Edge AI Workshop

Lecture II - Language Models

Workshop Overview

1. Part 0: Introduction to the Team & Rust for Edge AI
2. Part I: Lecture on Computer Vision
    a. Main problems in Computer Vision
    b. What exactly is a neural network? (CNNs / Transformers)
    c. What exactly is an image embedding?
    d. Computer vision on the Edge
3. **Hands-On I: Air-gapped Face recognition on the Pi**
4. Part II: Lecture on Natural Language Processing
    a. A bit of history & development of modern LLMs
    b. How does an LLM work? Tokenizers, pretraining, post-training
    c. Context Engineering: Tool calling, RAG
    d. Libraries: tokenizers-rs, llama.cpp
5. Hands-On II: Chat with a LLM on Pi

Workshop Overview

1. **Part 0: Introduction to the Team & Rust for Edge AI**
2. **Part I: Lecture on Computer Vision**
   a. Main problems in Computer Vision
   b. What exactly is a neural network? (CNNs / Transformers)
   c. What exactly is an image embedding?
   d. Computer vision on the Edge
3. **Hands-On I: Air-gapped Face recognition on the Pi**
4. **Part II: Lecture on Natural Language Processing**
   a. A bit of history & development of modern LLMs
   b. How does an LLM work? Tokenizers, pretraining, post-training
   c. Context Engineering: Tool calling, RAG
   d. Libraries: tokenizers-rs, llama.cpp
5. **Hands-On II: Chat with a LLM on Pi**

This lecture

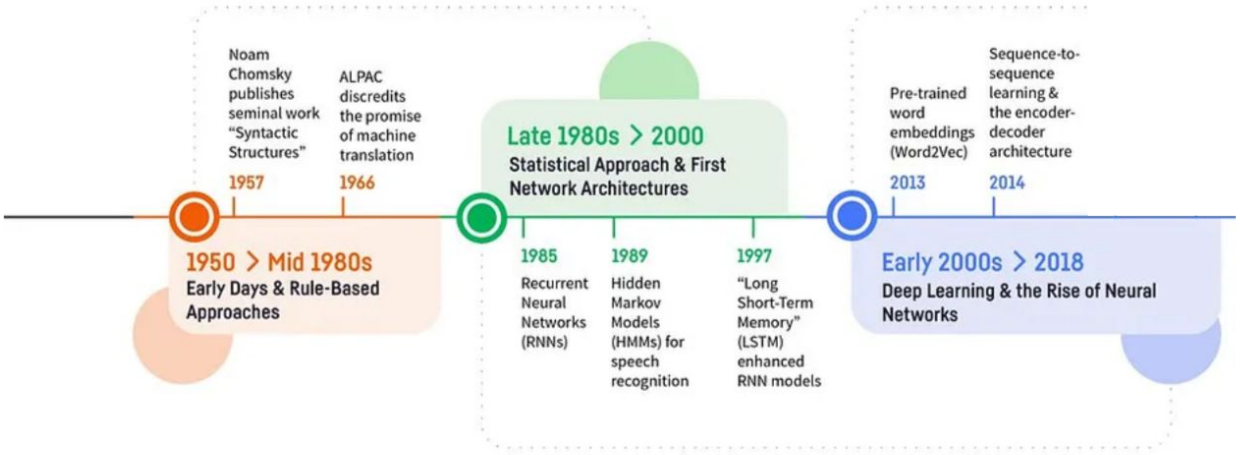
- The historical developments of the ideas behind modern language models
  - Mostly from OpenAI, since their path was quite clear and their models were highly influential to the research community

- High level overview of how modern LLMs are trained and fine-tuned

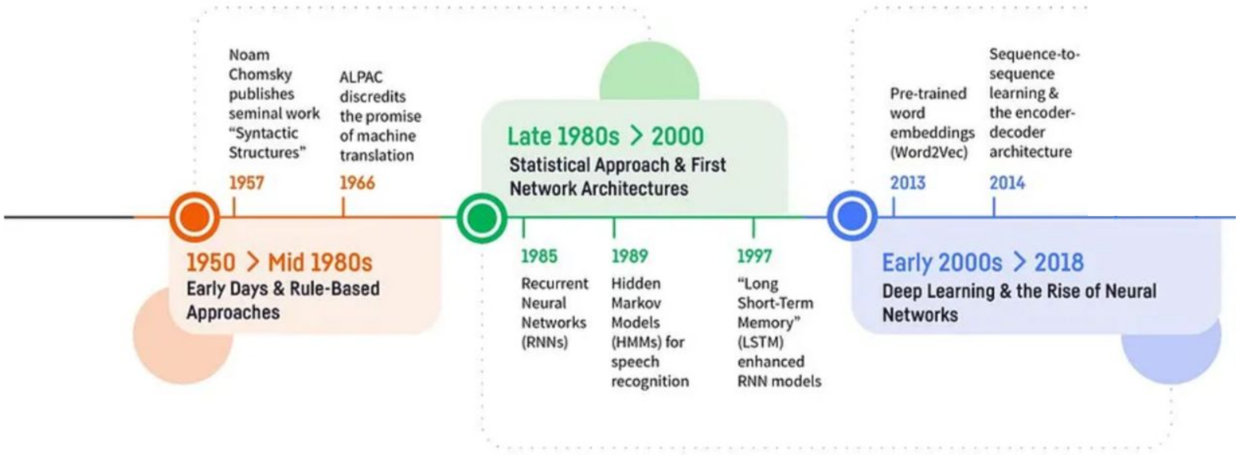
- Prompting methods

- Patterns for deployment on the edge

The old times (pre 2017)

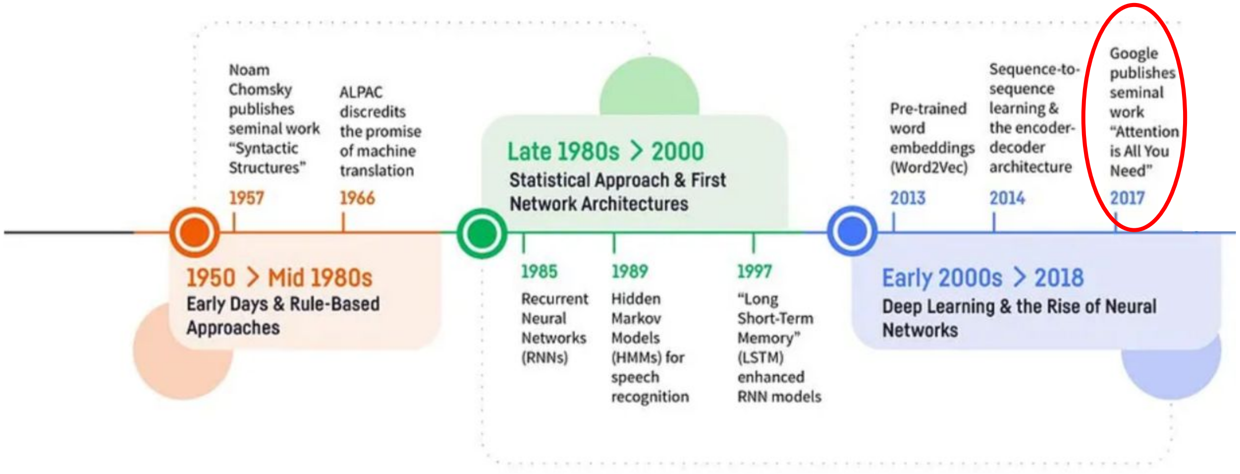


The old times (pre 2017)



- Each problem has its own architecture, with small quirks and problem-specific design choices
- Both in NLP and Computer Vision

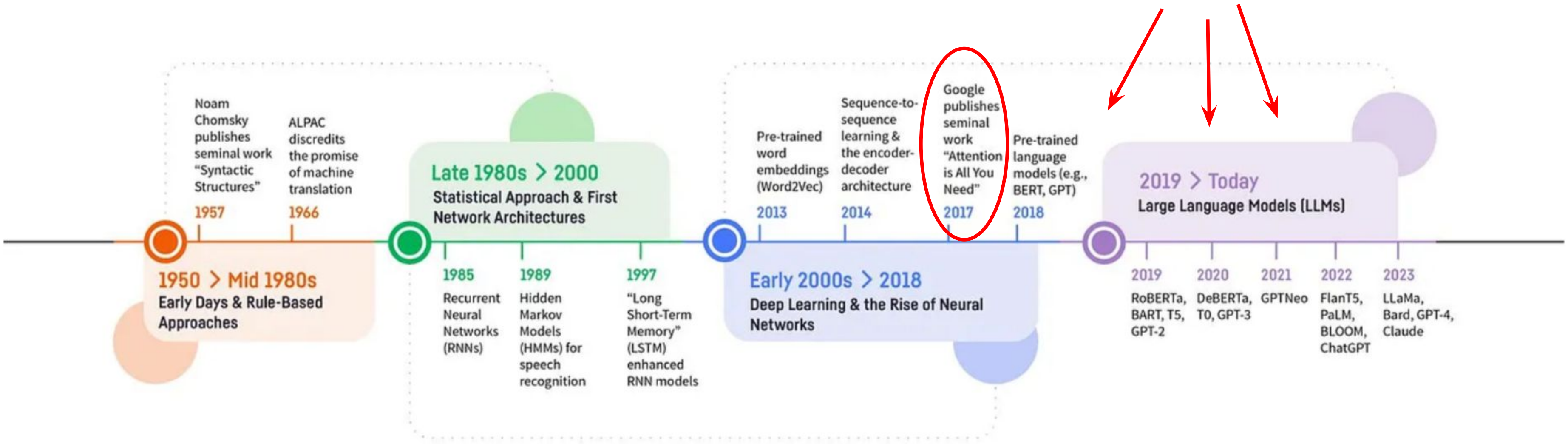
The old times (pre 2017)



- Each problem has its own architecture, with small quirks and problem-specific design choices
- Both in NLP and Computer Vision
- 2017 - Google published **"Attention is all you need"**

Now (post 2017)

Almost all models are transformers!



- Each problem has its own architecture, with small quirks and problem-specific design choices
- Both in NLP and Computer Vision
- 2017 - Google published **"Attention is all you need"**

The story of deep learning in a sentence

Old methods, developed in the ‘80s and ‘90s, when scaled up, started to work.

"We think the most benefits will go to whoever has the biggest computer."

Greg Brockman, OpenAI's CTO

"Compute is getting cheaper faster than we are becoming better researchers"

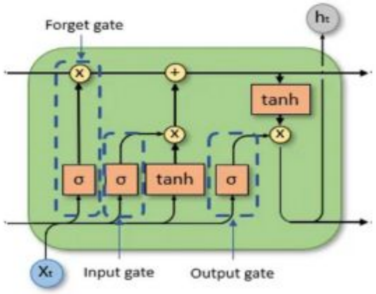
Hyung Won Chung
Research Scientist at OpenAI

Why now?

Past

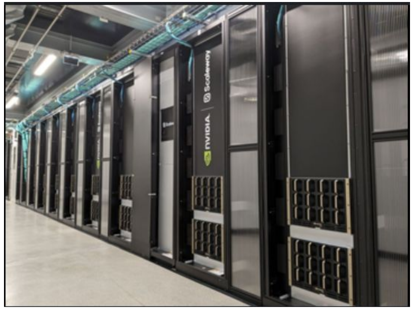
+

LSTM

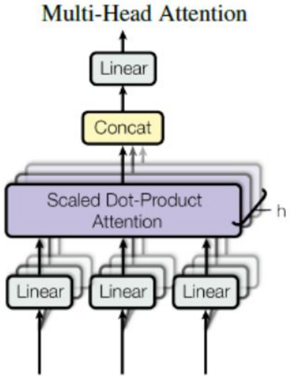


= Doesn't Work

Present



+



= Works

Received the “Test of Time Award” at NeurIPS 2024!

Seq2Seq, Google, 2014

- Use LSTM as encoder-decoder
- Input is a sequence → output is a sequence
- Solve machine translation → solve language modelling

- *“The success of our simple LSTM-based approach on MT suggests that it should do well on many other sequence learning problems, **provided they have enough training data**.”*

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT’14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM’s BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this

Pretraining on text - Next Token Prediction

- Given some words, what comes word next?

- Examples:
    - One, two, three, four … [?]

Pretraining on text - Next Token Prediction

- Given some words, what comes word next?

- Examples:
  - One, two, three, four … [?]
  - The capital of France is … [?]

Pretraining on text - Next Token Prediction

- Given some words, what comes word next?

- Examples:
  - One, two, three, four … [?]
  - The capital of France is … [?]
  - 25 + 13 = … [?]

Pretraining on text - Next Token Prediction

- Given some words, what comes word next?

- Examples:
  - One, two, three, four … [?]
  - The capital of France is … [?]
  - 25 + 13 = … [?]
  - $\int \frac{1}{x}\, dx =$ … [?]

Pretraining on text - Next Token Prediction

- Given some words, what comes word next?

- Examples:
  - One, two, three, four … [?]
  - The capital of France is … [?]
  - 25 + 13 = … [?]
  - $\int \frac{1}{x}\, dx =$ … [?]
  - The solution to the Riemann Hypothesis is … [?]

A form of **inductive reasoning**

Pretraining on text - Next Token Prediction

- Given some words, what comes word next?
- Train the model to do this for every word in a sentence.
    - [?] is computed as a classification problem over the vocabulary

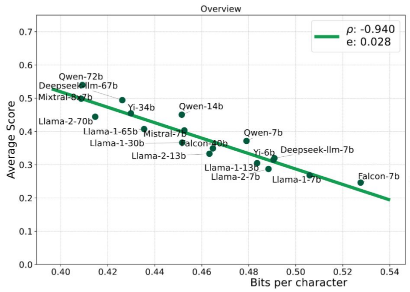
    - The … [?]
    - The capital … [?]
    - The capital of … [?]
    - The capital of France … [?]
    - The capital of France is … [?]

A form of (pretext-based) self-supervised learning!

Next Token Prediction

- Simple to describe and implement
- Difficult for the model to do accurately
- For the model to accurately predict what comes next, it needs to “understand” the world
  - I.e., **compress** the information in the corpus

Intelligence = Compression



Compression Represents Intelligence Linearly

Yuzhen Huang[*1] **Jinghan Zhang**[*1] **Zifei Shan**[2] **Junxian He**[1]
[1]The Hong Kong University of Science and Technology [2]Tencent
{yhuanghj,jzhangjv,junxianh}@cse.ust.hk

Abstract

There is a belief that learning to compress well will lead to intelligence (Hutter, 2006). Recently, language modeling has been shown to be equivalent to compression, which offers a compelling rationale for the success of large language models (LLMs): the development of more advanced language models is essentially enhancing compression which facilitates intelligence. Despite such appealing discussions, little empirical evidence is present for the interplay between compression and intelligence. In this work, we examine their relationship in the context of LLMs, treating LLMs as data compressors. Given the abstract concept of “intelligence”, we adopt the average downstream benchmark scores as a surrogate, specifically targeting intelligence related to knowledge and commonsense, coding, and mathematical reasoning. Across 12 benchmarks, our study brings together 30 public LLMs that originate from diverse organizations. Remarkably, we find that LLMs’ intelligence – reflected by average benchmark scores – almost *linearly* correlates with their ability to compress external text corpora. These results provide concrete evidence supporting the belief that superior compression indicates greater intelligence. Furthermore, our findings suggest that compression efficiency, as an unsupervised metric derived from raw text corpora, serves as a reliable evaluation measure that is linearly associated with the model capabilities. We open-source our compression datasets as well as our data collection pipelines to facilitate future researchers to assess compression properly.[1]

2015 - Ideas were already in the collective consciousness

Andrej Karpathy blog

About

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

There's something magical about Recurrent Neural Networks (RNNs). I still remember when I trained my first recurrent network for Image Captioning. Within a few dozen minutes of training my first baby model (with rather arbitrarily-chosen hyperparameters) started to generate very nice looking descriptions of images that were on the edge of making sense. Sometimes the ratio of how simple your model is to the quality of the results you get out of it blows past your expectations, and this was one of those times. What made this result so shocking at the time was that the common wisdom was that RNNs were supposed to be difficult to train (with more experience I've in fact reached the opposite conclusion). Fast forward about a year: I'm training RNNs all the time and I've witnessed their power and robustness many times, and yet their magical outputs still find ways of amusing me. This post is about sharing some of that magic with you.

