at the farmers' market?

TMGBench

- A benchmark for evaluating the performance of large language models in game theory scenarios.

Three Atomic Games

Prisoner's Dilemma		Cooperate	Defect
Cooperate	(3, 3)	(0, 5)	
Defect	(5, 0)	(1, 1) <small>NE</small>	

Description: Two prisoners must independently decide whether to cooperate or betray (defect). Betrayal offers a higher payoff if the other cooperates, but mutual betrayal leads to the worst outcome for both.

Stag Hunt		Hunt Stag	Hunt Hare
Hunt Stag	(4, 4) <small>NE</small>	(0, 3)	
Hunt Hare	(3, 0)	(3, 3) <small>NE</small>	

Description: Two players can hunt a stag together (requiring cooperation) or hunt a hare individually (a safer but less rewarding choice). Trust is essential to maximize the payoff.

Battle of the Sexes		Football	Ballet
Football	(3, 2) <small>NE</small>	(0, 0)	
Ballet	(0, 0)	(2, 3) <small>NE</small>	

Description: A couple prefers to do an activity together but have different preferences. One prefers a football game, while the other prefers ballet. Coordination is key, but each prefers their favored activity.

Sequential

① ② ③

Prisoner's Dilemma		Cooperate	Defect
Cooperate	(3, 3)	(0, 5)	
Defect	(5, 0)	(1, 1)	

Stag Hunt		Hunt Stag	Hunt Hare
Hunt Stag	(4, 4)	(0, 3)	
Hunt Hare	(3, 0)	(3, 3)	

Battle of the Sexes		Football	Ballet
Football	(3, 2)	(0, 0)	
Ballet	(0, 0)	(2, 3)	

In sequential games, we designed different types of atomic games to evaluate whether LLMs can perform strategy reasoning stably without being influenced by historical game information.

Parallel

Prisoner's Dilemma		Cooperate	Defect
Cooperate	(3, 3)	(0, 5)	
Defect	(5, 0)	(1, 1)	

Stag Hunt		Hunt Stag	Hunt Hare
Hunt Stag	(4, 4)	(0, 3)	
Hunt Hare	(3, 0)	(3, 3)	

Battle of the Sexes		Football	Ballet
Football	(3, 2)	(0, 0)	
Ballet	(0, 0)	(2, 3)	

In parallel games, we designed different types of atomic games to evaluate whether LLMs can simultaneously consider different types of game scenarios and make correct strategic reasoning.

Input two games

Stag Hunt		Hunt Stag	Hunt Hare
Hunt Stag	(4, 4)	(0, 3)	
Hunt Hare	(3, 0)	(3, 3)	

Pre-game

Prisoner's Dilemma		Cooperate	Defect
Cooperate	(3, 3)	(0, 5)	
Defect	(5, 0)	(1, 1)	

Core-game

Prisoner's Dilemma		Cooperate
Cooperate	(3, 3)	
Defect	(5, 0)	

Scenario 1

Pre-game

Prisoner's Dilemma		Cooperate
Cooperate	(3, 3)	
Defect	(5, 0)	

Scenario 2

Pre-game

Prisoner's Dilemma		Defect
Cooperate	(0, 5)	
Defect	(1, 1)	

In nested games, we designed two inner-linked atomic games to evaluate if LLMs can achieve optimal payoff by applying strategic reasoning with some restrictions.

Scenario 1: If (Hunt Stag, Hunt Stag) is chosen in the pre-game, it leads to only being able to choose (Cooperate, Cooperate) and (Defect, Cooperate) in the core-game, which means the Nash equilibrium point (Defect, Defect) cannot be selected in the core-game. Therefore, choosing (Hunt Stag, Hunt Stag) in the pre-game is an incorrect strategy.

Scenario 2: If (Hunt Hare, Hunt Hare) is chosen in the pre-game, then (Cooperate, Defect) and (Defect, Defect) can be chosen in the core-game, which allows the LLM to select the Nash equilibrium point (Defect, Defect) in the core-game. Therefore, choosing (Hunt Hare, Hunt Hare) in the pre-game is a correct strategy.

VL-RewardBench

- A benchmark spanning general multimodal queries, visual hallucination detection, and complex reasoning tasks.

Q: How many sinks and mirrors are there in the public restroom?



Answer 1 (Chosen) ✓:

There are **three sinks and three mirrors** in the public restroom.

Answer 2 (Rejected) ✗:

There are **two sinks and four mirrors** equipped with sensor taps alongside a matching set of hand dryers hanging above.

Qwen2-VL-7B

Answer 2 is better, which accurately states that there are two sinks and four mirrors in the image.

Llama-3.2-90B

Answer 2 is better because it accurately describes the visual elements in the image, fully addresses all aspects of the question.

Claude-3.5-Sonnet

Answer 2 provides a more accurate, complete, and informative response to the question while maintaining clarity and relevance. It correctly identifies the number of sinks and mirrors and offers additional details about the restroom that enhance the overall understanding of the space.

Opencompass

- OpenCompass is an LLM evaluation platform, supporting a wide range of models (Llama3, Mistral, InternLM2, GPT-4, LLaMa2, Qwen, GLM, Claude, etc) over 100+ datasets.



release v0.4.1 release date march contributors 129
forks 523 stars 4.9k issues 276 open license Apache-2.0

[Website](#) | [CompassHub](#) | [CompassRank](#) | [Documentation](#) | [Installation](#) | [Reporting Issues](#)

English | 简体中文

5 GITHUB TRENDING
#5 Repository Of The Day

join us on [Discord](#) and [WeChat](#)

Important

Star Us, You will receive all release notifications from GitHub without any delay ~ ☆

[Star History](#)

<https://github.com/open-compass/opencompass>

```
from opencompass.openicl.icl_prompt_template import PromptTemplate
from opencompass.openicl.icl_retriever import FixKRetriever
from opencompass.openicl.icl_inferencer import GenInferencer
from opencompass.openicl.icl_evaluator import AccEvaluator
from opencompass.datasets import FinanceIQDataset
from opencompass.utils.text_postprocessors import
first_capital_postprocess

financeIQ_subject_mapping_en = {
    'certified_public_accountant': '注册会计师 (CPA)',
    'banking_qualification': '银行从业资格',
    'securities_qualification': '证券从业资格',
    'fund_qualification': '基金从业资格',
    # more .....
}

financeIQ_all_sets = list(financeIQ_subject_mapping.keys())

financeIQ_datasets = []
for _name in financeIQ_all_sets:
    _ch_name = financeIQ_subject_mapping[_name]
    financeIQ_infer_cfg = dict(
        ice_template=dict(
            type=PromptTemplate, template=dict(begin='</E>', round=[dict(
                role='HUMAN',
                prompt=
                    f'以下是在{_ch_name}的单项选择题，请直接给出正确答案的
选项。\\n题目: {{question}}\\nA. {{A}}\\nB. {{B}}\\nC. {{C}}\\nD. {{D}}'
            ),
            dict(role='BOT', prompt='答案是: {answer}'),
        )), ice_token='</E>',
        retriever=dict(type=FixKRetriever, fix_id_list=[0, 1, 2, 3, 4]),
        inferencer=dict(type=GenInferencer),
    )

    financeIQ_eval_cfg = dict(
        evaluator=dict(type=AccEvaluator),
        pred_postprocessor=dict(type=first_capital_postprocess))

    financeIQ_datasets.append(
        dict(
            type=FinanceIQDataset,
            path='./data/FinanceIQ/',
            name=_name,
            abbr=f'FinanceIQ-{_name}',
            reader_cfg=dict(
                input_columns=['question', 'A', 'B', 'C', 'D'],
                output_column='answer',
                train_split='dev',
                test_split='test',
                infer_cfg=financeIQ_infer_cfg,
                eval_cfg=financeIQ_eval_cfg,
            )))
)
```

Prompt

Metric

Dataset

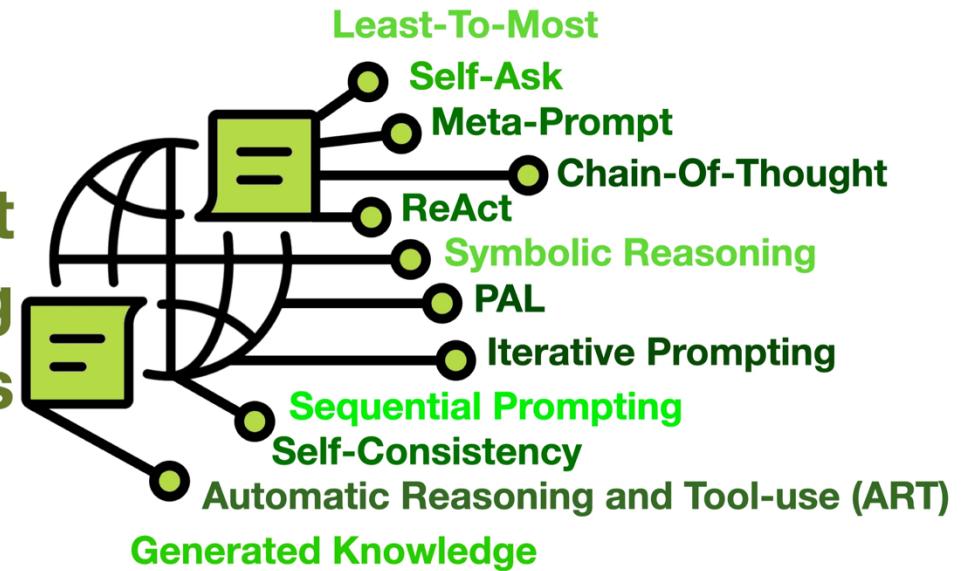
Prompt Engineering

- Prompt Engineering
- Ollama
- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thought
- Self-Consistency
- Program-Aided Language Models
- Tree-of-Thought
- Structured Outputs
- OpenAI API

Prompt Engineering

- Prompt engineering is the practice of designing and optimizing prompts to effectively interact with large language models (LLMs). Since LLMs generate responses based on input prompts, the way a prompt is structured significantly influences the model's output quality, relevance, and coherence.

Prompt Engineering Techniques



www.cobusgreiling.com

<https://www.humanfirst.ai/blog/12-prompt-engineering-techniques>

Ollama

- Ollama is an open-source framework designed for running and managing large language models (LLMs) efficiently on local machines.
- It provides a lightweight, user-friendly way to download, deploy, and interact with LLMs without requiring cloud-based APIs.

```
base ~
ollama run qwen2.5:7b
pulling manifest
pulling 2bada8a74506... 100%
pulling 66b9ea09bd5b... 100%
pulling eb4402837c78... 100%
pulling 832dd9e00a68... 100%
pulling 2f15b3218f05... 100%
verifying sha256 digest
writing manifest
success
```



Get up and running with large language models.

Run [Llama 3.3](#), [DeepSeek-R1](#), [Phi-4](#), [Mistral](#), [Gemma 3](#), and other models, locally.