> *We'll train RNNs to generate text character by character and ponder the question "how is that even possible?"*

By the way, together with this post I am also releasing code on Github that allows you to train character-level language models based on multi-layer LSTMs. You give it a large chunk of text and it will learn to generate text like it one character at a time. You can also use it to reproduce my experiments below. But we're getting ahead of ourselves; What are RNNs anyway?

Recurrent Neural Networks

Sequences. Depending on your background you might be wondering: *What makes Recurrent Networks so special*? A glaring limitation of Vanilla Neural Networks (and also Convolutional Networks) is that their API is too constrained: they accept a fixed-sized vector as input (e.g. an image) and produce a fixed-sized vector as output

Linux Source Code

I wanted to push structured data to its limit, so for the final challenge I decided to use code. In particular, I took all the source and header files found in the Linux repo on Github, concatenated all of them in a single giant file (474MB of C code) (I was originally going to train only on the kernel but that by itself is only ~16MB). Then I trained several as-large-as-fits-on-my-GPU 3-layer LSTMs over a period of a few days. These models have about 10 million parameters, which is still on the lower end for RNN models. The results are superfun:

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

Wikipedia

We saw that the LSTM can learn to spell words and copy general syntactic structures. Lets further increase the difficulty and train on structured markdown. In particular, lets take the Hutter Prize 100MB dataset of raw Wikipedia and train an LSTM. Following Graves et al., I used the first 96MB for training, the rest for validation and ran a few models overnight. We can now sample Wikipedia articles! Below are a few fun excerpts. First, some basic markdown output:

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
    siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

The sentiment neuron - OpenAI, 2017

- Train byte-level LSTM to generate IMDB reviews, unsupervised
- 4 Pascal GPUs for a month

- “we find a single unit which performs sentiment analysis”
    - A particular neuron activates when the review in positive and deactivates when the review is negative

Learning to Generate Reviews and Discovering Sentiment

Alec Radford [1] Rafal Jozefowicz [1] Ilya Sutskever [1]

Abstract

We explore the properties of byte-level recurrent language models. When given sufficient amounts of capacity, training data, and compute time, the representations learned by these models include disentangled features corresponding to high-level concepts. Specifically, we find a single unit which performs sentiment analysis. These representations, learned in an unsupervised manner, achieve state of the art on the binary subset of the Stanford Sentiment Treebank. They are also very data efficient. When using only a handful of labeled examples, our approach matches the performance of strong baselines trained on full datasets. We also demonstrate the sentiment unit has a direct influence on the generative process of the model. Simply fixing its value to be positive or negative generates samples with the corresponding positive or negative sentiment.

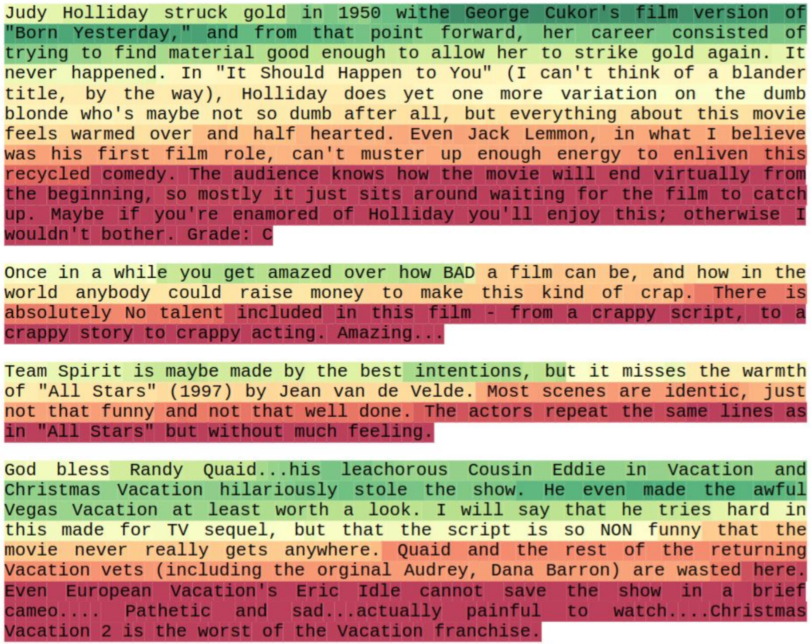
it is now commonplace to reuse these representations on a broad suite of related tasks - one of the most successful examples of transfer learning to date (Oquab et al., 2014).

There is also a long history of unsupervised representation learning (Olshausen & Field, 1997). Much of the early research into modern deep learning was developed and validated via this approach (Hinton & Salakhutdinov, 2006) (Huang et al., 2007) (Vincent et al., 2008) (Coates et al., 2010) (Le, 2013). Unsupervised learning is promising due to its ability to scale beyond only the subsets and domains of data that can be cleaned and labeled given resource, privacy, or other constraints. This advantage is also its difficulty. While supervised approaches have clear objectives that can be directly optimized, unsupervised approaches rely on proxy tasks such as reconstruction, density estimation, or generation, which do not directly encourage useful representations for specific tasks. As a result, much work has gone into designing objectives, priors, and architectures meant to encourage the learning of useful representations. We refer readers to Goodfellow et al. (2016) for a detailed

The sentiment neuron - OpenAI, 2017

- Train byte-level LSTM to generate IMDB reviews, unsupervised
- 4 Pascal GPUs for a month

- “we find a single unit which performs sentiment analysis”
    - A particular neuron activates when the review in positive and deactivates when the review is negative



The sentiment neuron - OpenAI, 2017

"… our work encourages further research into ***language modelling*** *as it demonstrates that the standard language modelling objective*

with no modifications

is sufficient to learn high-quality representations …"

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Attention Is All You Need

Adept
Ashish Vaswani*
Google Brain
avaswani@google.com

character.ai
Noam Shazeer*
Google Brain
noam@google.com

Adept
Niki Parmar*
Google Research
nikip@google.com

Inceptive
Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

co:here

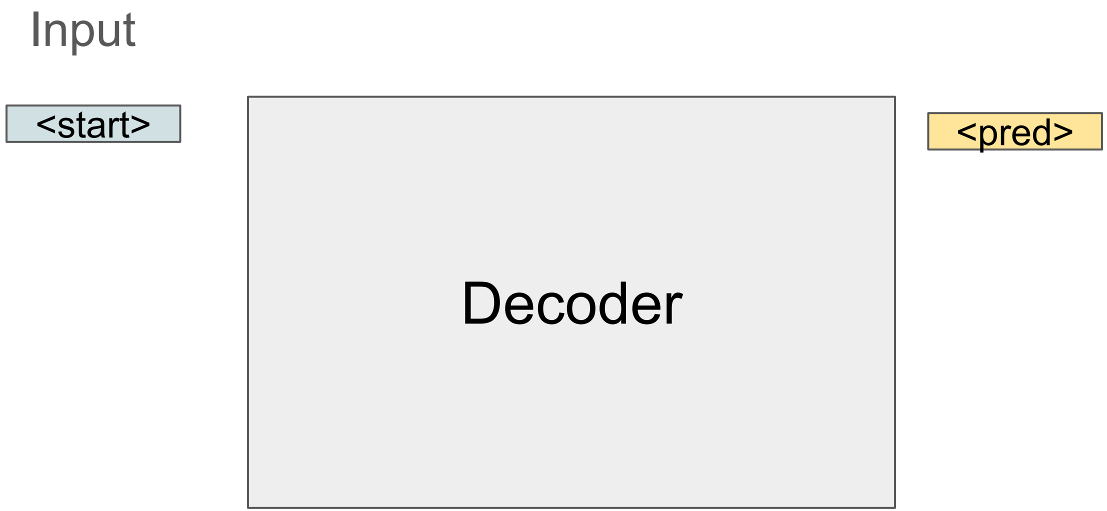
Illia Polosukhin* ‡
illia.polosukhin@gmail.com