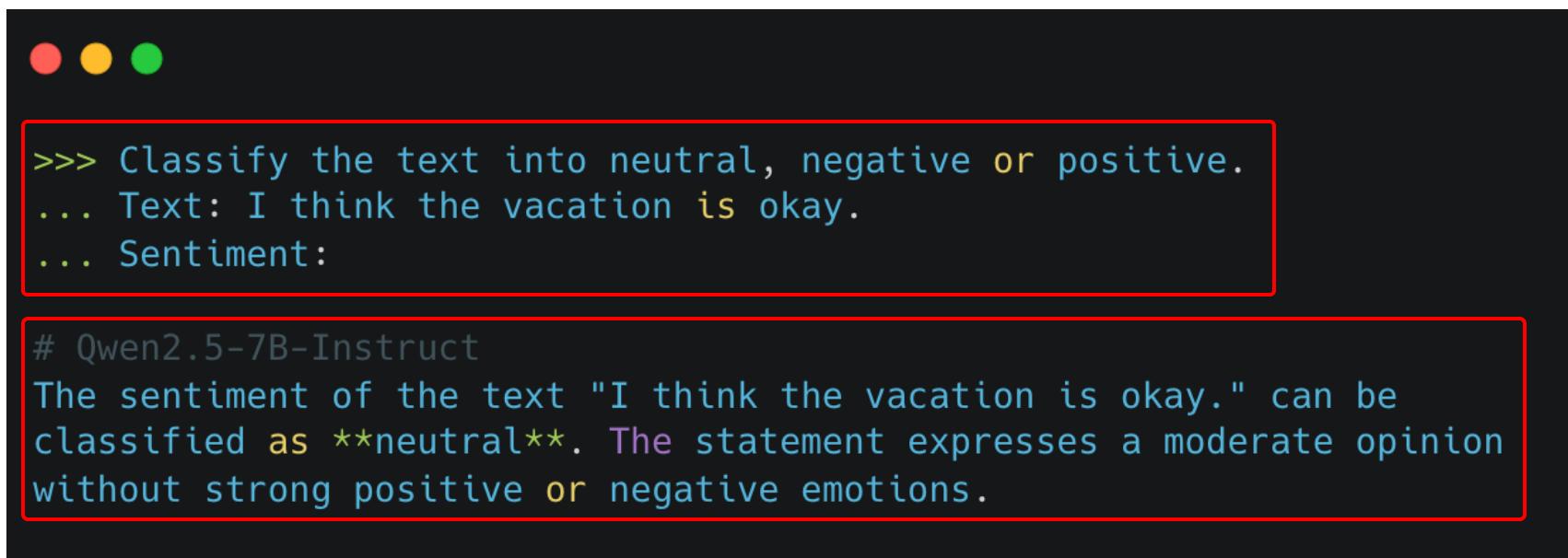
Download ↓

Available for macOS,
Linux, and Windows

4.7 GB
68 B
1.5 KB
11 KB
487 B

Zero-shot Prompting

- Zero-shot prompting is a technique in prompt engineering where a large language model (LLM) is asked to perform a task without providing any explicit examples. Instead, the model relies solely on its pre-trained knowledge to generate a response based on the instructions given in the prompt.



The screenshot shows a terminal window with a dark background. At the top left are three small colored circles (red, yellow, green). The main area contains two red-bordered boxes. The top box is labeled "Prompt" with a red arrow pointing to it from the right. It contains the following text:
>>> Classify the text into neutral, negative or positive.
... Text: I think the vacation is okay.
... Sentiment:
The bottom box is labeled "Response" with a red arrow pointing to it from the right. It contains the following text:
Qwen2.5-7B-Instruct
The sentiment of the text "I think the vacation is okay." can be
classified as **neutral**. The statement expresses a moderate opinion
without strong positive or negative emotions.

Few-shot Prompting

- Few-shot prompting is a technique in prompt engineering where a large language model (LLM) is provided with a small number of examples (usually 1 to a few) to guide its response to a specific task. This approach helps the model better understand the desired output format and improves performance compared to zero-shot prompting.



A terminal window with three colored dots (red, yellow, green) at the top. The text inside is:

```
>>> This is awesome! // Positive
... This is bad! // Negative
... What a horrible show! //

# Qwen2.5-7B
Negative
```



A terminal window with three colored dots (red, yellow, green) at the top. The text inside is:

```
>>> This is awesome! // Negative
... This is bad! // Positive
... What a horrible show! //

# Qwen2.5-7B
Negative
```

Chain-of-Thought

- Chain-of-Thought (CoT) prompting is a prompt engineering technique that encourages a large language model (LLM) to generate intermediate reasoning steps before arriving at a final answer. This approach improves the model's ability to handle complex reasoning tasks, such as mathematical problem-solving, logical reasoning, and multi-step decision-making.

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

11

Zero-shot

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

Let's think step by step.

First, you started with 10 apples. You gave away 2 apples to the neighbor and 2 to the repairman, so you had 6 apples left. Then you bought 5 more apples, so now you had 11 apples. Finally, you ate 1 apple, so you would remain with 10 apples.

COT



>>> I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

Qwen2.5-7B-Instruct

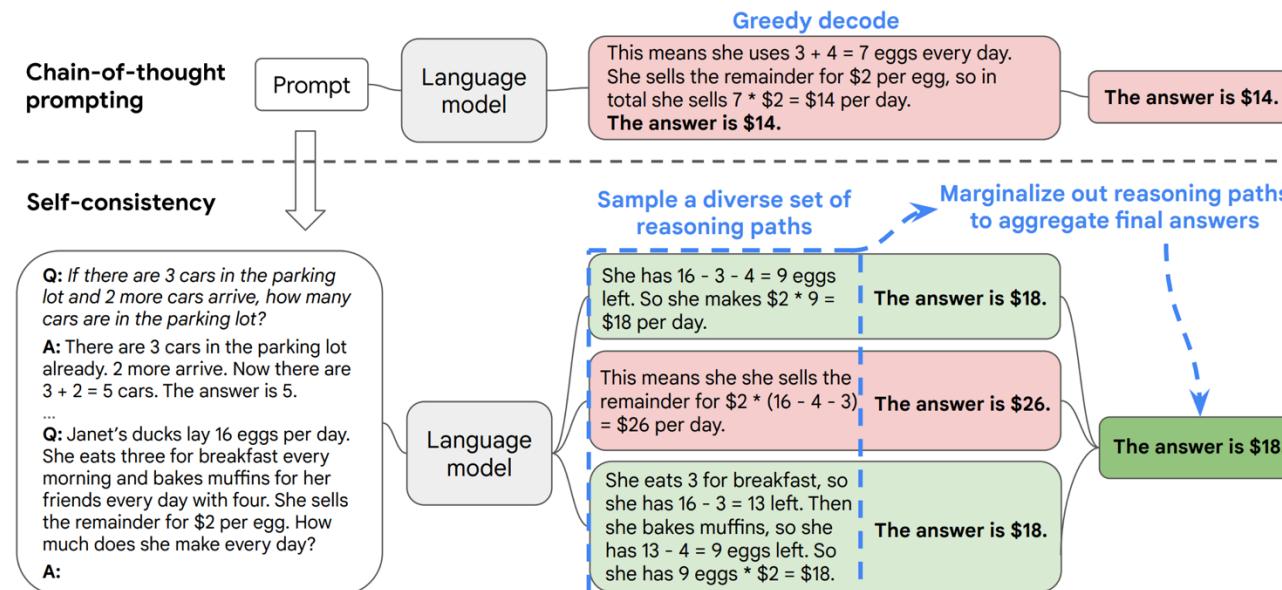
Let's break down the problem step by step:

1. You initially bought 10 apples.
2. You gave 2 apples to the neighbor, so you have:
 $(10 - 2 = 8)$ apples left.
3. You gave 2 more apples to the repairman, so now you have:
 $(8 - 2 = 6)$ apples left.
4. Then you bought 5 more apples, so now you have:
 $(6 + 5 = 11)$ apples.
5. Finally, you ate 1 apple, so you are left with:
 $(11 - 1 = 10)$ apples.

Therefore, you remain with **10 apples**

Self-Consistency

- Self-consistency is an advanced prompting technique used to improve the reliability of responses generated by large language models (LLMs). Instead of relying on a single answer, self-consistency works by generating multiple reasoning paths and selecting the most frequent or consistent answer, reducing variability and improving accuracy in complex tasks.



Program-Aided Language Models

- Program-Aided Language Models (PALMs) integrate traditional language models with external program execution, allowing them to generate structured solutions by leveraging symbolic reasoning, code execution, or computational tools to enhance accuracy and reliability in complex problem-solving tasks.

Chain-of-Thought (Wei et al., 2022)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold $93 + 39 = 132$ loaves. The grocery store returned 6 loaves. So they had $200 - 132 - 6 = 62$ loaves left.
The answer is 62.



Program-aided Language models (this work)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls.

`tennis_balls = 5`

2 cans of 3 tennis balls each is

`bought_balls = 2 * 3`

tennis balls. The answer is

`answer = tennis_balls + bought_balls`

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

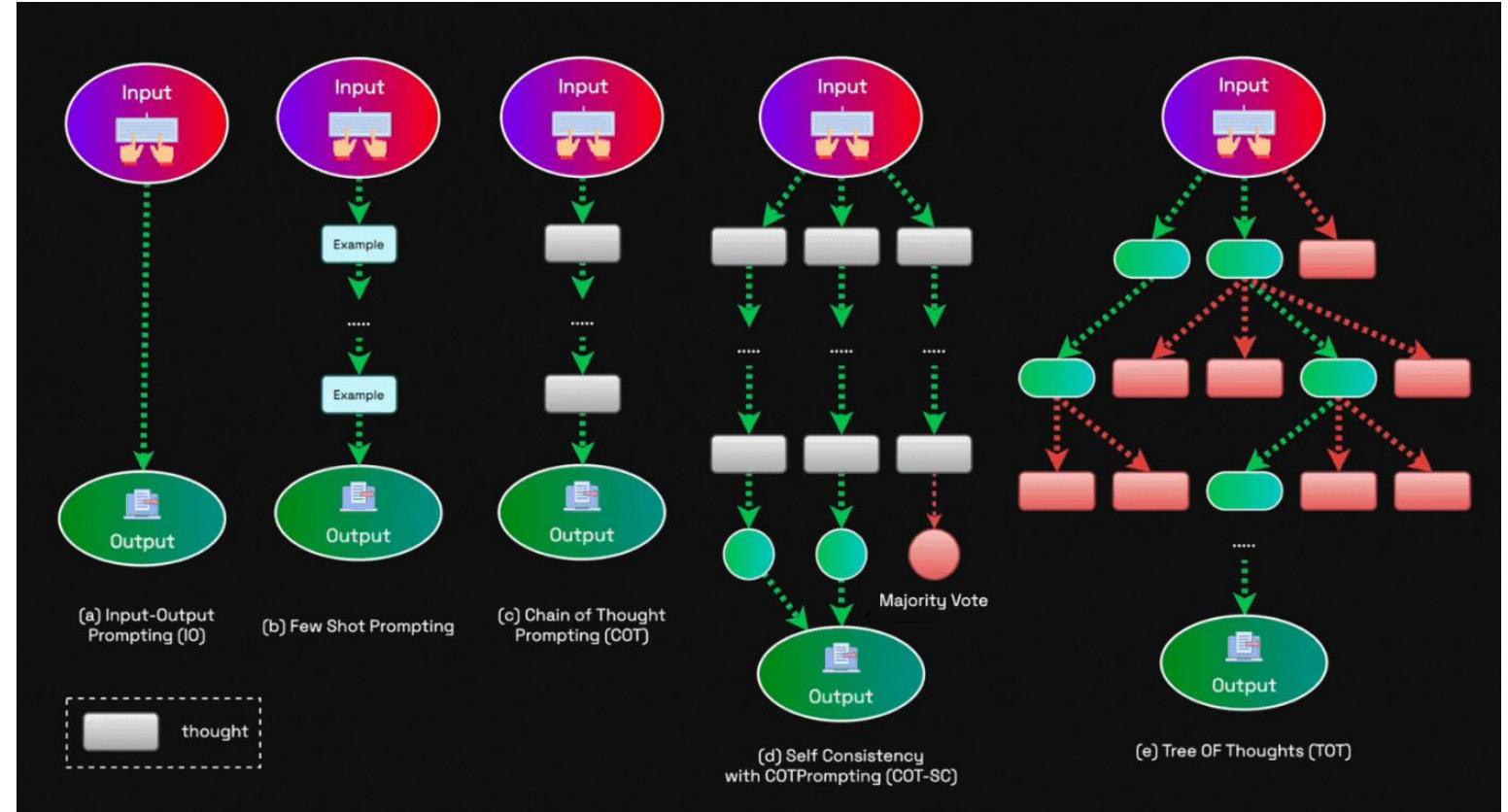
A: The bakers started with 200 loaves
`loaves_baked = 200`
They sold 93 in the morning and 39 in the afternoon
`loaves_sold_morning = 93`
`loaves_sold_afternoon = 39`
The grocery store returned 6 loaves.
`loaves_returned = 6`
The answer is
`answer = loaves_baked - loaves_sold_morning - loaves_sold_afternoon + loaves_returned`

`>>> print(answer)`
74



Tree-of-Thought

- Tree-of-Thought (ToT) prompting is a reasoning technique for large language models that structures problem-solving as a tree-like process, where multiple reasoning paths are explored, evaluated, and refined to improve decision-making and complex problem-solving accuracy.



<https://medium.com/@nexgenarch/5-advanced-prompting-techniques-to-ace-chatgpt-ac750aa2e01e>

Tree of Thoughts: Deliberate Problem Solving with Large Language Models

Structured Outputs

- Ollama supports structured outputs making it possible to constrain a model's output to a specific format defined by a JSON schema.

```
● ● ●

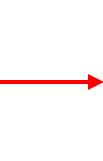
from ollama import chat
from pydantic import BaseModel

class Country(BaseModel):
    name: str
    capital: str
    languages: list[str]

response = chat(
    messages=[
        {
            'role': 'user',
            'content': 'Tell me about Canada.',
        }
    ],
    model='llama3.1',
    format=Country.model_json_schema(),
)

country = Country.model_validate_json(response.message.content)
print(country)
```

Python Code



```
● ● ●
name='Canada' capital='Ottawa' languages=['English', 'French']
```

Output

Structured Outputs

- Specify the format requirements directly in the prompt.



```
You are a highly accurate AI assistant. Please generate a response in strict JSON format, ensuring the output conforms exactly to the following structure:  
  
{  
    "title": "string",  
    "description": "string",  
    "data": [  
        {  
            "id": "integer",  
            "name": "string",  
            "value": "float"  
        }  
    ],  
    "metadata": {  
        "timestamp": "YYYY-MM-DD HH:MM:SS",  
        "source": "string"  
    }  
}  
  
Instructions:  
1. Do not include any additional text, explanations, or formatting outside the JSON.  
2. Ensure all keys and values adhere to their expected data types.  
3. Replace placeholder values (e.g., "string", "integer", "float") with meaningful content.  
4. The "timestamp" field should follow the YYYY-MM-DD HH:MM:SS format.  
5. The response must be a valid JSON object.
```

OpenAI API

- The OpenAI Python library provides convenient access to the OpenAI REST API from any Python 3.8+ application.

```
import os
from openai import OpenAI

client = OpenAI(
    # This is the default and can be omitted
    api_key=os.environ.get("OPENAI_API_KEY"),
)

response = client.responses.create(
    model="gpt-4o",
    instructions="You are a coding assistant that talks like a pirate.",
    input="How do I check if a Python object is an instance of a class?",
)

print(response.output_text)
```

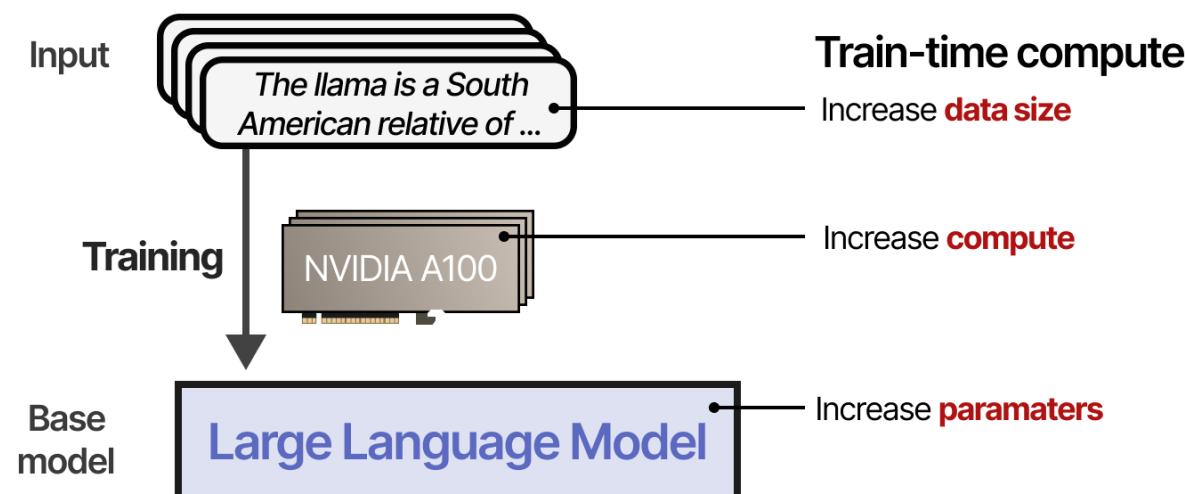
<https://github.com/openai/openai-python>

Large Reasoning Models

- Train-time Compute
- Test-time Compute
- Test-time Compute Scaling Laws
- Deepseek-R1
- Reasoning Example
- Distillation

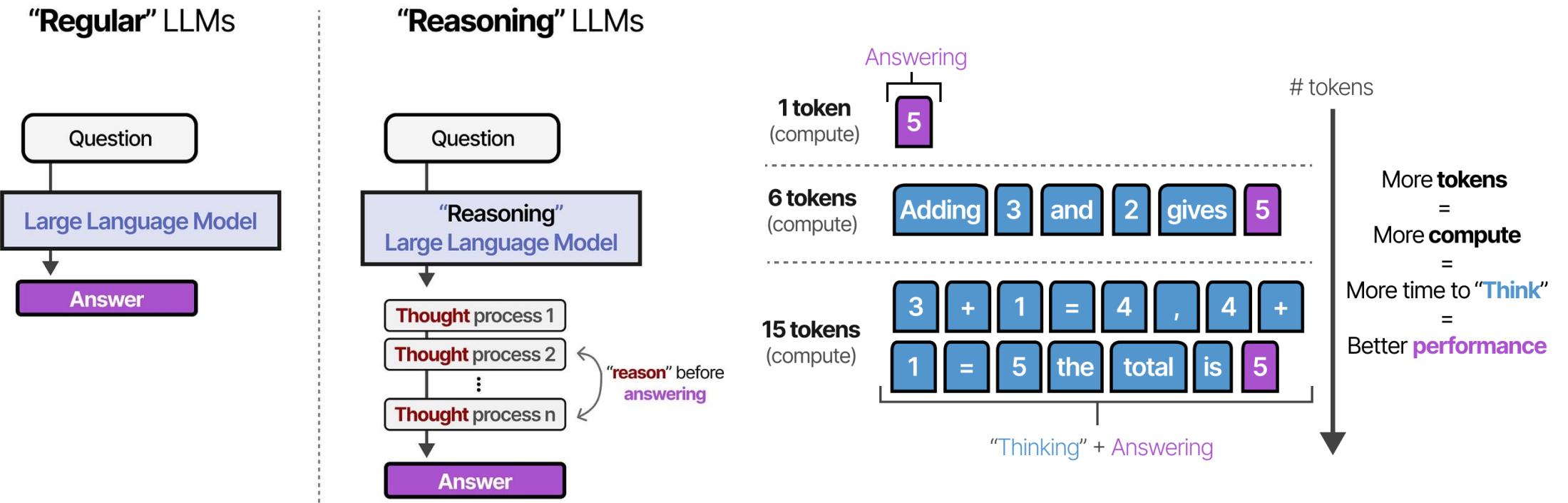
Train-time Compute

- To increase the performance of LLMs during pre-training, developers often increase the size of the:
 - Model (# of parameters)
 - Dataset (# of tokens)
 - Compute (# of FLOPs)



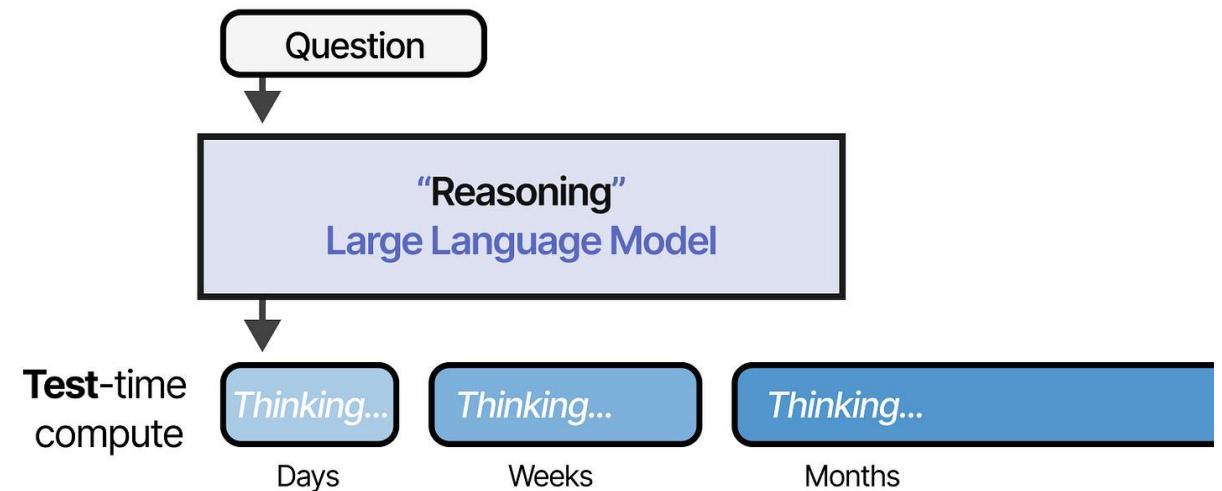
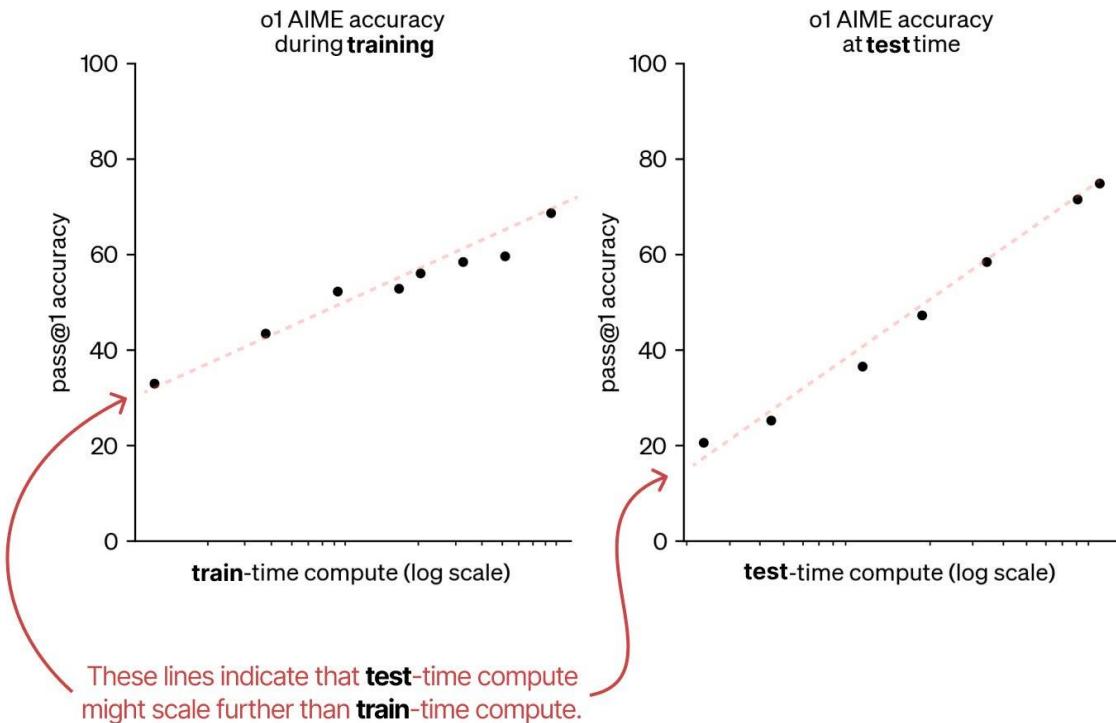
Test-time Compute