NEAR
INCORPORATED

Transformer Decoder for Next Token Prediction

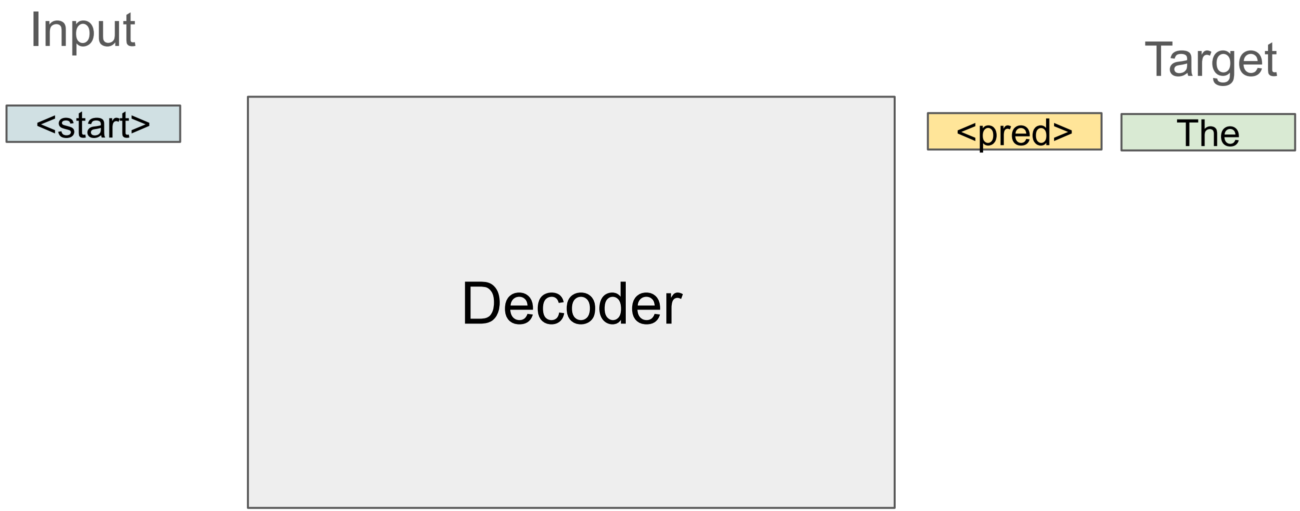
Transformer Decoder

Transformer Decoder for Next Token Prediction



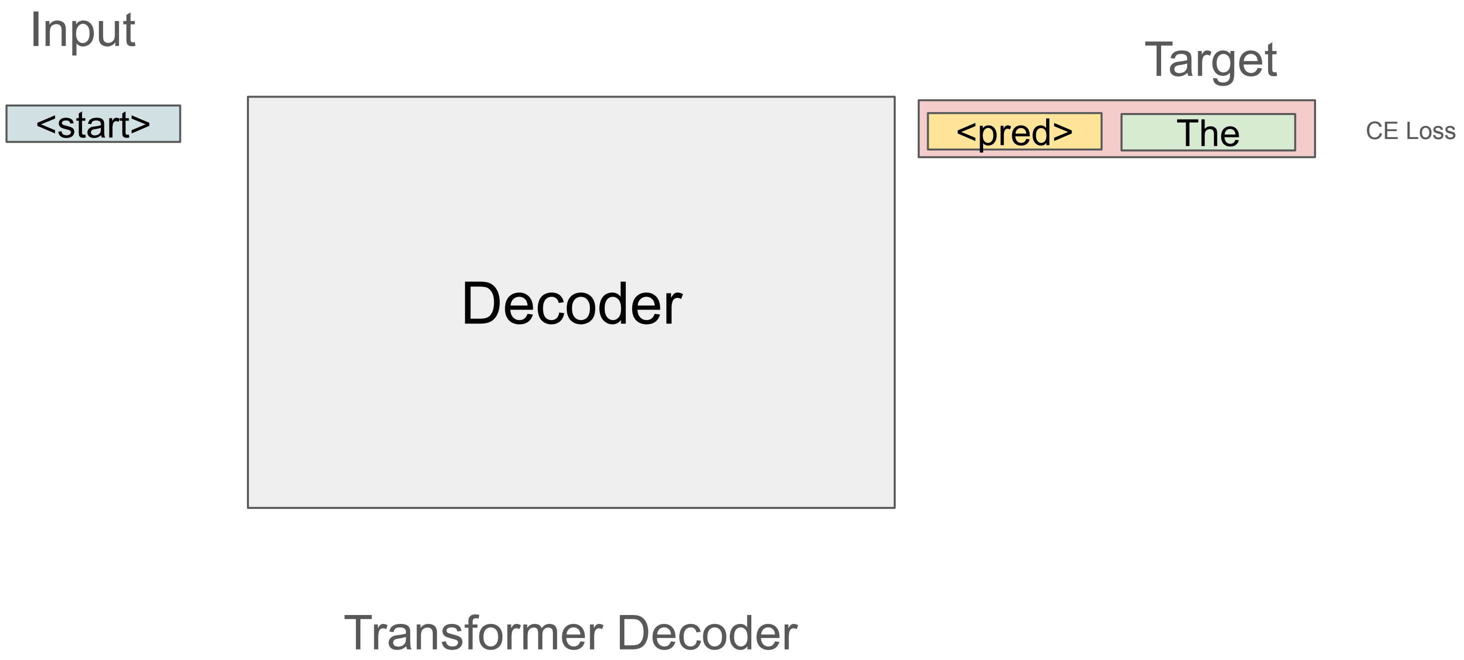
Transformer Decoder

Transformer Decoder for Next Token Prediction

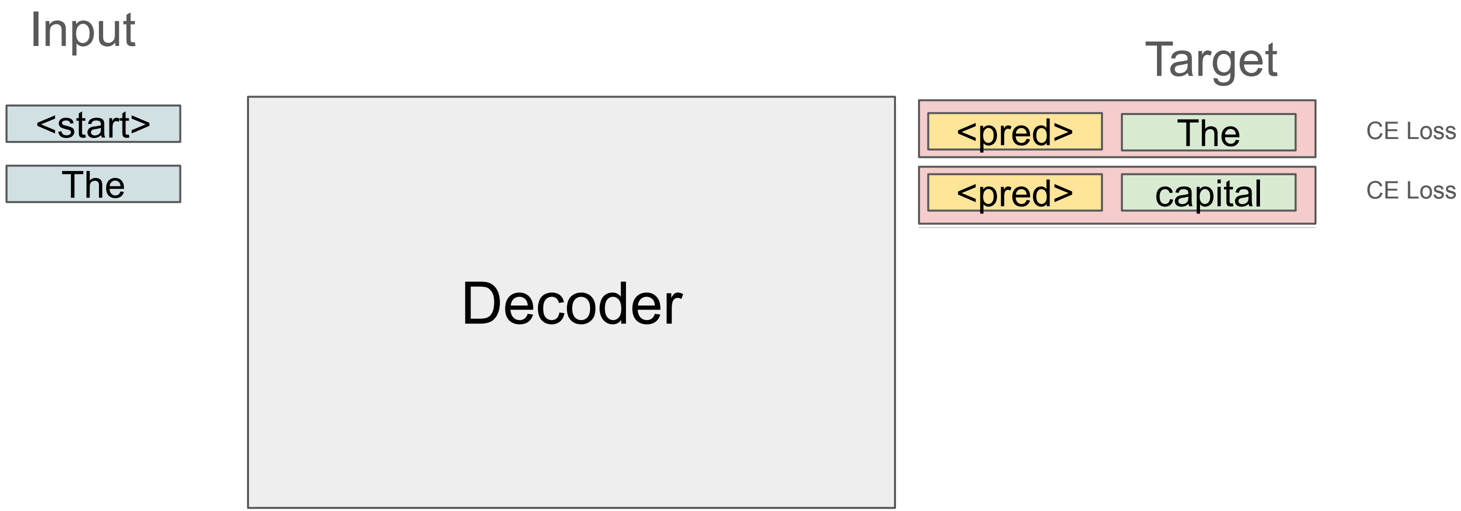


Transformer Decoder

Transformer Decoder for Next Token Prediction

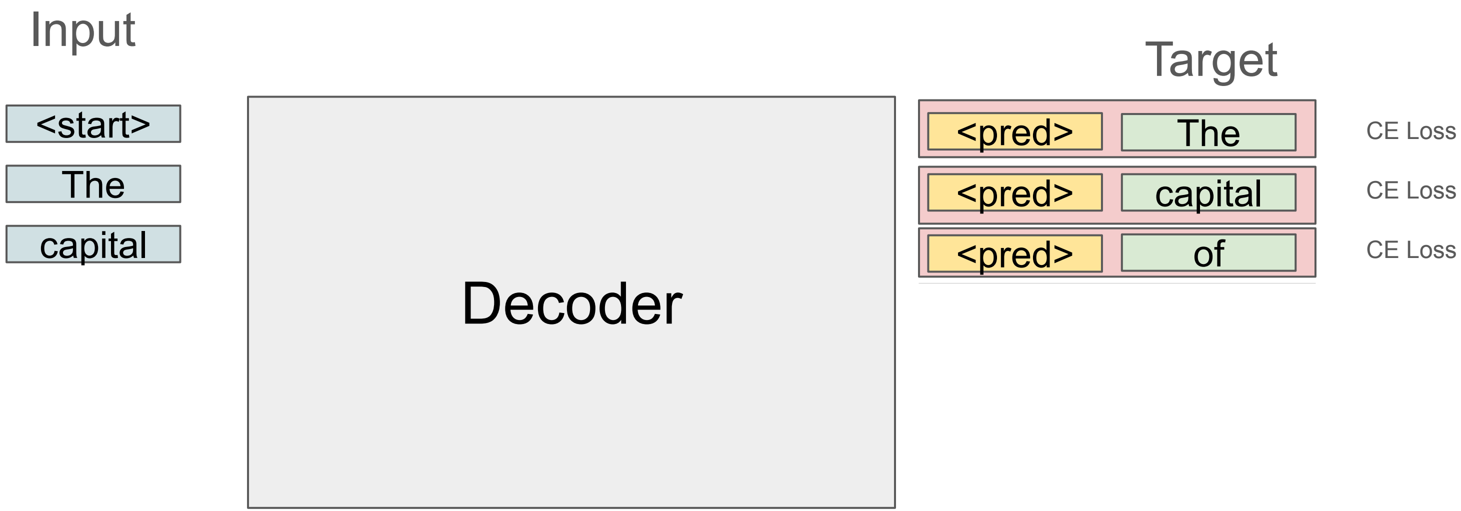


Transformer Decoder for Next Token Prediction



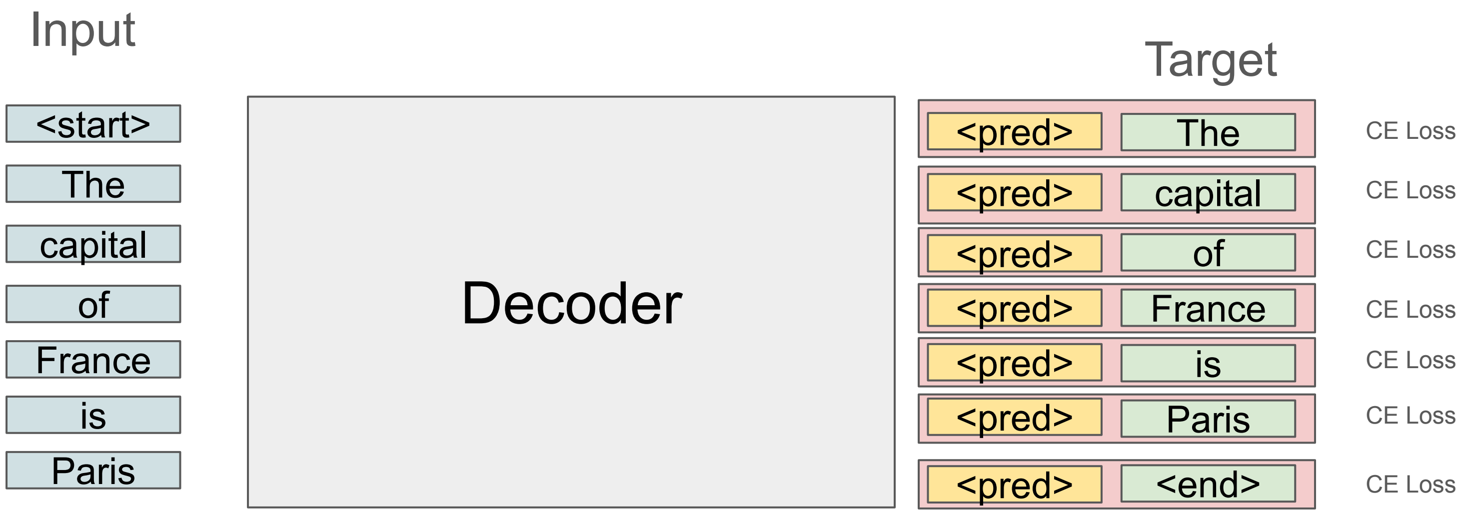
Transformer Decoder

Transformer Decoder for Next Token Prediction



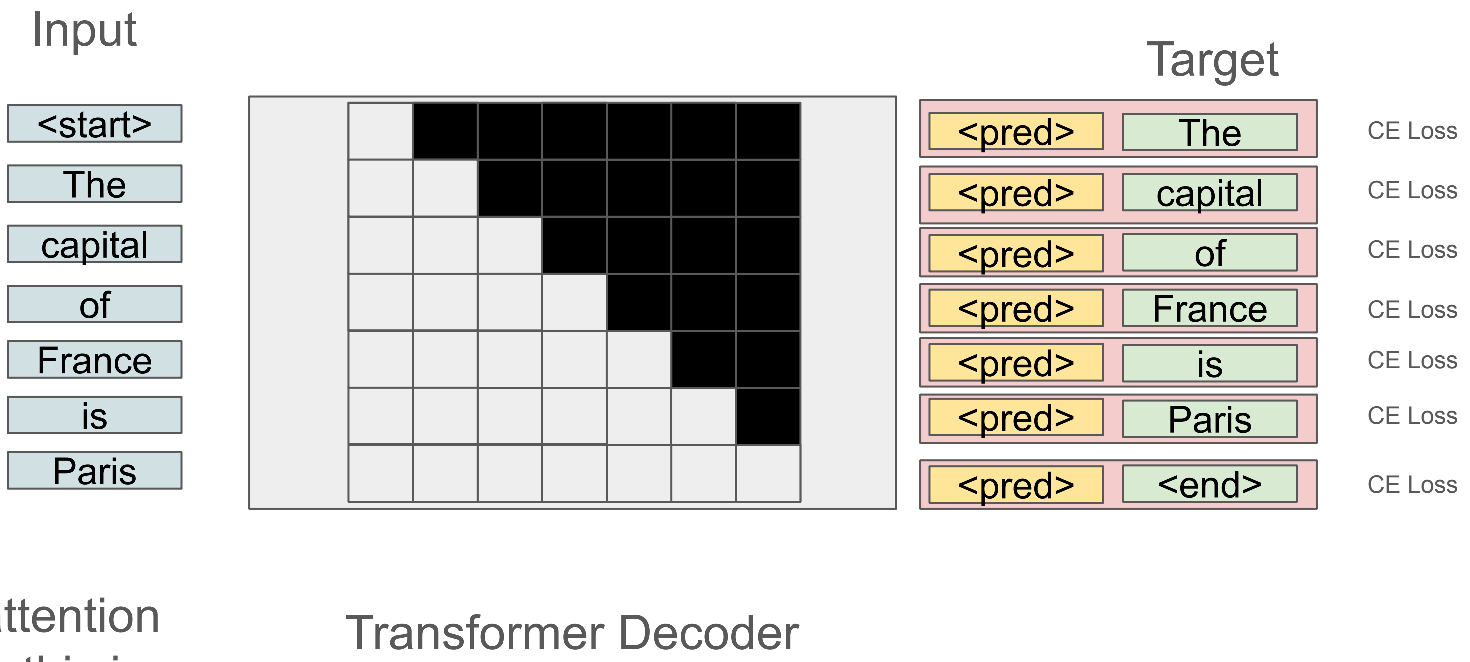
Transformer Decoder

Transformer Decoder for Next Token Prediction



Transformer Decoder

Transformer Decoder for Next Token Prediction



Using a causal attention mask, we can do this in a single forward pass!

Seq2Seq is all you need?

- People started to realize that all NLP problems can be cast as text in –> text out
- Example: Text-to-Text Transfer Transformer (T5), 2019

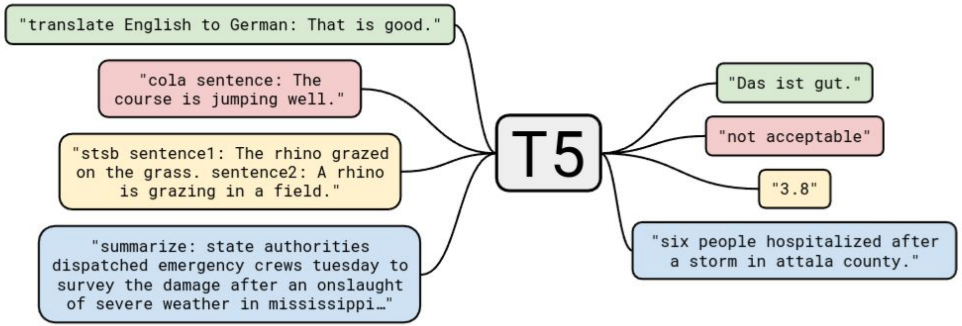


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

An aside: Tokenization

- Text tokenization = splitting a string into a sequence of tokens

- Most methods so far used either
  - **Word-level tokenization**
  - **Character-level tokenization**

- Both are inadequate
  - Splitting by words results in many “out-of-vocabulary“ words
    - Misspellings? Many edge cases.
  - Splitting text by characters results in very large sequence length which is computationally intractable

Tokenization: Word-level splitting

- What happens when we encounter a word that we have never seen in our training data?
  - Not much we can do
  - Assign a special <UNK> token
  - Lose a lot of meaning
  - Especially hurts in texts/languages with many rare words/entities

- “amazing!”, “state-of-the-art”, “un-thinkable”, “prize-winning”, “aren’t”, “O’Neill”

- Some languages don’t even use spaces to mark word boundaries!

Tokenization: Character-level splitting

- Vocabulary size is small
  - 256 entries
- No unknown tokens

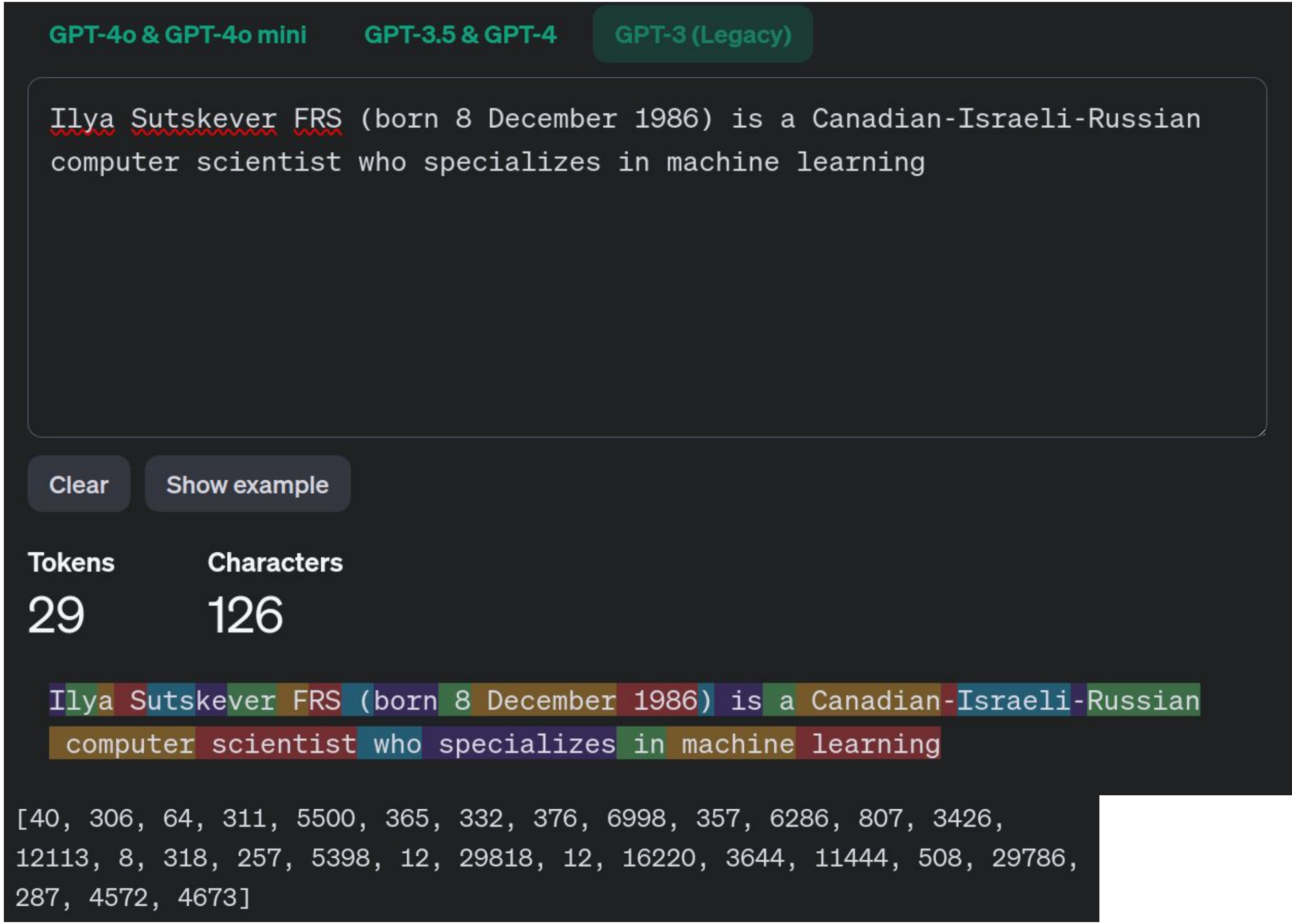
- However, sequence is much larger
- Need to learn from scratch how to combine characters into words

Byte-Pair-Encoding Tokenization: The middle ground

- Tokenize into subwords using BPE
- A **greedy** compression algorithm
- Vocabulary size fixed and specified by us
- No unknown words
    - Always fallback to small subwords

- $\mathscr{V} \leftarrow$ All characters in the training data (as base **tokens**)
- For k steps:
    - Tokenize the data, taking the longest prefix each time
    - Count the frequency of adjacent token pairs in the data
    - Choose the pair $\langle l, r \rangle$ that occurs most frequently
    - Add the pair to the vocabulary as a new token $\mathscr{V} \leftarrow \mathscr{V} \cup \{lr\}$
- Return $\mathscr{V}$

BPE - Example



GPT-4o & GPT-4o mini
GPT-3.5 & GPT-4
GPT-3 (Legacy)
Ilya Sutskever FRS (born 8 December 1986) is a Canadian-Israeli-Russian computer scientist who specializes in machine learning
Clear
Show example
Tokens
29
Characters
126
Ilya Sutskever FRS (born 8 December 1986) is a Canadian-Israeli-Russian computer scientist who specializes in machine learning
[40, 306, 64, 311, 5500, 365, 332, 376, 6998, 357, 6286, 807, 3426, 12113, 8, 318, 257, 5398, 12, 29818, 12, 16220, 3644, 11444, 508, 29786, 287, 4572, 4673]

HuggingFace tokenizers in Rust

tokenizers
0.21.4

- Official tokenizers library is built in Rust
- Python bindings
- Anyone who ever used a HuggingFace LLM has used this implementation

```rust
use tokenizers::tokenizer::{Result, Tokenizer};

fn main() -> Result<()> {
    // needs http feature enabled
    let tokenizer = Tokenizer::from_pretrained("bert-base-cased", None)?;

    let encoding = tokenizer.encode("Hey there!", false)?;
    println!("{:?}", encoding.get_tokens());
    Ok(())
}
```

GPT-1, OpenAI, 2018

- Pretrain decoder-only transformer on BPE tokenized text
    - 12 layers, dmodel = 768
    - Books corpus
    - 12 GPUs for a month

- Pretraining for with next-token prediction helps a lot!