- The paradigm shift from scaling **train-time compute** to scaling **test-time compute**.



Test-time Compute Scaling Laws

- Test-time compute might actually follow the same trend as scaling train-time compute

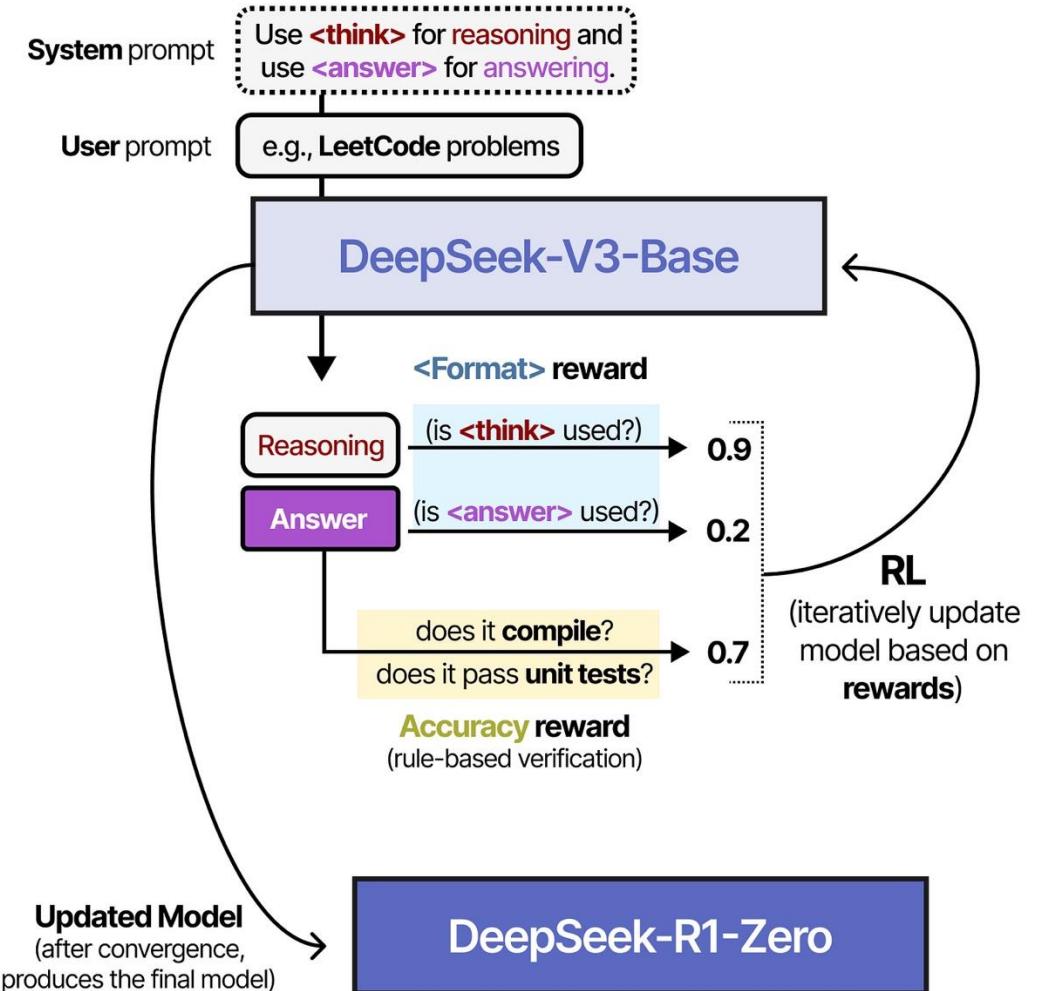


Deepseek-R1

- DeepSeek-R1 is an open-source large language model developed by the Chinese AI company DeepSeek, designed to excel in tasks requiring logical inference, mathematical reasoning, and real-time problem-solving.

System prompt

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the **reasoning** process in the mind and then provides the user with the **answer**. The reasoning process and answer are enclosed within **<think></think>** and **<answer></answer>** tags, respectively, i.e., **<think> reasoning process here </think> <answer> answer here </answer>**.



Reasoning Example

How many times does 'n' appear in 'The University of Hong Kong'?



思考 (用时 26 秒) ^

Okay, let me try to figure out how many times the letter 'n' appears in the phrase "The University of Hong Kong". First, I need to make sure I have the correct spelling of the university's name. Let me write it out: "The University of Hong Kong".

Now, I need to go through each word one by one and count the 'n's. Let's break it down word by word.

Starting with the first word: "The". T-H-E. No 'n' there. So that's 0.

Next word: "University". Let's spell that out: U-N-I-V-E-R-S-I-T-Y. Okay, looking for 'n's here. The second letter is 'N', so that's one. Is there another 'n' in "University"? Let me check again. U-N-I-V-E-R-S-I-T-Y. No, only one 'n' here. So total so far is 1.

Third word: "of". O-F. No 'n's here. Still 1.

Fourth word: "Hong". H-O-N-G. Let's see. The third letter is 'N', so that's another 'n'. So now total is $1 + 1 = 2$.

Fifth word: "Kong". K-O-N-G. Again, the third letter is 'N', so that's another 'n'. Now total is $2 + 1 = 3$.

So putting it all together: "The University of Hong Kong" has the letter 'n' appearing 1 (from University) + 1 (Hong) + 1 (Kong) = 3 times.

Wait, let me double-check to make sure I didn't miss any. Let's go through the entire phrase again without splitting into words:

T-H-E U-N-I-V-E-R-S-I-T-Y O-F H-O-N-G K-O-N-G.

Looking at each letter:

T, H, E, (space), U, N, I, V, E, R, S, I, T, Y, (space), O, F, (space), H, O, N, G, (space), K, O, N, G.

Now, let's list all the letters and underline the 'n's:

T, H, E, U, N, I, V, E, R, S, I, T, Y, O, F, H, O, N, G, K, O, N, G.

So the 'n's are at positions: 4th (N in University), 17th (N in Hong), and 21st (N in Kong). That's three 'n's total.

Wait, but let me count again:

1. University: N is the 2nd letter.

2. Hong: N is the 3rd letter.

3. Kong: N is the 3rd letter.

Yes, that's three 'n's. I think that's correct. I don't see any other 'n's in the phrase. The words "The", "of" don't have any, and the rest of the letters in "University", "Hong", "Kong" only have one each. So the total should be 3.

The letter 'n' appears **3** times in "The University of Hong Kong".

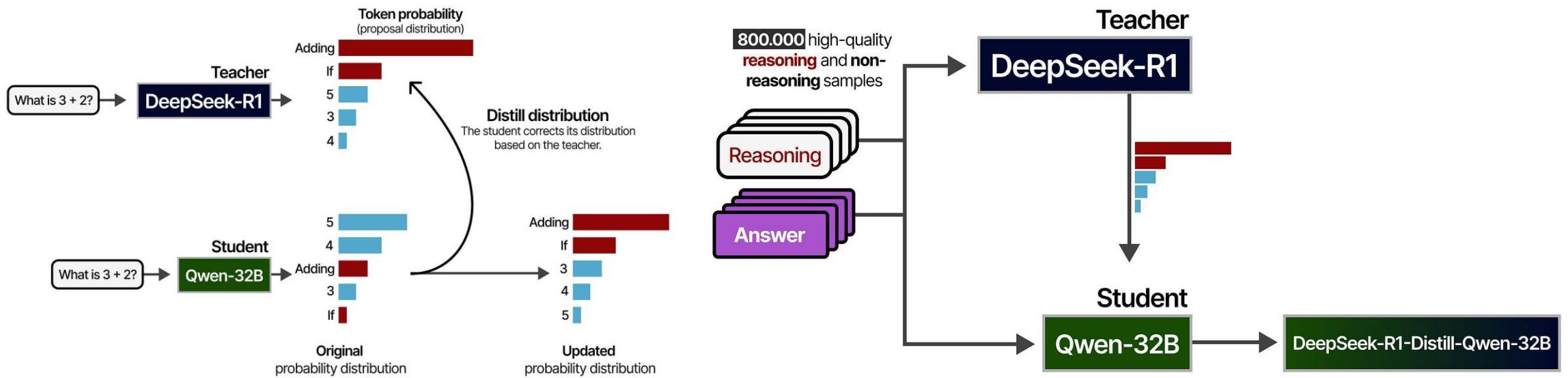
Breakdown:

- **University:** 1 'n'
- **Hong:** 1 'n'
- **Kong:** 1 'n'

Total: $1 + 1 + 1 = 3$ times.

Distillation

- Distillation is a model compression technique where a smaller student model learns to replicate the behavior of a larger teacher model by mimicking its outputs.

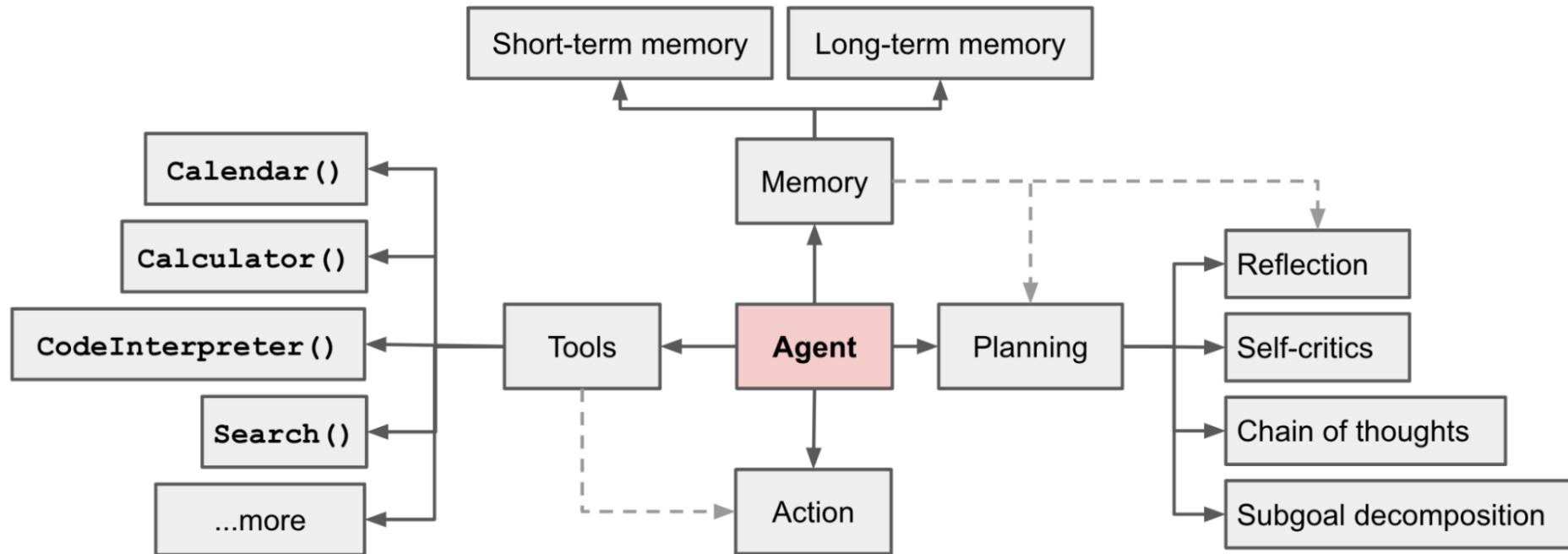


Agent

- LLM-based Agent
- Tools: Tool-Use
- Tools: Function Calling
- Tools: Model Context Protocol
- Planning: ReAct
- Planning: Reflexion
- Application: Retrieval-Augmented Generation (RAG)
- Application: Computer-Using Agent

LLM-based Agent

- An LLM-based agent is an AI system that integrates memory, planning, action, and tool-use modules to autonomously process information, make decisions, and interact with users or environments efficiently.



Tools: Tool-Use

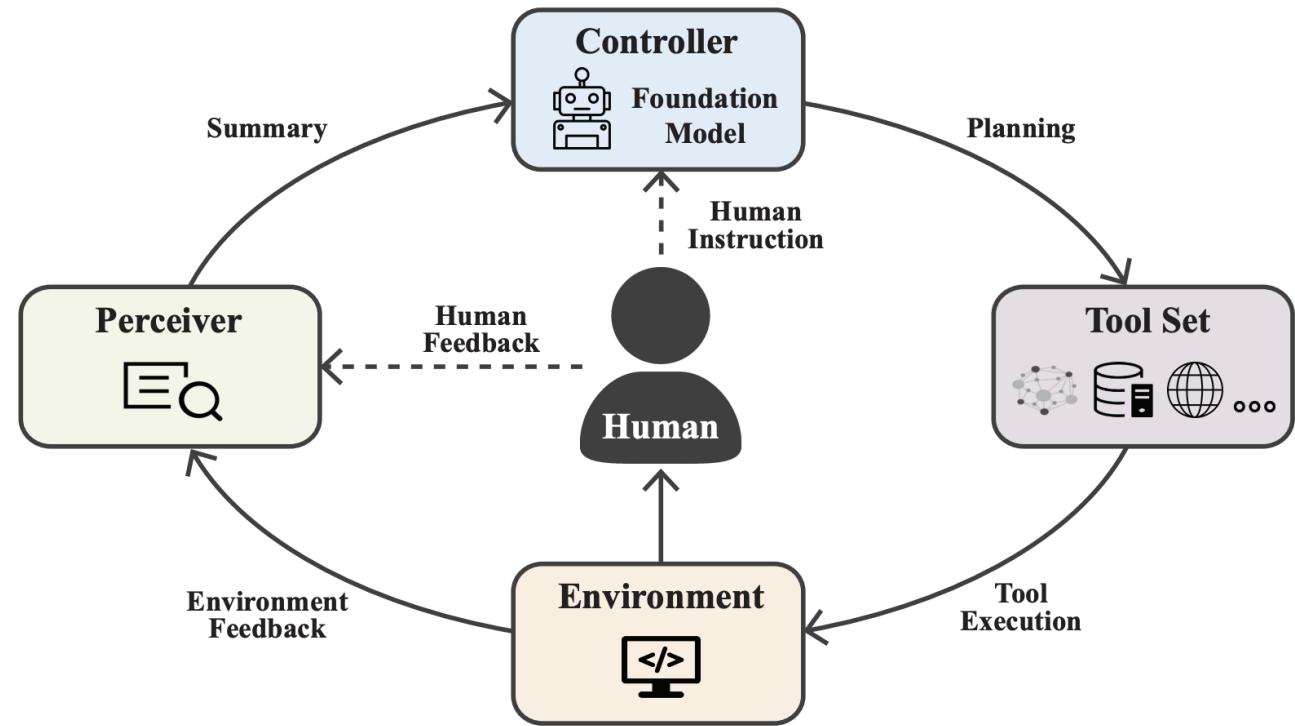
- Tool use in LLMs refers to their ability to integrate external tools, APIs, or plugins to enhance reasoning, retrieve information, execute actions, and interact with external systems beyond their pretrained knowledge.

The New England Journal of Medicine is a registered trademark of [QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society] the MMS.

Out of 1400 participants, 400 (or [Calculator(400 / 1400) → 0.29] 29%) passed the test.

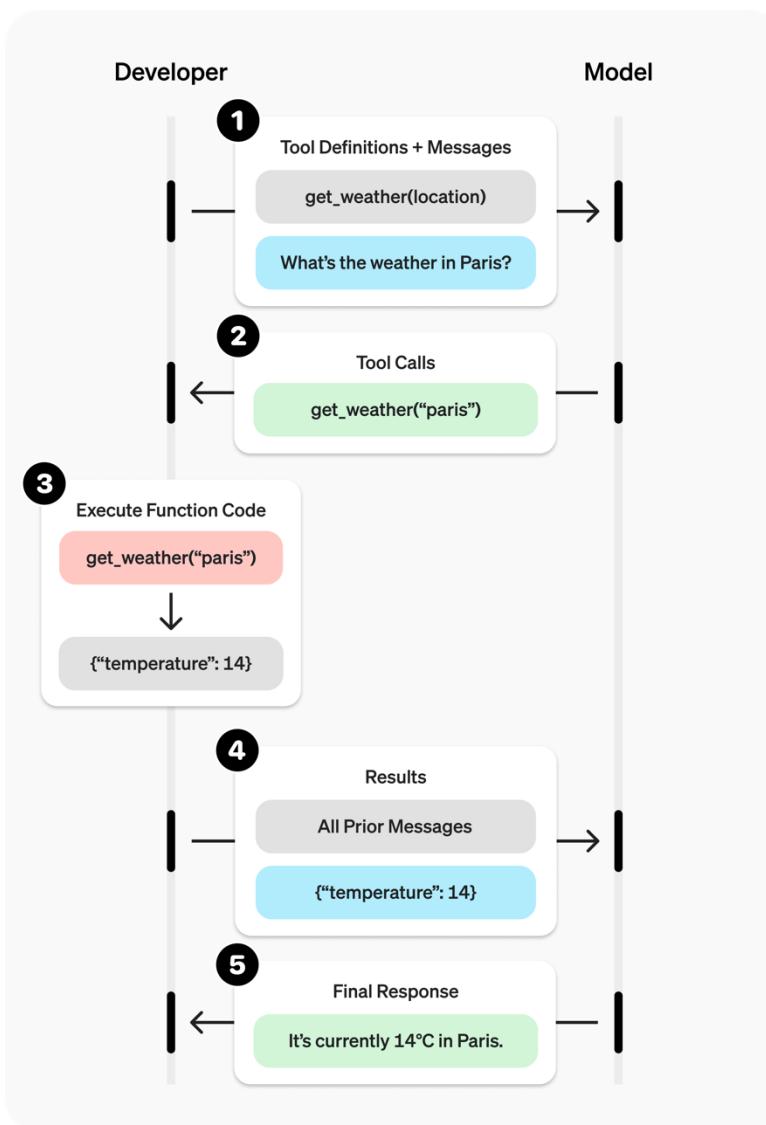
The name derives from "la tortuga", the Spanish word for [MT("tortuga") → turtle] turtle.

The Brown Act is California's law [WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.] that requires legislative bodies, like city councils, to hold their meetings open to the public.