- Fine-tune for classification last embedding

Improving Language Understanding by Generative Pre-Training

Alec Radford
OpenAI
alec@openai.com

Karthik Narasimhan
OpenAI
karthikn@openai.com

Tim Salimans
OpenAI
tim@openai.com

Ilya Sutskever
OpenAI
ilyasu@openai.com

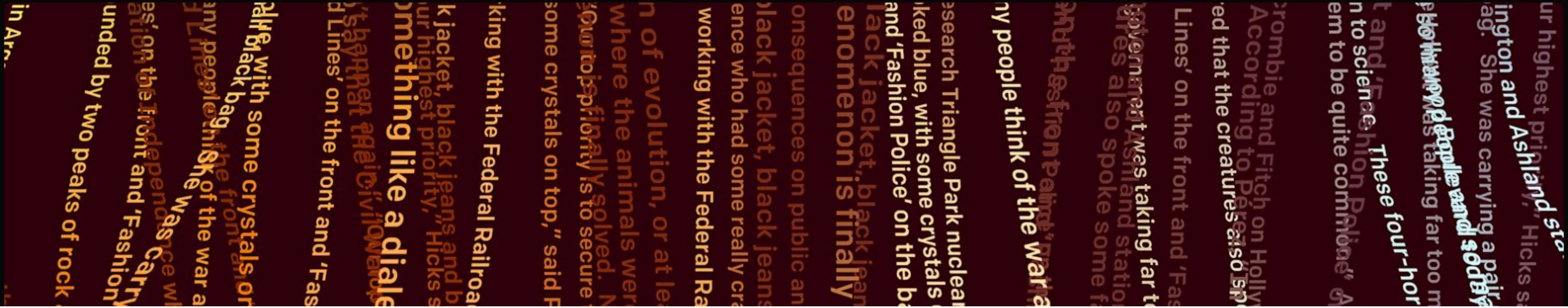
Abstract

Natural language understanding comprises a wide range of diverse tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. Although large unlabeled text corpora are abundant, labeled data for learning these specific tasks is scarce, making it challenging for discriminatively trained models to perform adequately. We demonstrate that large gains on these tasks can be realized by *generative pre-training* of a language model on a diverse corpus of unlabeled text, followed by *discriminative fine-tuning* on each specific task. In contrast to previous approaches, we make use of task-aware input transformations during fine-tuning to achieve effective transfer while requiring minimal changes to the model architecture. We demonstrate the effectiveness of our approach on a wide range of benchmarks for natural language understanding. Our general task-agnostic model outperforms discriminatively trained models that use architectures specifically crafted for each task, significantly improving upon the state of the art in 9 out of the 12 tasks studied. For instance, we achieve absolute improvements of 8.9% on commonsense reasoning (Stories Cloze Test), 5.7% on question answering (RACE), and 1.5% on textual entailment (MultiNLI).

February 14, 2019

Better language models and their implications

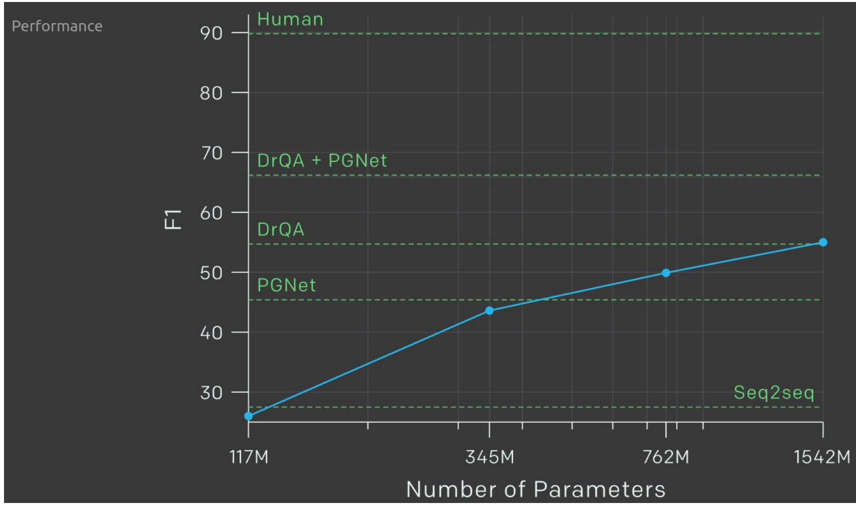
Read paper ↗ View code ↗



GPT-2, OpenAI, 2019

- Direct scale-up of GPT-1
- 1.5B parameters
- 40GB of text
  - Web pages that were linked on a reddit posts with at least 3 likes

- More model parameters improves performance



When people started to notice: GPT-3, OpenAI, 2020

- 175B parameters
- 300B tokens
- New capabilities **emerged**

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

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

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry
Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan
Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic,

Scaling law: performance follows a power-trend with compute

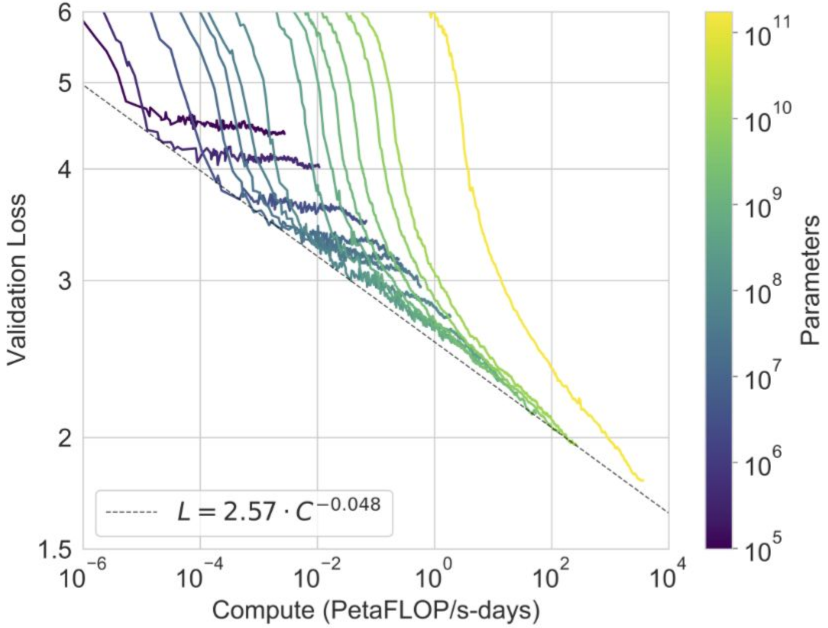
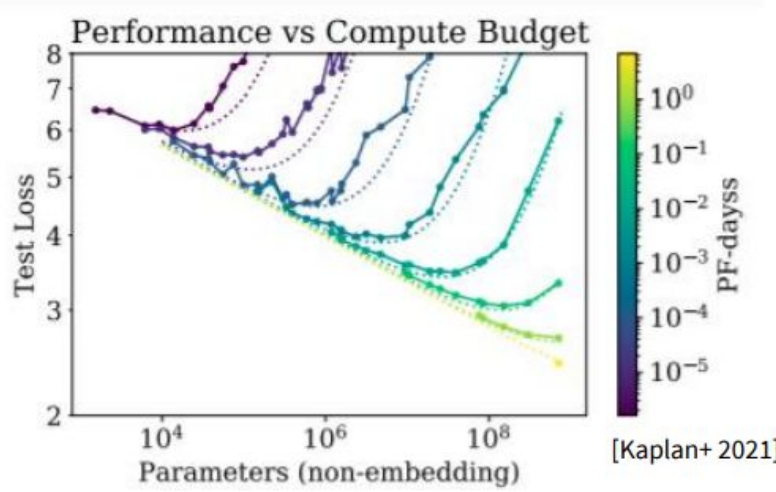
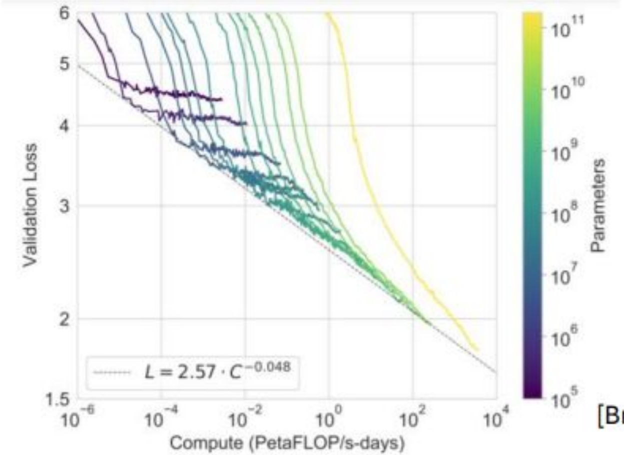


Figure 3.1: Smooth scaling of performance with compute. Performance (measured in terms of cross-entropy validation loss) follows a power-law trend with the amount of compute used for training. The power-law behavior observed in [KMH+20] continues for an additional two orders of magnitude with only small deviations from the predicted curve. For this figure, we exclude embedding parameters from compute and parameter counts.

Scaling is all you need!



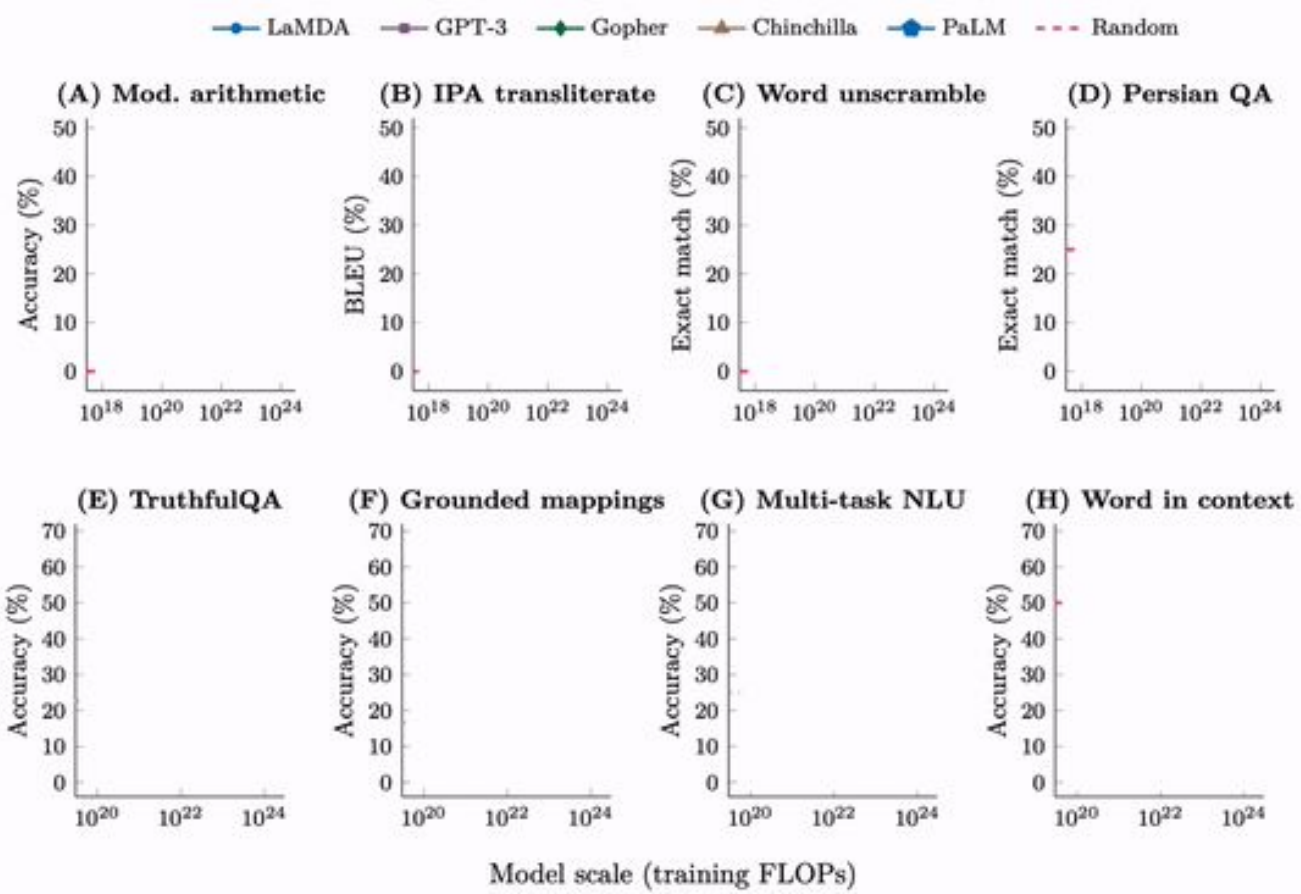
[Kaplan+ 2021]



[Brown+ 2020]

Bigger is not only better!

Bigger is different!



[Emergent Abilities of Large Language Models](#)

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph et al. (2022)

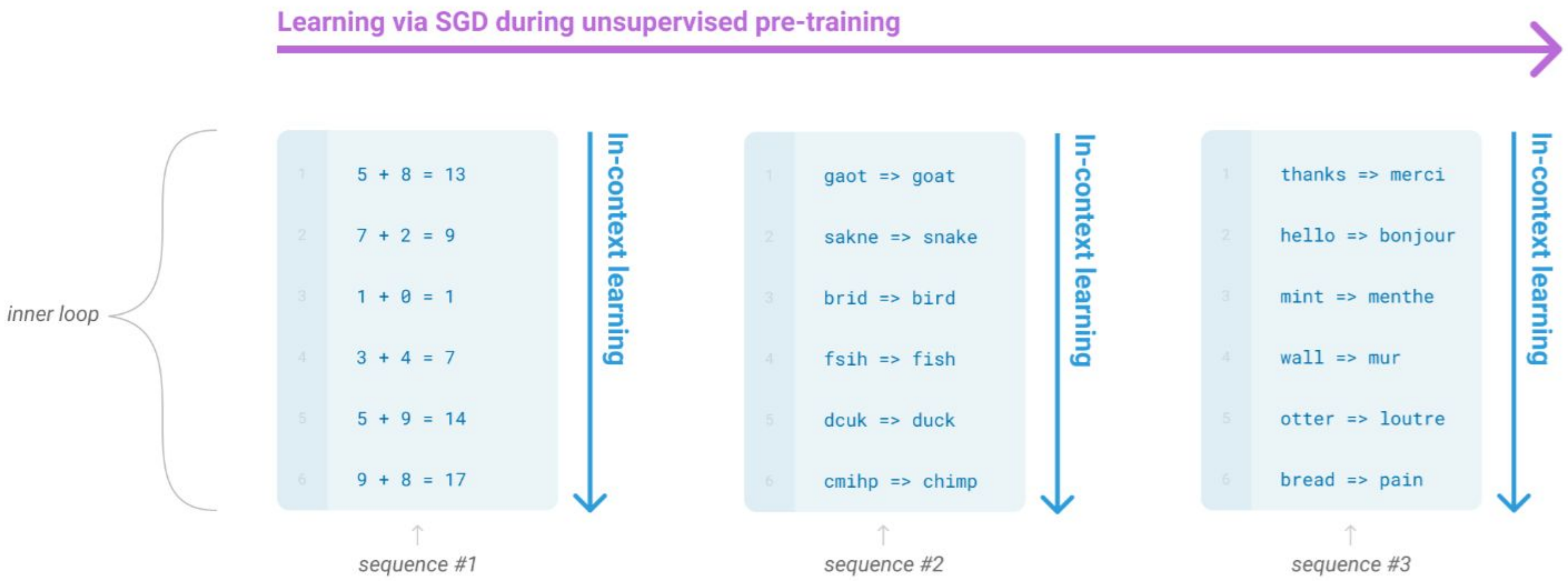
Paradigm shift: In-Context Learning

- Don’t need to fine-tune directly, put examples directly in the prompt.

- Paradigm shift.
  - Initially, people got data and trained a model on it
  - Then, general pretrained models (i.e., BERT) were made available, but they were useless
    - People still needed to fine-tune.

  - Now, fine-tuning became “obsolete”,
    - use model as-is, put examples in the prompt

Examples of “in-context-learning” in training data



Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

“Base-models” don’t follow instructions

Prompt	Explain the moon landing to a 6 year old in a few sentences.
Completion	GPT-3
	Explain the theory of gravity to a 6 year old.
	Explain the theory of relativity to a 6 year old in a few sentences.
	Explain the big bang theory to a 6 year old.
	Explain evolution to a 6 year old.