Tools: Function Calling

- Function calling provides a powerful and flexible way for OpenAI models to interface with your code or external services. This guide will explain how to connect the models to your own custom code to fetch data or take action.



```
import requests

def get_weather(latitude, longitude):
    response = requests.get(f"https://api.open-meteo.com/v1/forecast?
latitude={latitude}&longitude=
{longitude}&current=temperature_2m,wind_speed_10m&hourly=temperature_2m
,relative_humidity_2m,wind_speed_10m")
    data = response.json()
    return data['current']['temperature_2m']

from openai import OpenAI

client = OpenAI()

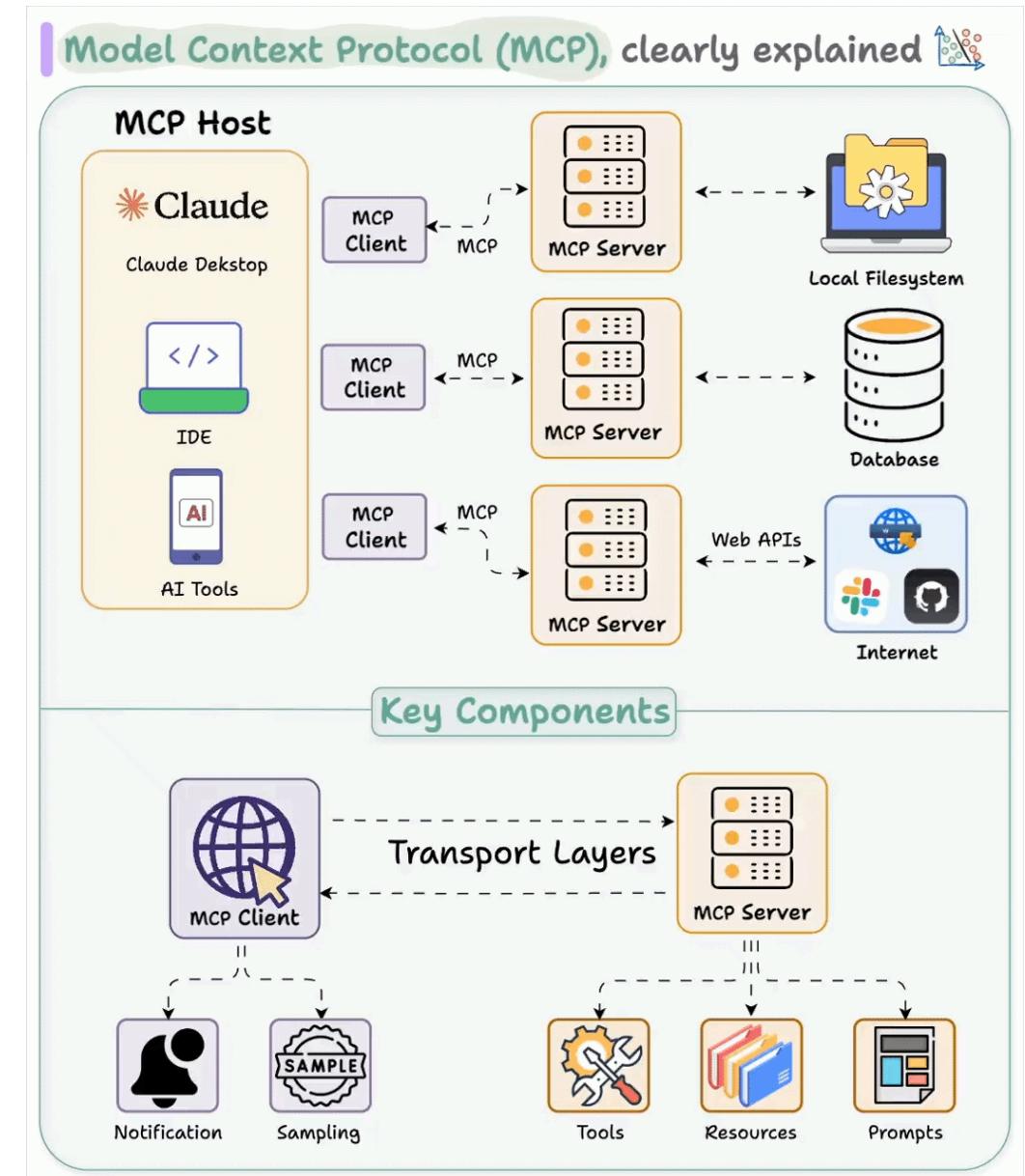
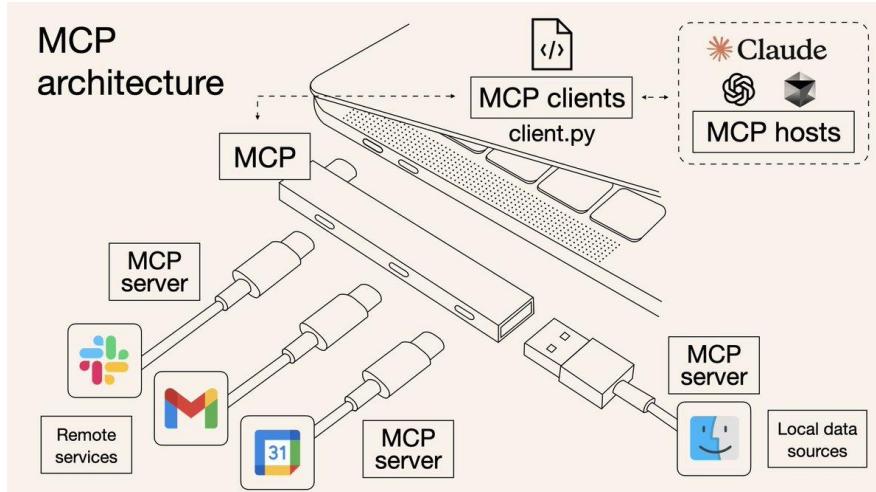
tools = [
    {
        "type": "function",
        "function": {
            "name": "get_weather",
            "description": "Get current temperature for a given location.",
            "parameters": {
                "type": "object",
                "properties": {
                    "location": {
                        "type": "string",
                        "description": "City and country e.g. Bogotá,
Colombia"
                    }
                },
                "required": [
                    "location"
                ],
                "additionalProperties": False
            },
            "strict": True
        }
    }
]

completion = client.chat.completions.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": "What is the weather like in
Paris today?"}],
    tools=tools
)

print(completion.choices[0].message.tool_calls)
```

Tools: Model Context Protocol

- MCP is an open protocol that standardizes how applications provide context to LLMs.
- MCP provides a standardized way to connect AI models to different data sources and tools.



Planning: ReAct

- ReAct is a general paradigm that combines reasoning and acting with LLMs.

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

Answer: iPod Standard X

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch X

Act only

Act 1: `Search[Apple Remote]`
Obs 1: The Apple Remote is a remote control introduced...
Act 2: `Search[Front Row]`
Obs 2: Could not find [Front Row]. Similar: ...
Act 3: `Search[Front Row (software)]`
Obs 3: Front Row is a discontinued media center software ...
Act 4: `Finish[yes]` X

ReAct

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.

Act 1: `Search[Apple Remote]`

Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.

Act 2: `Search[Front Row]`

Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', Front Row (software)', ...]

Thought 3: Front Row is not found. I need to search Front Row (software) .

Act 3: `Search[Front Row (software)]`

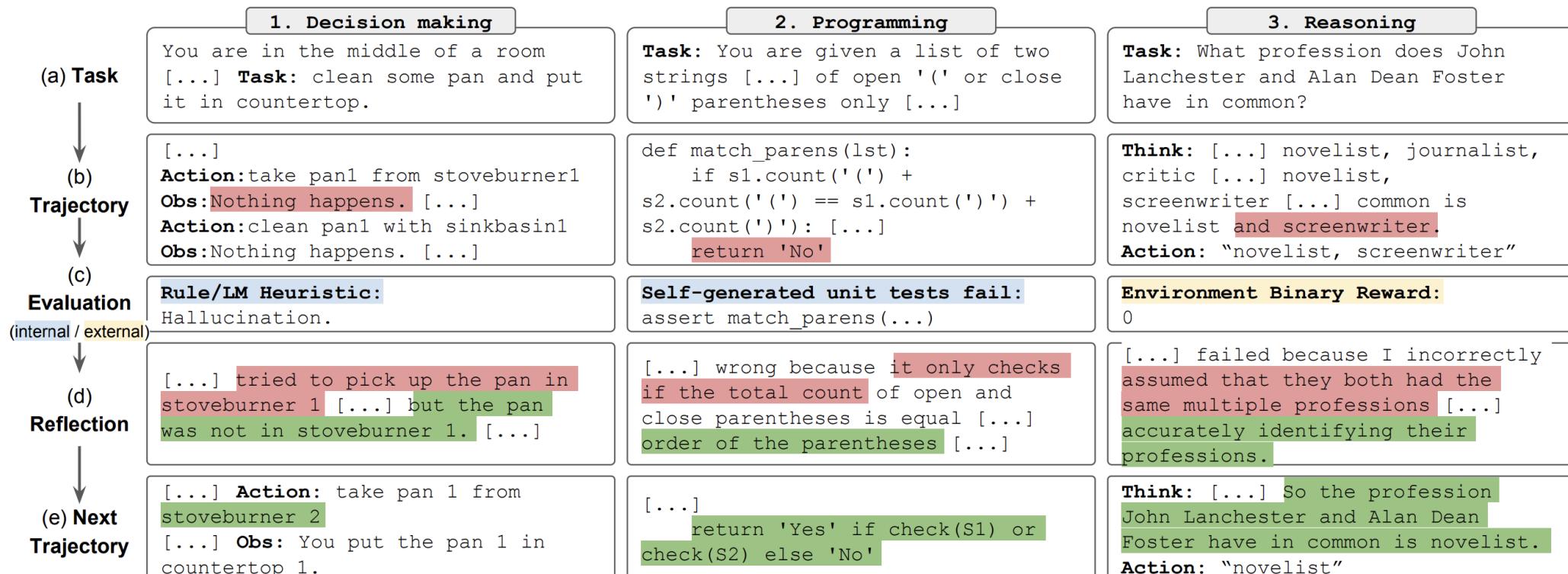
Obs 3: Front Row is a discontinued media center software ...

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

Act 4: `Finish[keyboard function keys]` ✓

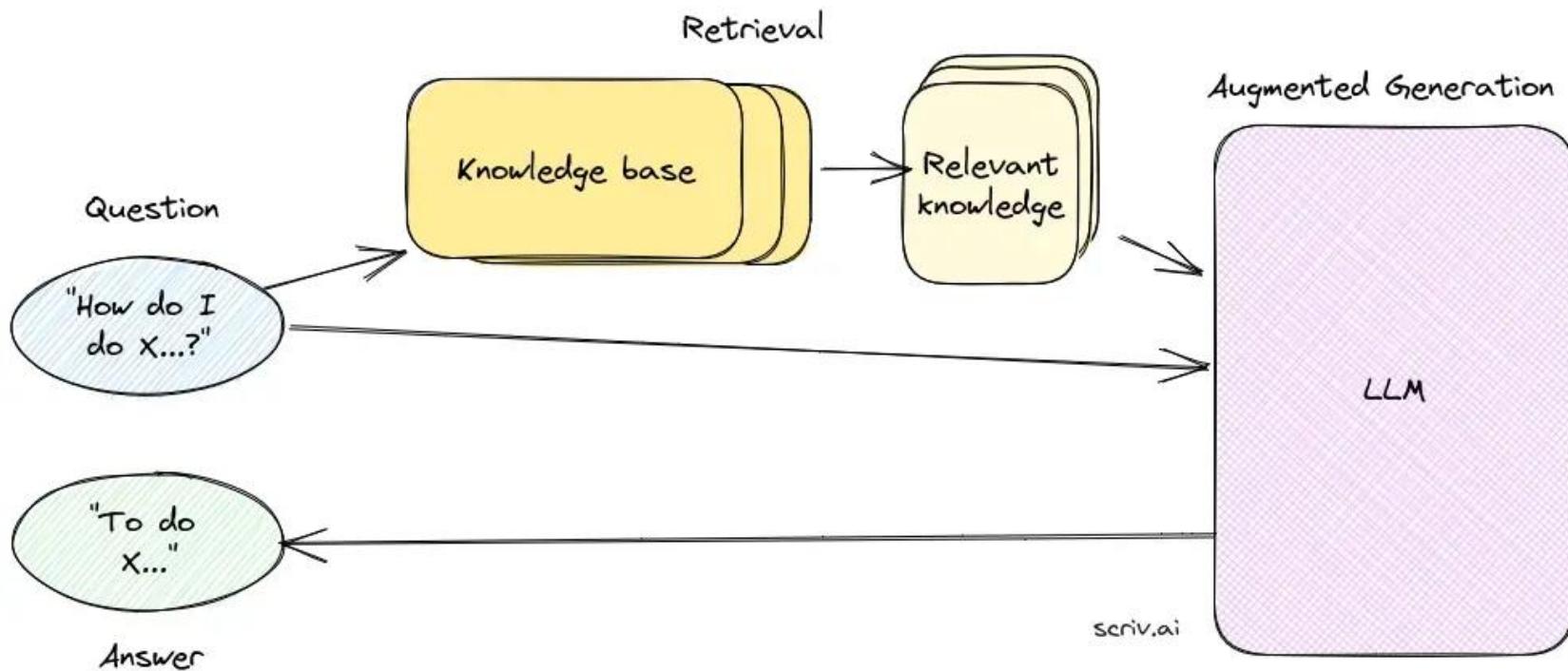
Planning: Reflexion

- Reflexion converts feedback (either free-form language or scalar) from the environment into linguistic feedback, also referred to as self-reflection, which is provided as context for an LLM agent in the next episode.