InstructGPT, OpenAI, 2022

January 27, 2022

Aligning language models to follow instructions

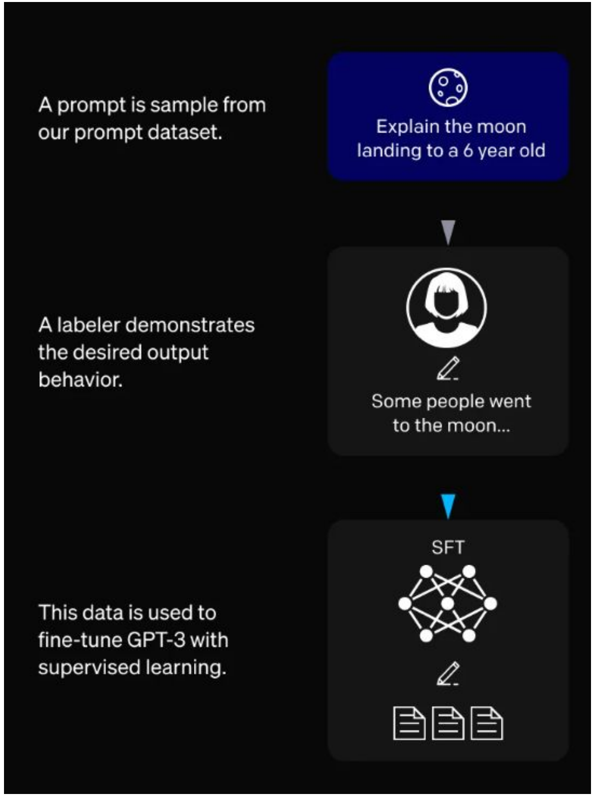
Read paper ↗ View model card ↗

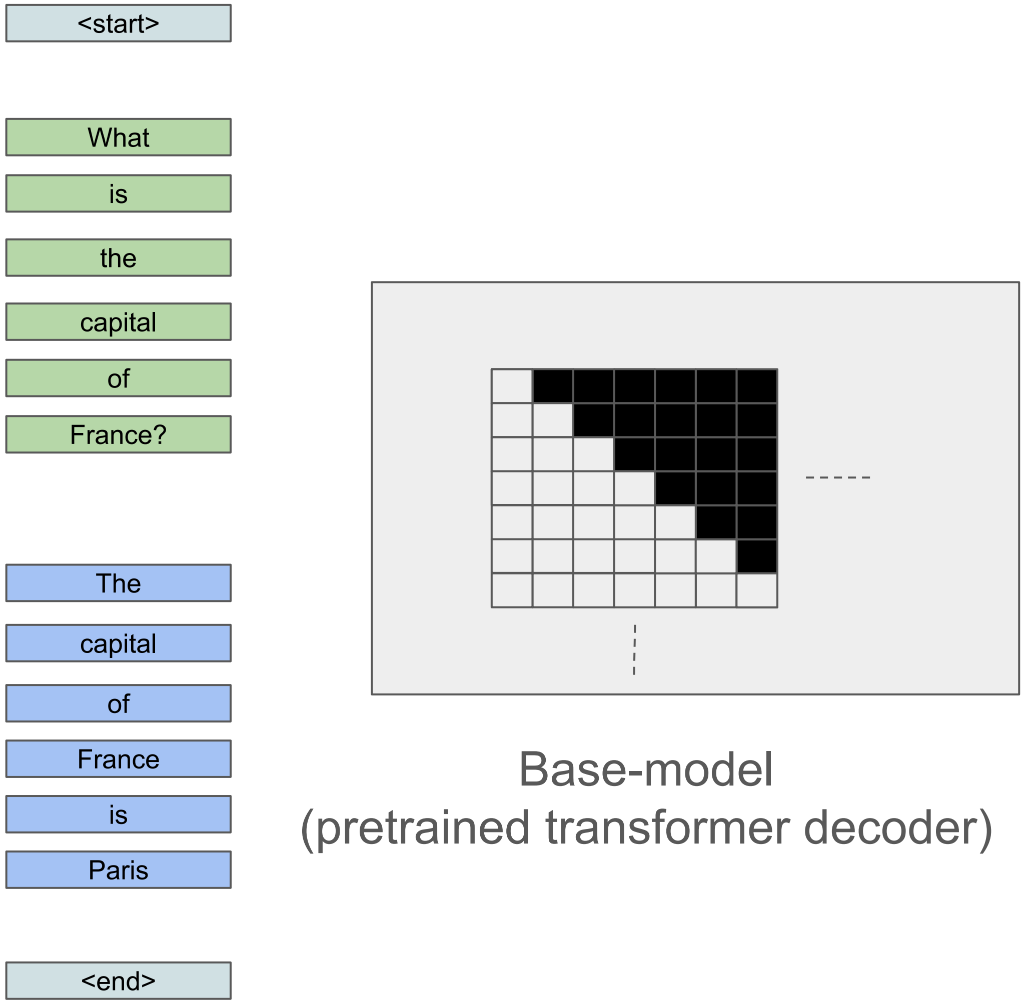
Supervised Fine-Tuning (SFT)

- Instruction-following

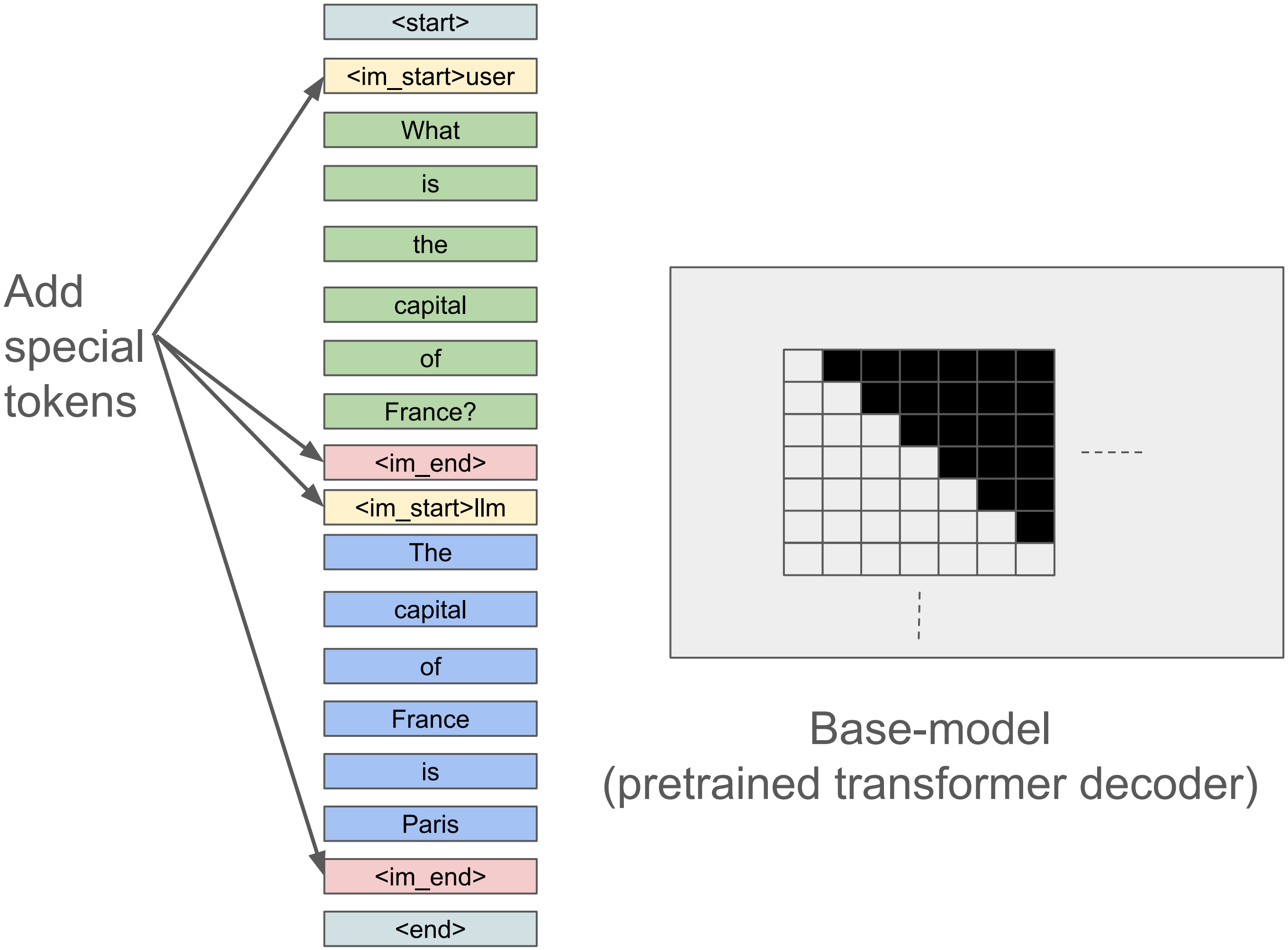
- Gather a dataset of instruction and human-generated responses

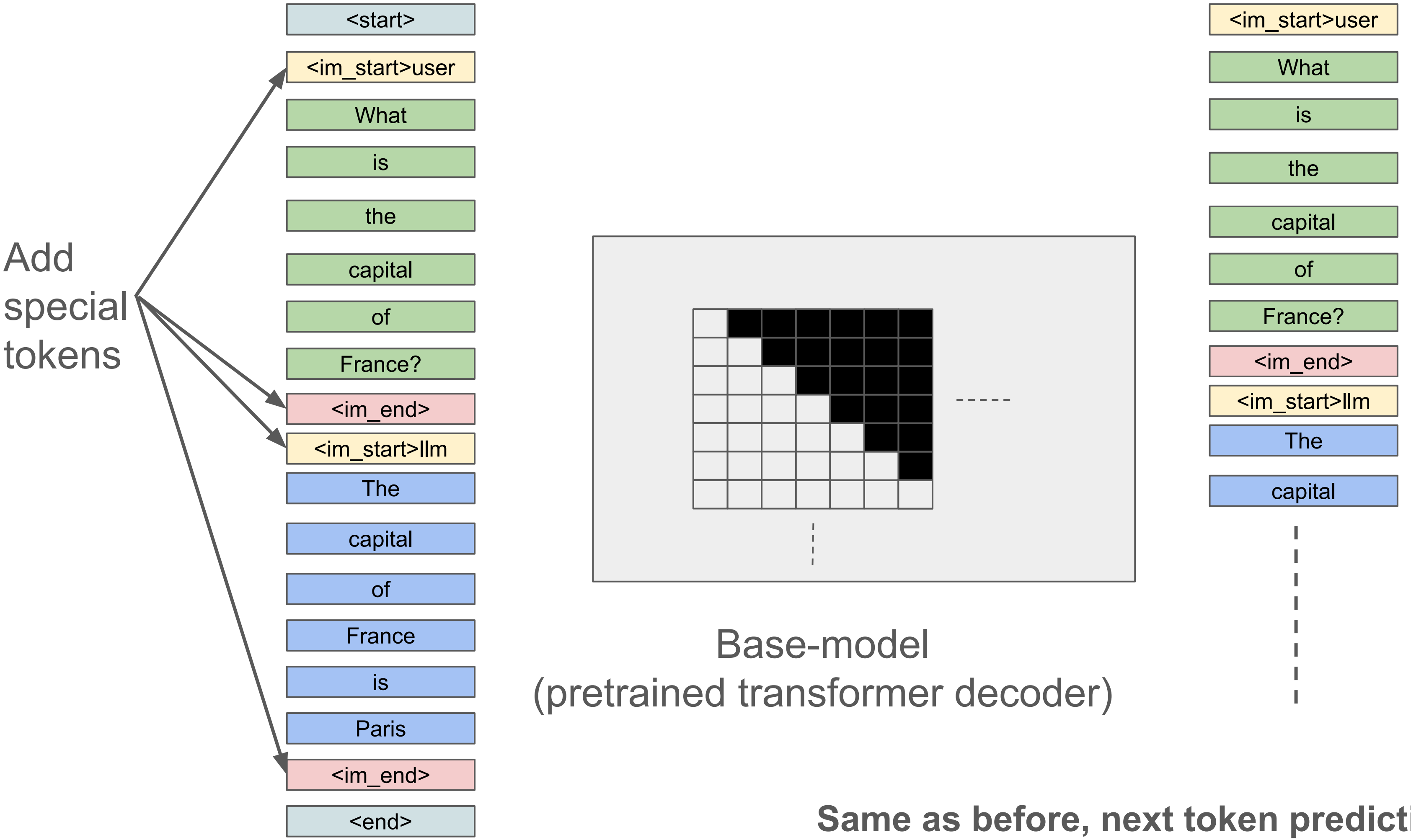
- Train the model to output the response





Base-model
(pretrained transformer decoder)


Add special tokens
<start>
<im_start>user
What
is
the
capital
of
France?
<im_end>
<im_start>llm
The
capital
of
France
is
Paris
<im_end>
<end>
Base-model
(pretrained transformer decoder)



Same as before, next token prediction!

Now, the model can follow instruction (i.e. chat)

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

Completion GPT-3

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

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

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

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

November 30, 2022

Introducing ChatGPT

Try ChatGPT ↗ Download ChatGPT desktop › Learn about ChatGPT ›

We've trained a model called ChatGPT which interacts in a conversational way. The dialogue format makes it possible for ChatGPT to answer followup questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests.

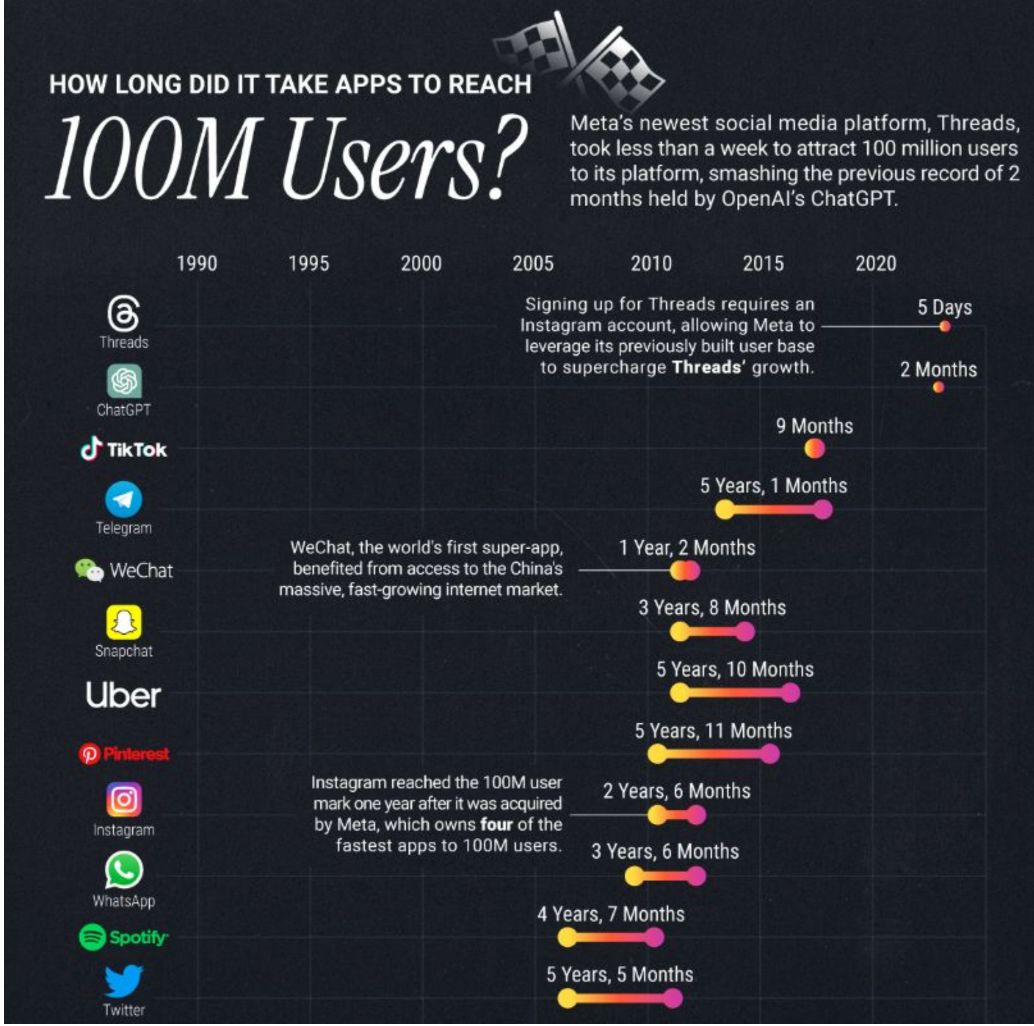
ChatGPT is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response.

We are excited to introduce ChatGPT to get users' feedback and learn about its strengths and weaknesses. During the research preview, usage of ChatGPT is free. Try it now at chatgpt.com.

ChatGPT, 2022

- 2 months to reach 100M users

- First time a chatbot can answer questions, refuse to answer, argue with you, “understands” you



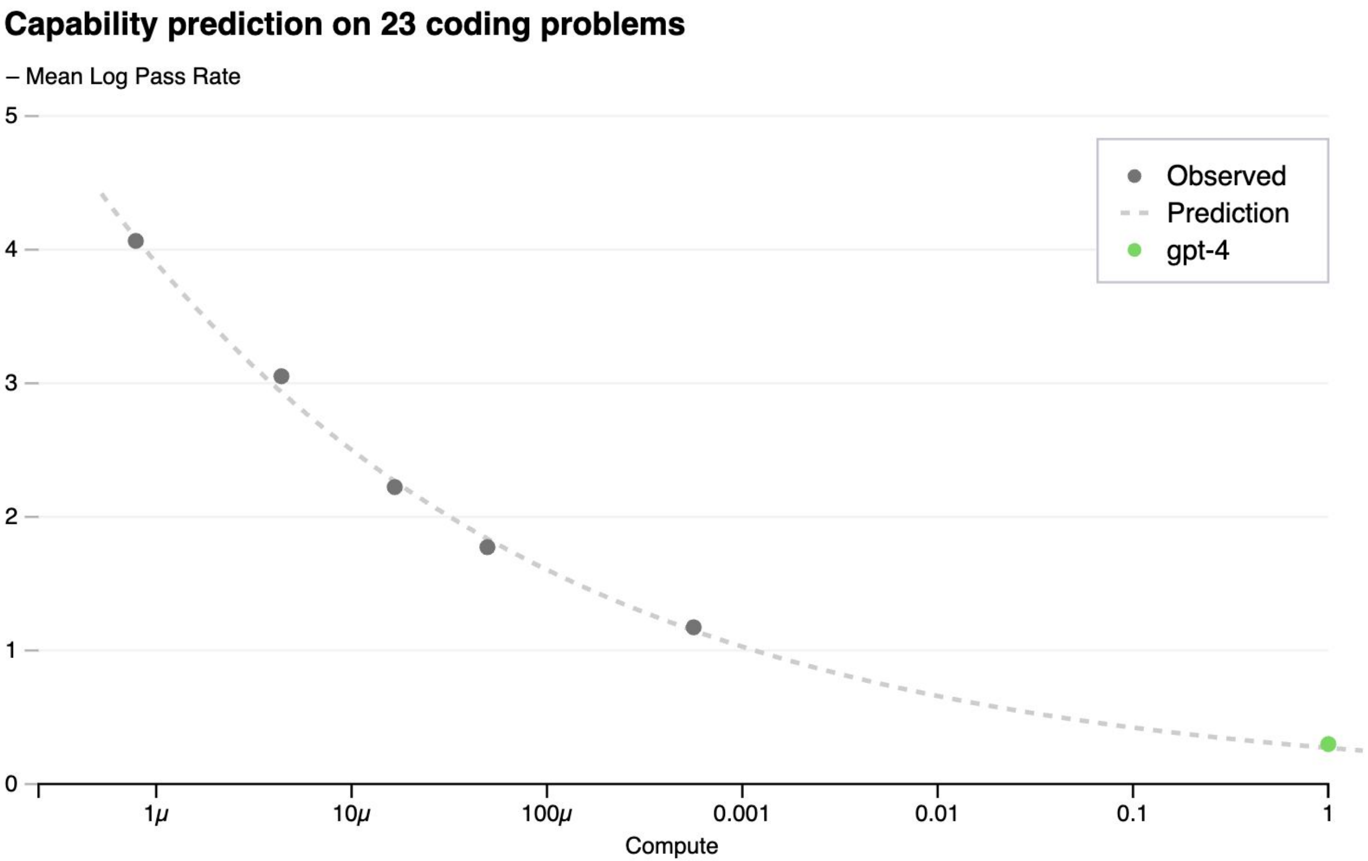
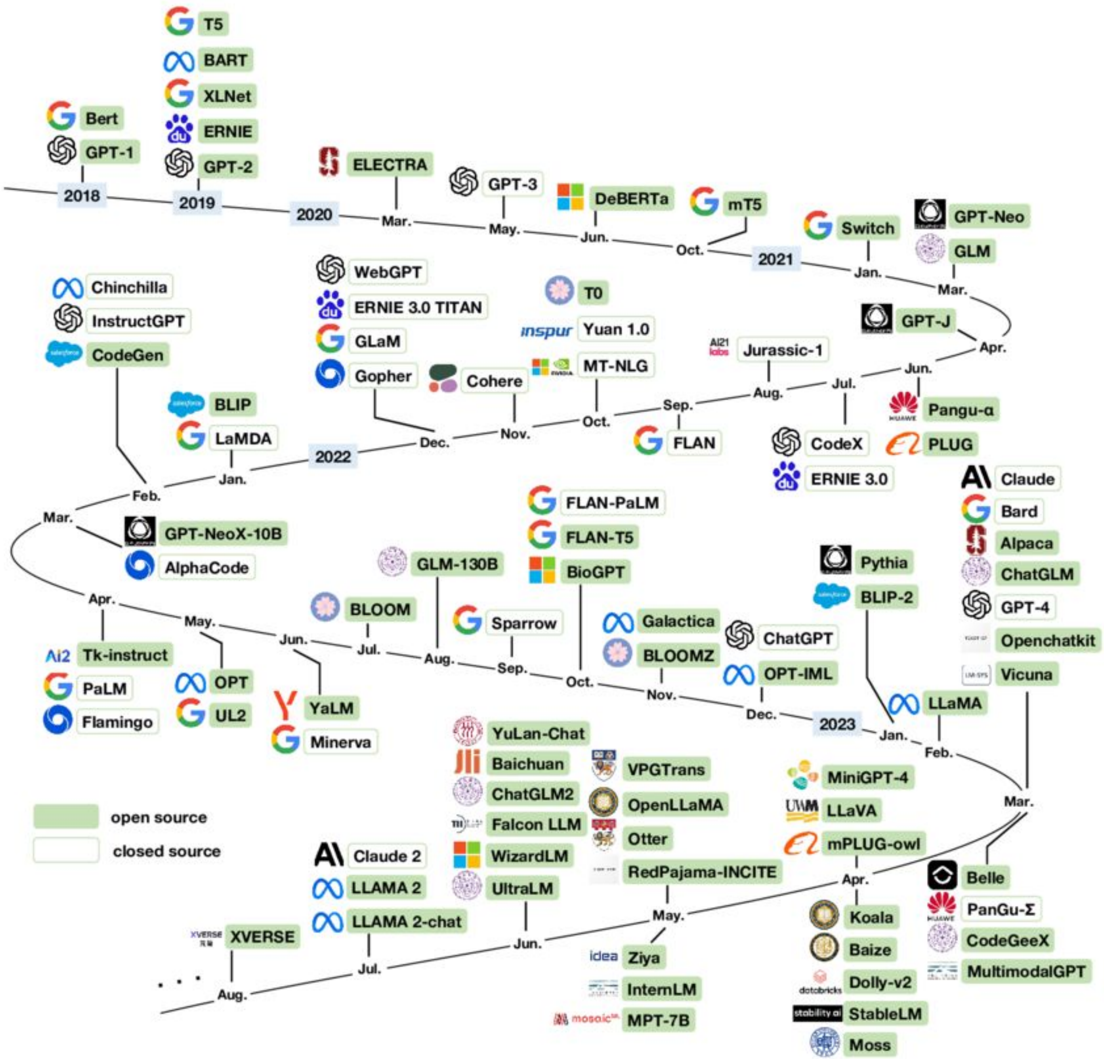


Figure 2. Performance of GPT-4 and smaller models. The metric is mean log pass rate on a subset of the HumanEval dataset. A power law fit to the smaller models (excluding GPT-4) is shown as the dotted line; this fit accurately predicts GPT-4's performance. The x-axis is training compute normalized so that GPT-4 is 1.

Cambrian Explosion of LLMs



Ilya Sutskever - NeurIPS talk 2024

Pre-training as we know it will end

Compute is growing:

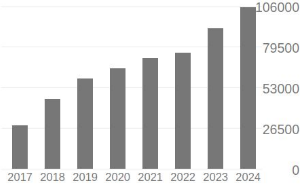
- Better hardware
- Better algorithms
- Larger clusters

Data is not growing:

- We have but one internet
- **The fossil fuel of AI**

Cited by VIEW ALL

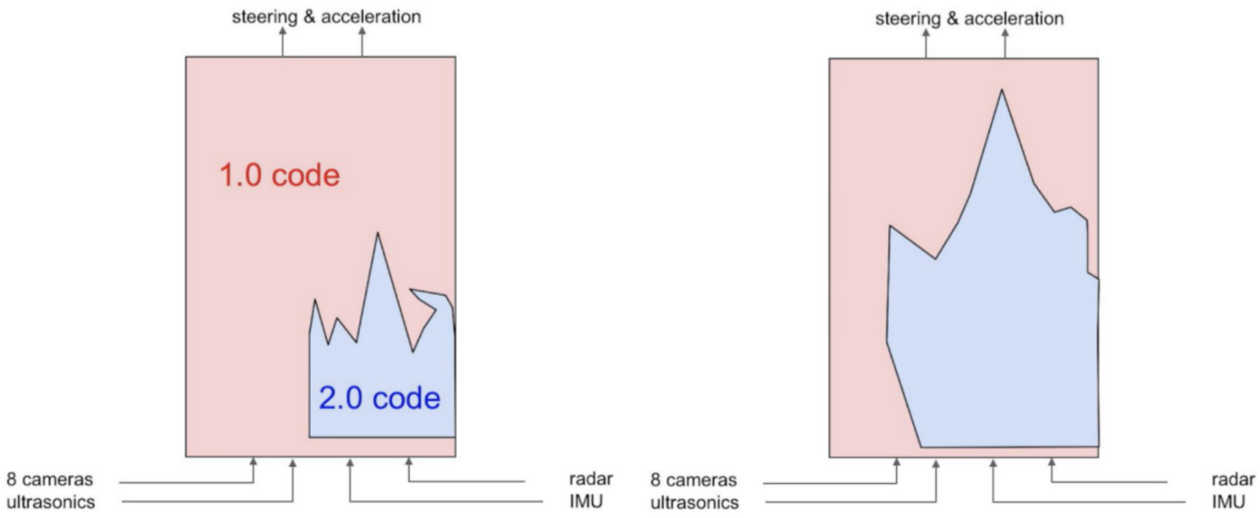
	All	Since 2019
Citations	581019	473083
h-index	93	88
i10-index	140	137



Recall from last lecture …

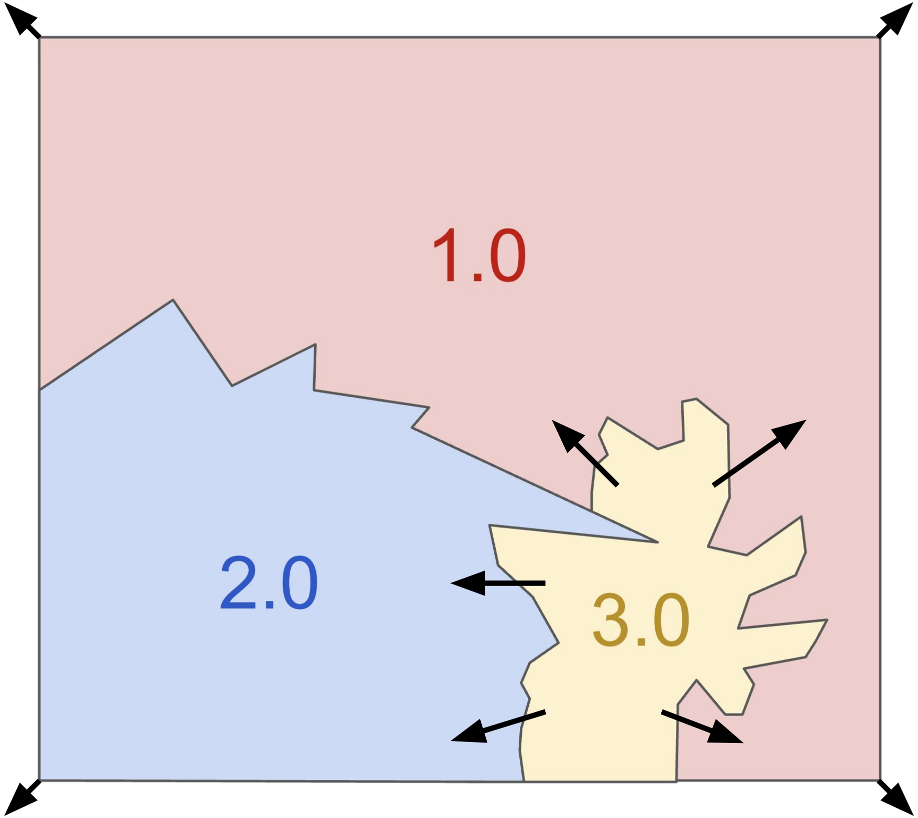
Software 1.0: “classical” software engineering

Software 2.0: (neural network) weights



Source: Andrej Karpathy

Software 3.0: Prompt / Context Engineering



Source: Andrej Karpathy

Test-time compute

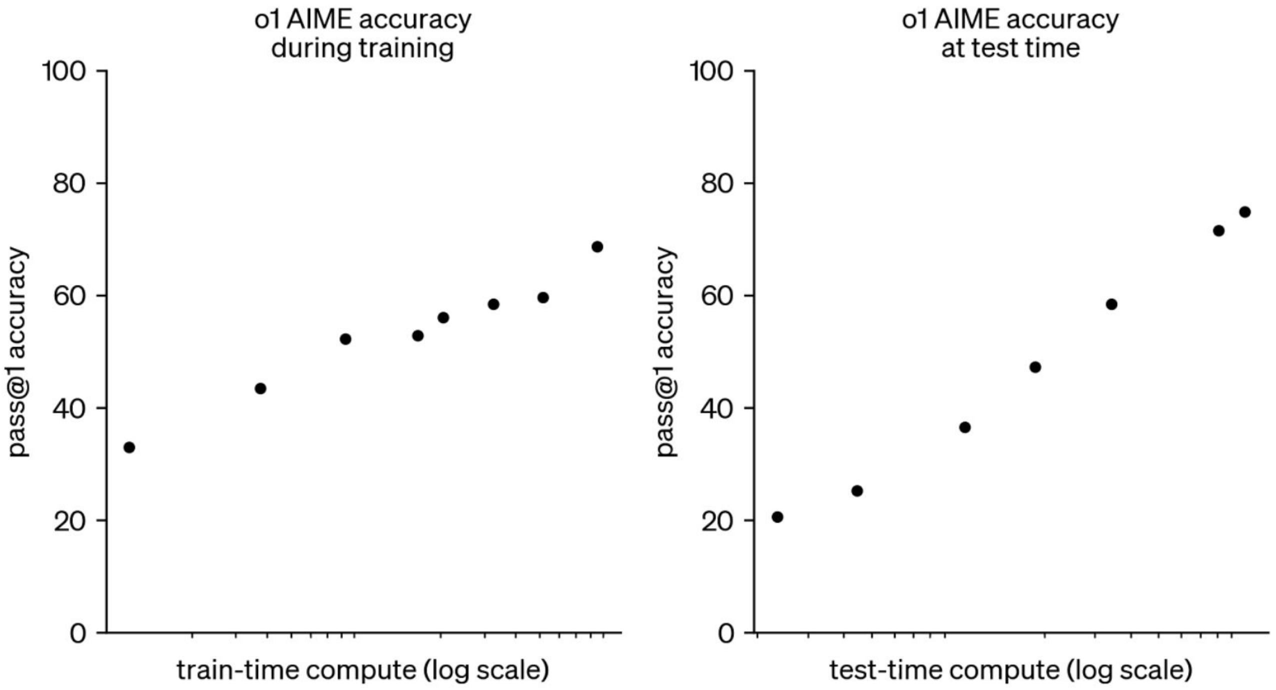
- We can trade test-time compute for model size
- Bigger models can store more knowledge

- But do we really need to store knowledge in the model? What if we can get answers by thinking / reasoning more vs memorizing?