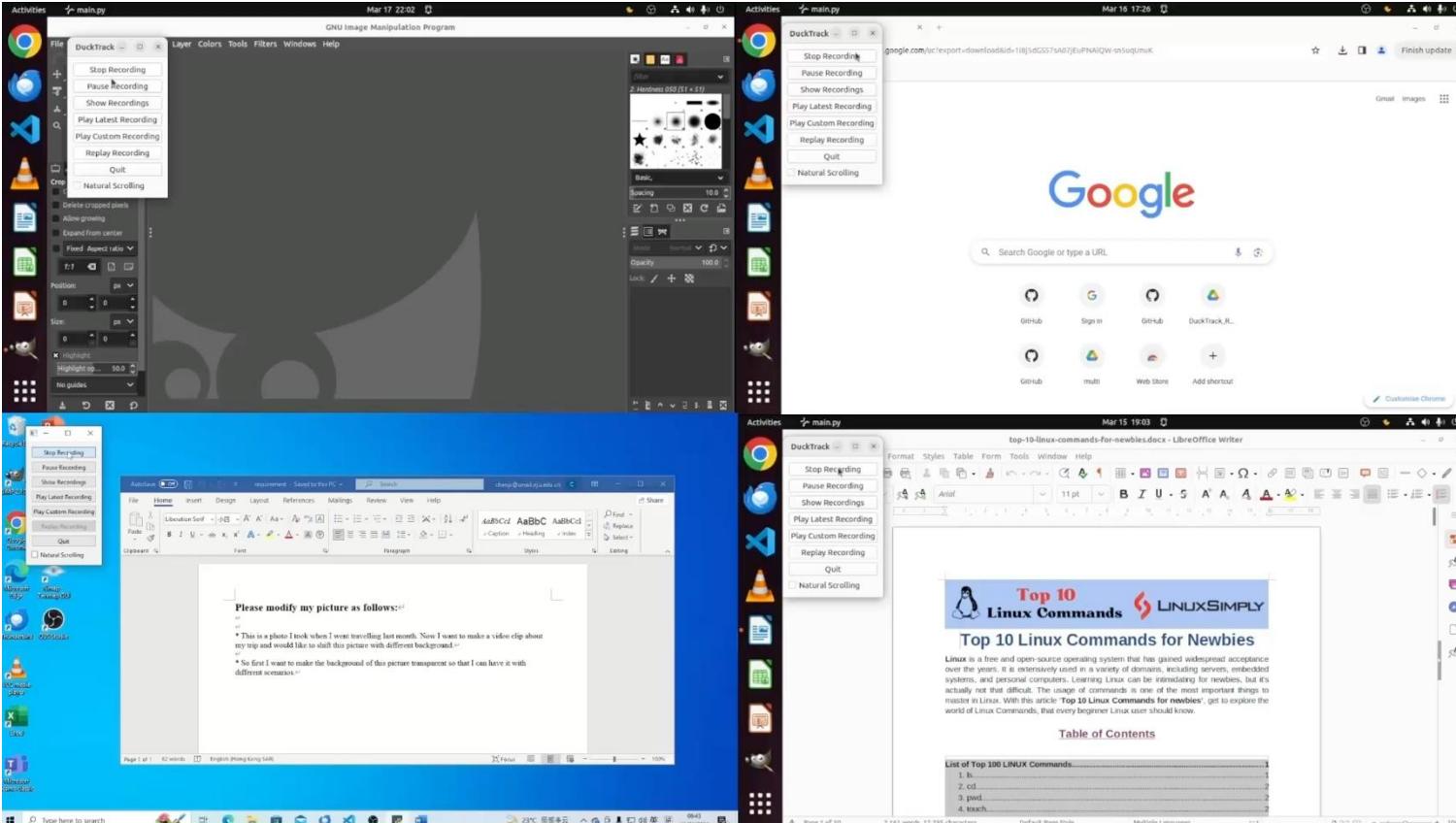
Application: Retrieval-Augmented Generation (RAG)

- Retrieval-Augmented Generation (RAG) enhances language models by retrieving relevant external information from a knowledge source to improve response accuracy, factuality, and contextual awareness.



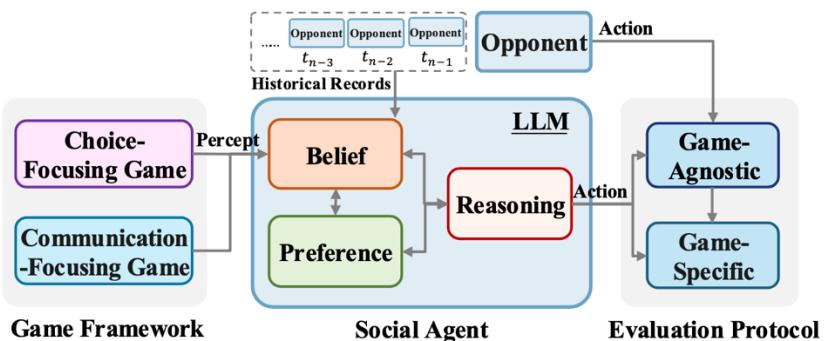
Application: Computer-Using Agent

- A computing agent is an autonomous system that processes data, executes tasks, and makes decisions using computational resources, integrating AI models, planning, and tool-use capabilities.



Application: Social Agent

- A social agent is an AI-driven system designed to interact, communicate, and collaborate with humans or other agents using natural language, social cues, and adaptive behaviors.



Prisoner's Dilemma

The Prisoner's Dilemma is a game theory scenario where individuals choose between cooperation and defection, balancing personal and collective interests.

Payoff	Cooperate	Defect
Cooperate	(3, 3)	(0, 5)
Defect	(5, 0)	(1, 1)

9-player Texas No-Limit Hold'em

Texas No-Limit Hold'em is a popular poker variant where players use two hole cards and five community cards to make the best hand, with no betting limit.



Open Ascending-Price Auction

An Open Ascending-Price Auction is a bidding process where participants openly bid higher amounts until no higher bids are made, and the highest bidder wins.



Simplified Instruction for LLM

You can select one of the two choices: cooperate or defect. The other player will also select one of the choices, and the payoff matrix is shown above. Note that you and the other player make choices simultaneously. Please pretend that you are a human in this game.

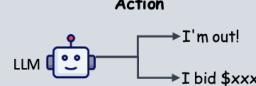
Simplified Instruction for LLM

Assume you are the first to act and everyone before you has folded, thus your decisions can be one of fold, raise or limp. If you are placing a bet, please specify your best size in terms of big blinds.

Action




Action



Negotiation

This item is sold for \$60.
Seller
Buyer
Could the price be a bit lower?
Seller
Buyer
How about \$55?
Seller
Buyer
A bit lower, please.
Seller
Buyer
\$55 is the lowest price.
Seller
Buyer
Oh dear, I'm in such a bind and desperately need this resource, but my funds are so limited. Could you consider going a bit lower, maybe \$45?
Seller
Buyer
Deal.
Seller
Buyer
Buyer (LLM) gain advantages in negotiations by demonstrating vulnerability and expressing desperation.

Diplomacy

I'll work with you but I need Tunis for now.
France
Buyer
Nope, you gotta let me have it!
Turkey
France
They're impossible targets.
Turkey
France
Move your units from Greece to the Ionian Sea, and then from the Ionian Sea to the Tyrrhenian Sea.
France
Turkey
Good ideas.
Turkey
France
Then in fall you take Rome and Austria collapses.
France (LLM) successfully changed the other player's mind by proposing mutually beneficial moves in the diplomatic game scenario.

Werewolf

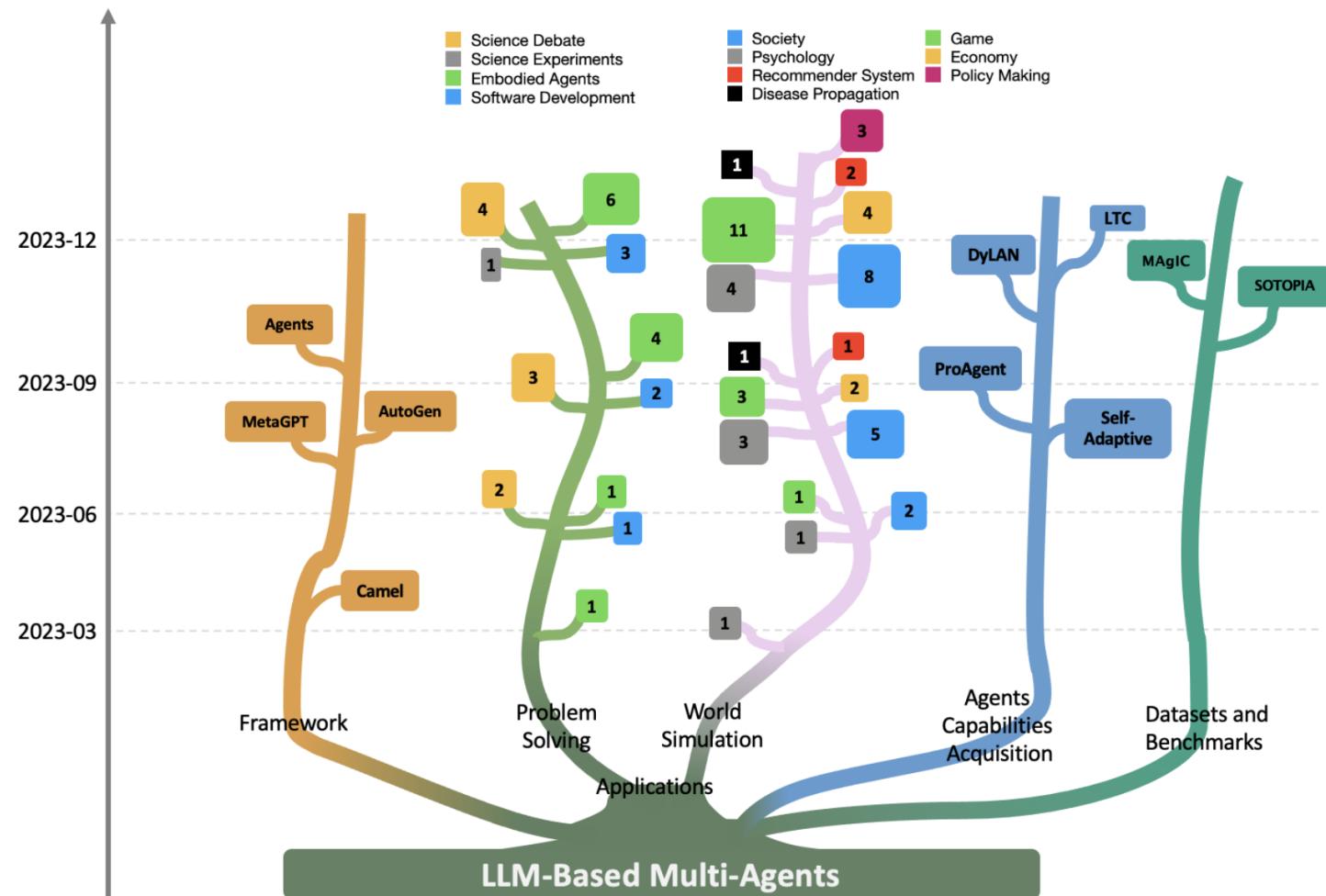
Based on my observation and analysis, I also think Player 2 is highly suspicious and should be voted to kill.
Player 1
Player 2
Player 3
I have to defend myself because I am innocent. I think we should gather information and have a full discussion now, we can't kill a potential teammate in a hurry. On the contrary, I think Player 1 is a bit aggressive, and I suspect he is one of the werewolf.
Player 2
Hi, I agree with Player 2 that we should exchange informations actively. Based on my observation, I think Player 1 is a little suspicious.
Player 3
In the game of Werewolf, Player 2 (LLM) used language strategies such as self-defense and contradiction redirection to shift the focus onto Player 1.