- What if we can retrieve knowledge from an external source on demand?

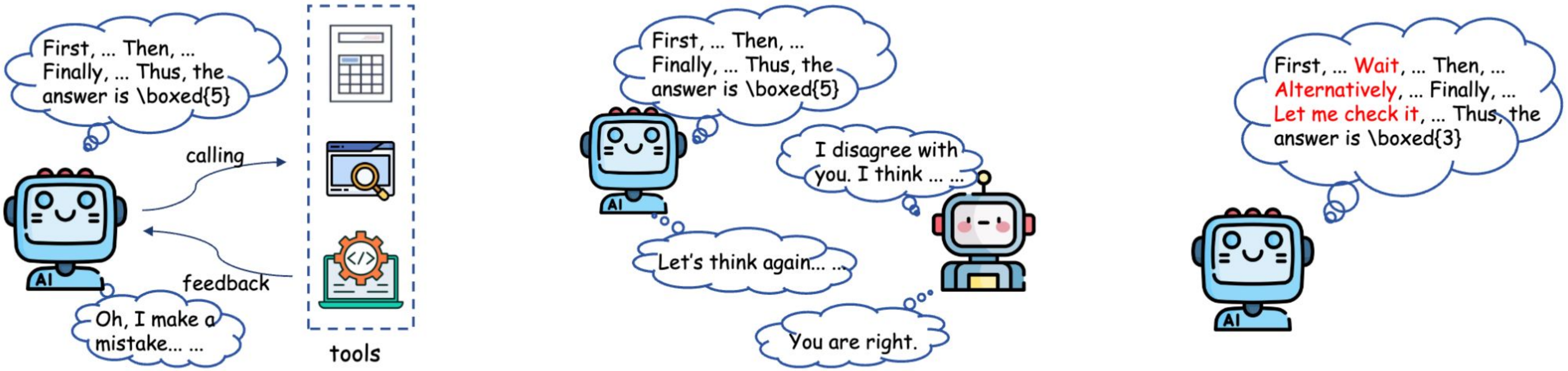
- For on-device LLMs, ideally what we want is a **"Small Reasoning Model"** (SLM)

We can “trade” training-time compute with test-time compute



The impact of compute time on OpenAI’s o1 model accuracy during training (left) and test time (right), highlighting the effectiveness of Test Time Compute. Credit: OpenAI.

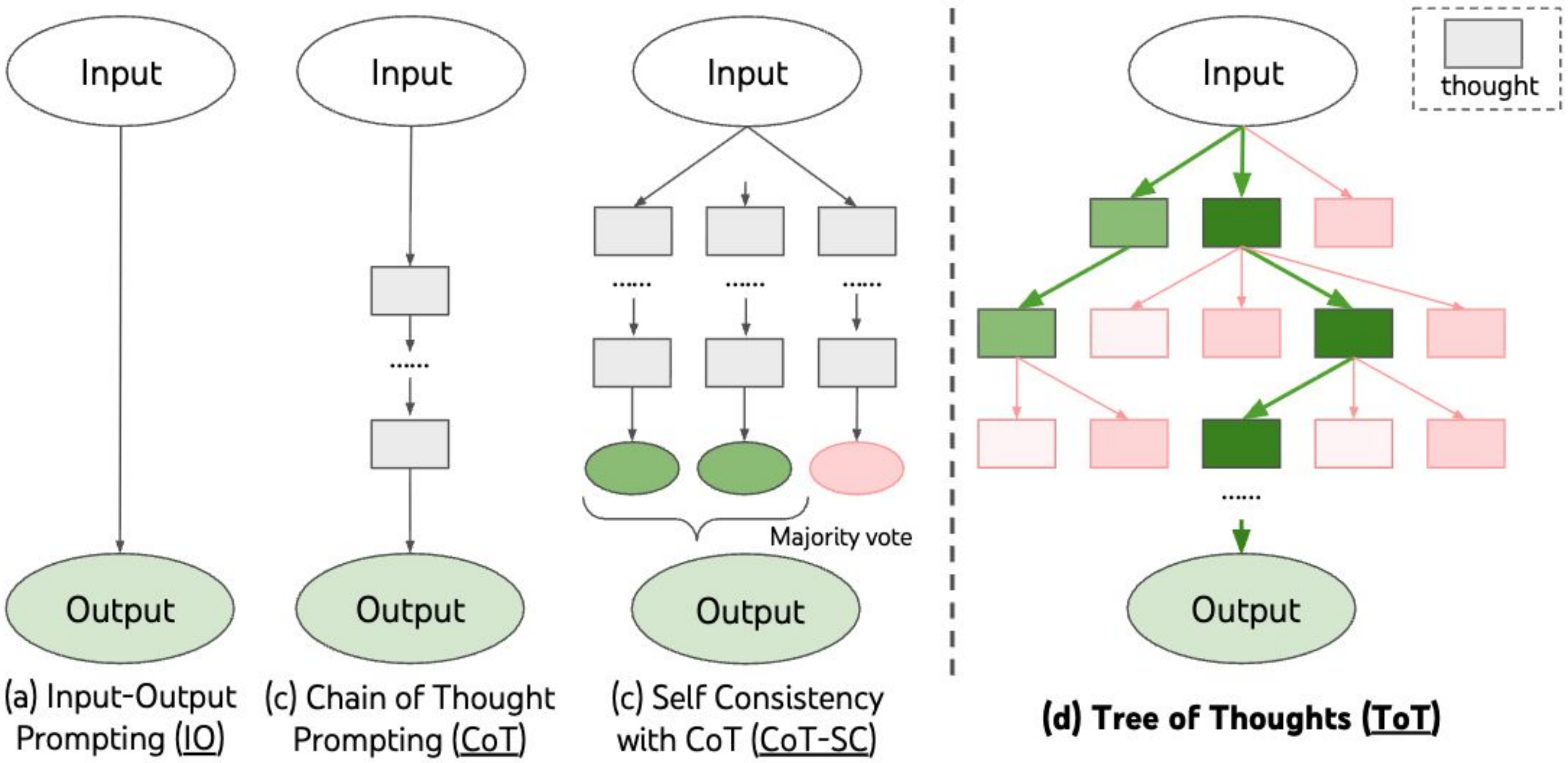
Other types of test-time compute



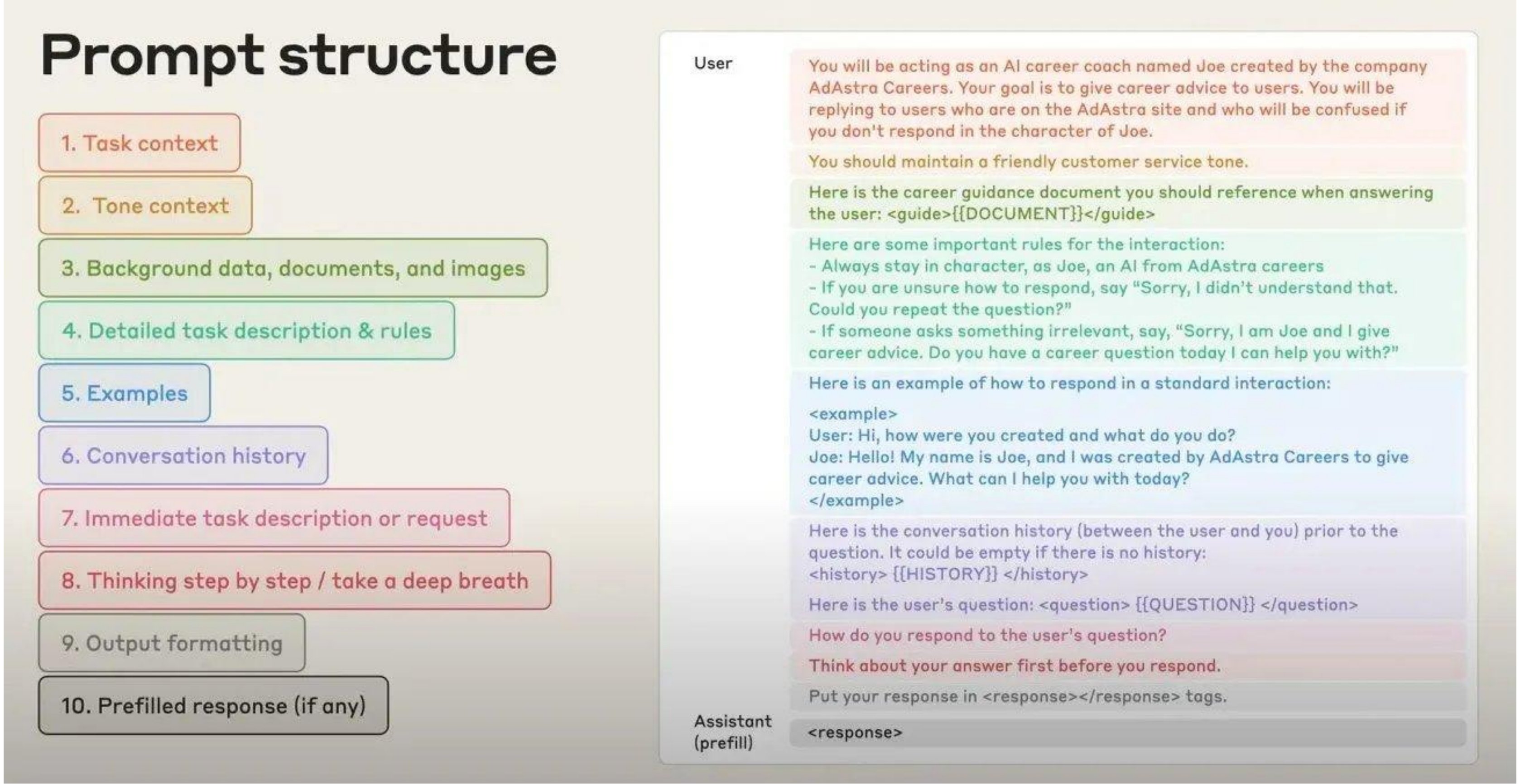
(a) Tool checking (b) External model evaluation (c) Self-critique

Ji, Yixin, Juntao Li, Hai Ye, Kaixin Wu, Kai Yao, Jia Xu, Linjian Mo, and Min Zhang. "Test-time compute: from system-1 thinking to system-2 thinking." arXiv preprint arXiv:2501.02497 (2025).

Chain of Thoughts / Self-Consistency / Tree of Thoughts



Anthropic recommended prompt structure



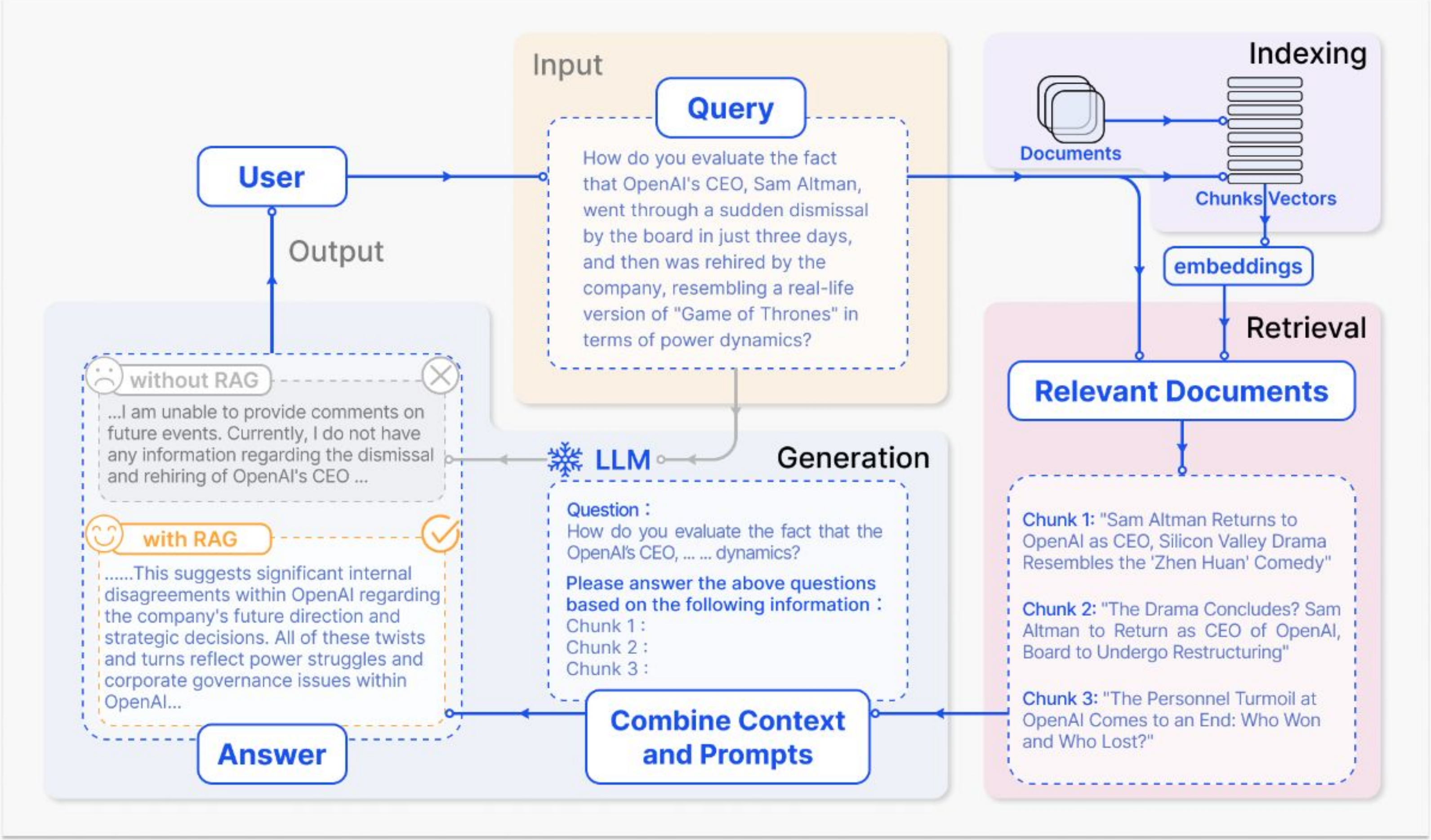
Retrieval-Augmented Generation (RAG)

- From the LLM's perspective, it is impossible to know, for example, an updated documentation for an API, private company documents / policy etc