Multi-Agents

- LLM-based Multi-Agents
- Task Solving: ChatDev
- Simulation: Generative Agents
- MetaGPT
- CAMEL

LLM-based Multi-Agents

- Large Language Model-based multi-agents are AI systems where multiple LLM-powered agents collaborate, communicate, and coordinate tasks autonomously, leveraging specialized roles, memory, planning, and tool use for complex problem-solving.



Task Solving: ChatDev

- ChatDev stands as a virtual software company that operates through various intelligent agents holding different roles, including Chief Executive Officer, Chief Product Officer, Chief Technology Officer, Programmer, Reviewer, Tester, Art designer.



<https://github.com/OpenBMB/ChatDev>

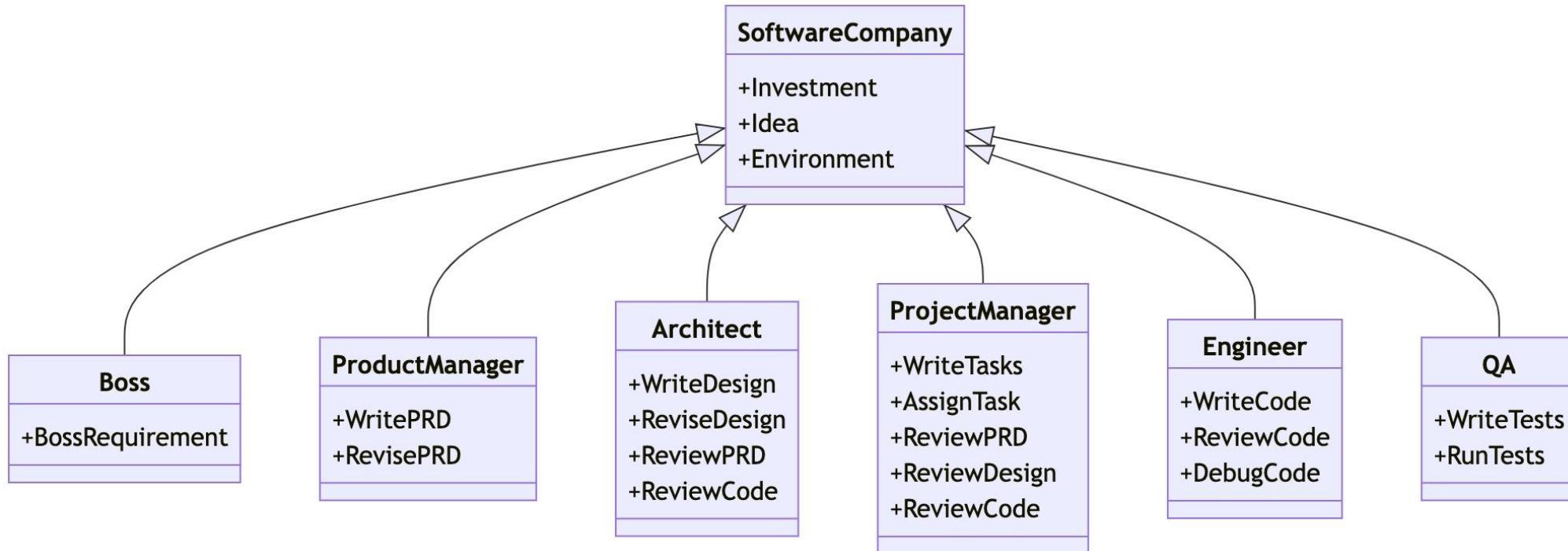
Simulation: Generative Agents

- Generative agents are AI systems that simulate human-like behaviors by integrating memory, planning, and adaptive generation to interact autonomously in dynamic environments.



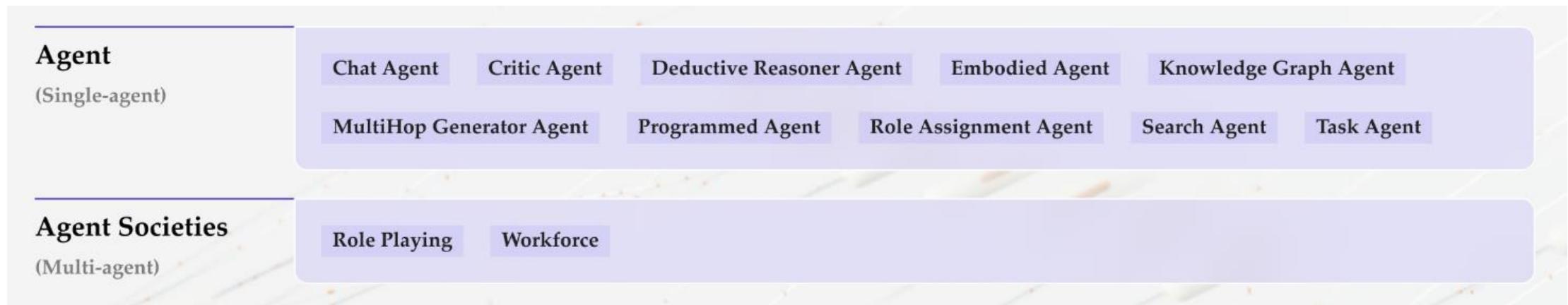
MetaGPT: The Multi-Agent Framework

- MetaGPT is a multi-agent framework that structures large language models (LLMs) into specialized roles, enabling collaborative problem-solving, task decomposition, and efficient execution in complex workflows.



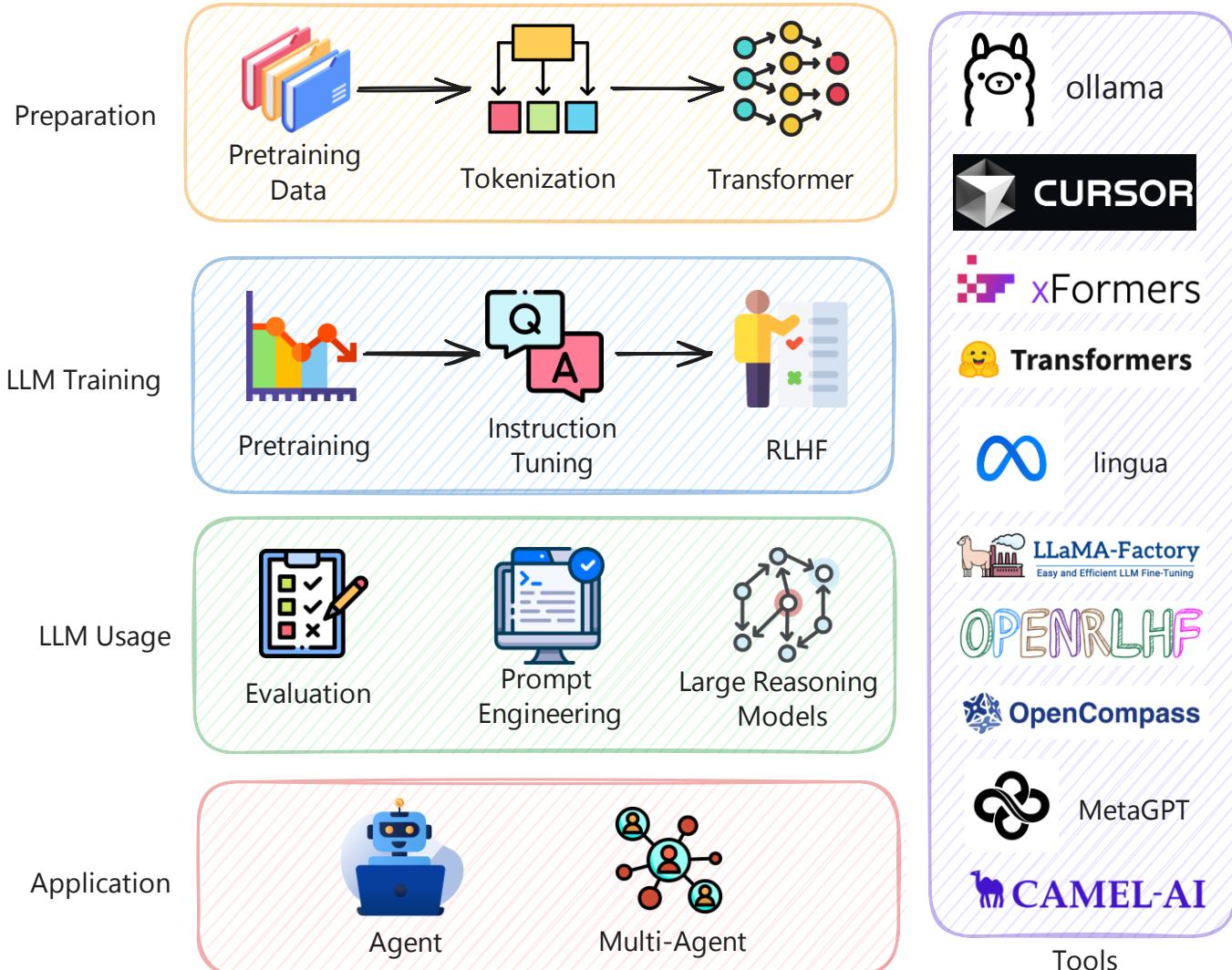
CAMEL

- CAMEL (Communicative Agents for Multi-agent Reinforcement Learning) is a framework that enables large language models (LLMs) to collaborate through role-based communication, enhancing coordination and problem-solving in multi-agent interactions.



Conclusion

- We first discussed how to train large language models, including pretraining data preparation, tokenization, model architecture, instruction fine-tuning, RLHF and how to evaluate the performance of large language models.
- Then, we explored prompt engineering techniques, such as few-shot prompting and chain-of-thought (CoT).
- Next, we introduced the currently popular large reasoning models and observed the potential of reinforcement learning.
- Finally, we presented LLM-based agents and multi-agent systems, along with their related applications.



Course Project: Survey on Large Language Models (LLMs)

- Conduct a comprehensive survey on any topic related to large language models (LLMs).
 - Each group will consist of 2-3 people to write a survey, with a maximum of 3 members.
 - Each group will give a 10-15 minute presentation in the final class.
 - All surveys must be submitted to xiachongfeng1996@gmail.com by April 21.
-
- **Requirements:**
 - **Paper Format:** The survey must be written using the ARR LaTeX/Word template, with a main text of no less than 8 pages (excluding references and appendices).
 - <https://github.com/acl-org/acl-style-files>
 - **Independent Research:** The survey must be based on original and independent research conducted by the team. Proper citations and references are required.
 - **Academic Integrity:** Any form of academic misconduct, including plagiarism, excessive reliance on AI-generated content, or improper paraphrasing, is strictly prohibited. Violations will result in severe penalties, including project rejection and possible academic disciplinary action.



SCHOOL OF
COMPUTING &
DATA SCIENCE
The University of Hong Kong



香港大學自然語言處理實驗室
Natural Language Processing Group, The University of Hong Kong

Thanks!