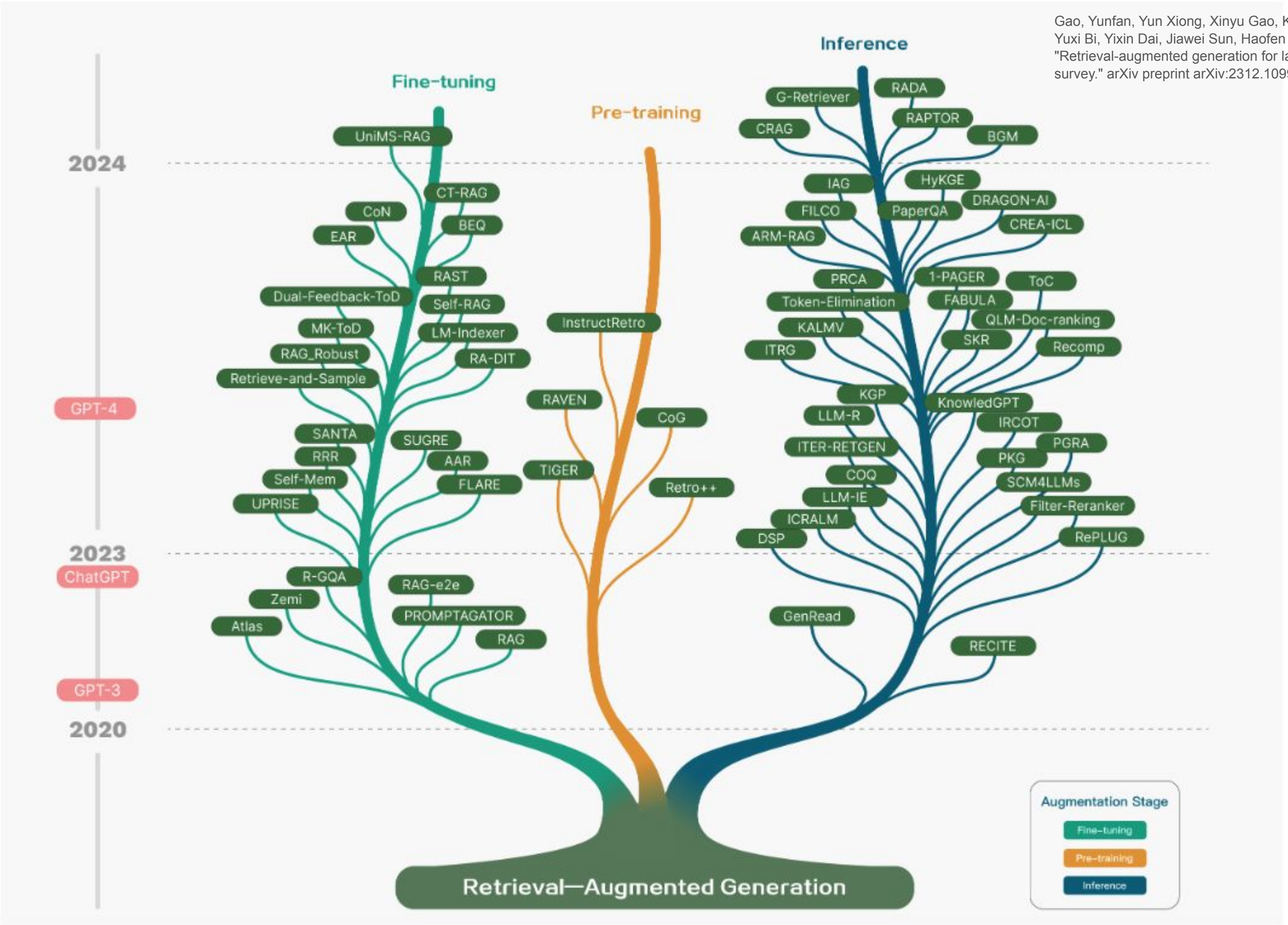
- Why should we expect it to know this information?

- Open-book and closed-book questions.

- Hope to reduce hallucinations



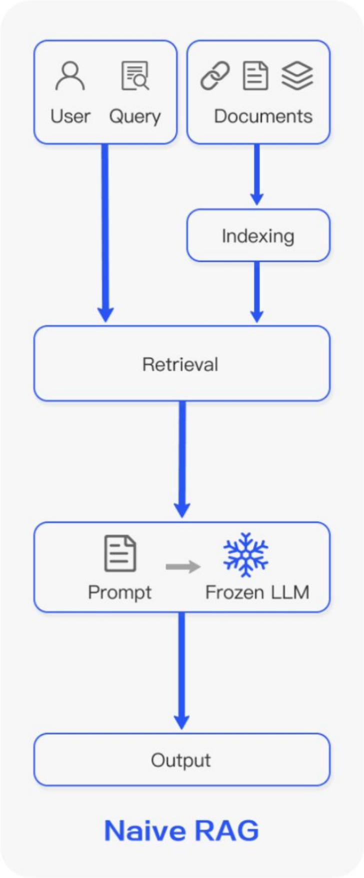
Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 2, no. 1 (2023).



Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 2, no. 1 (2023).

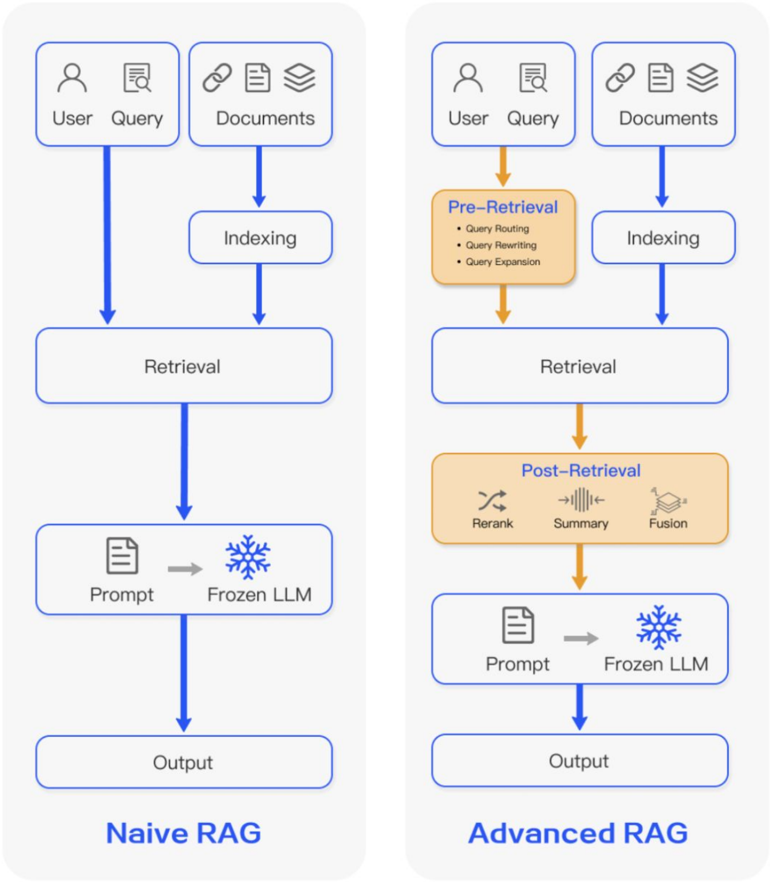
“Agentic” RAG

Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 2, no. 1 (2023).



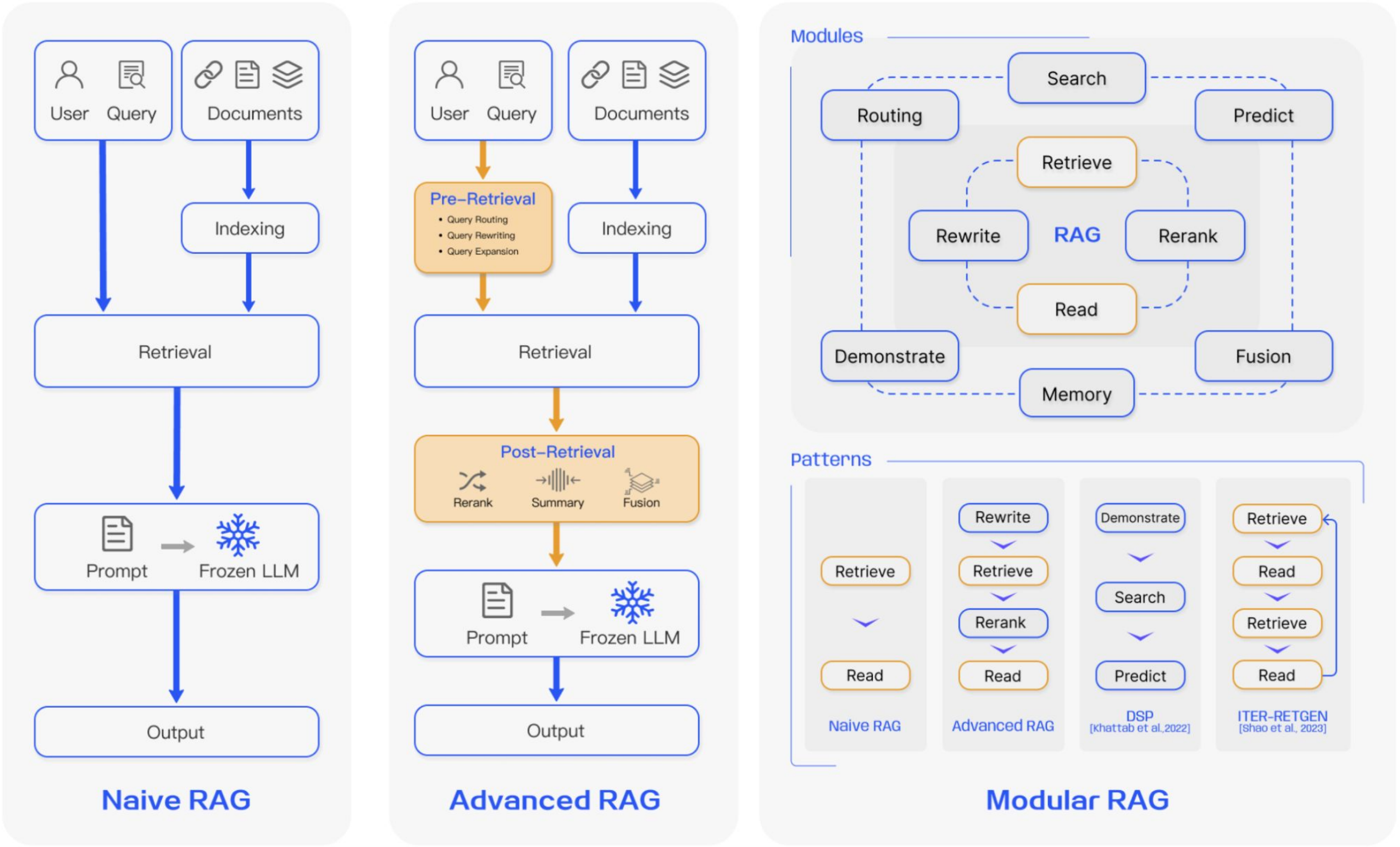
“Agentic” RAG

Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 2, no. 1 (2023).



"Agentic" RAG

Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 2, no. 1 (2023).



Tool calling

- Sometimes the model cannot perform a task by itself
- Needs to do some external computation

- For example, LLMs are notoriously bad at doing arithmetic
  - Partially due to tokenization

- But there is no need for LLMs to do arithmetic, they can just call a calculator if needed!

Tool calling - Example

<im_start>llm

<start>

<im_start>user

What

is

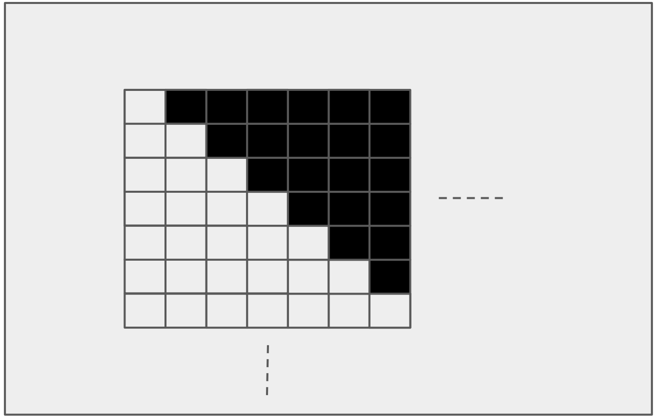
2

+

2

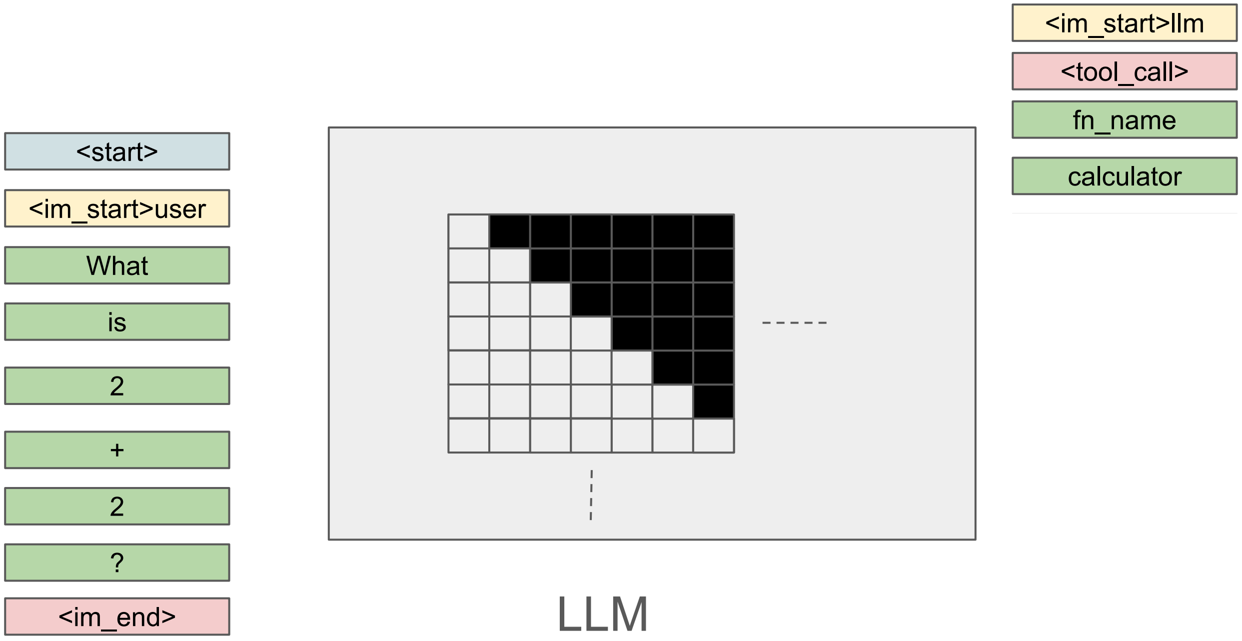
?

<im_end>

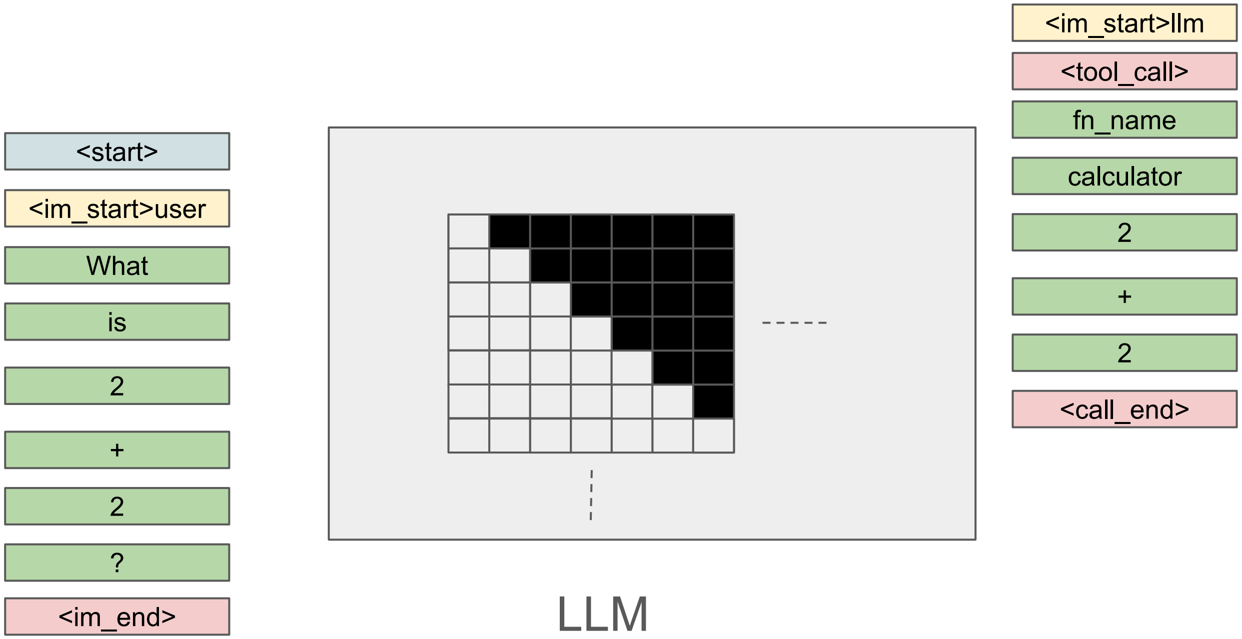


LLM

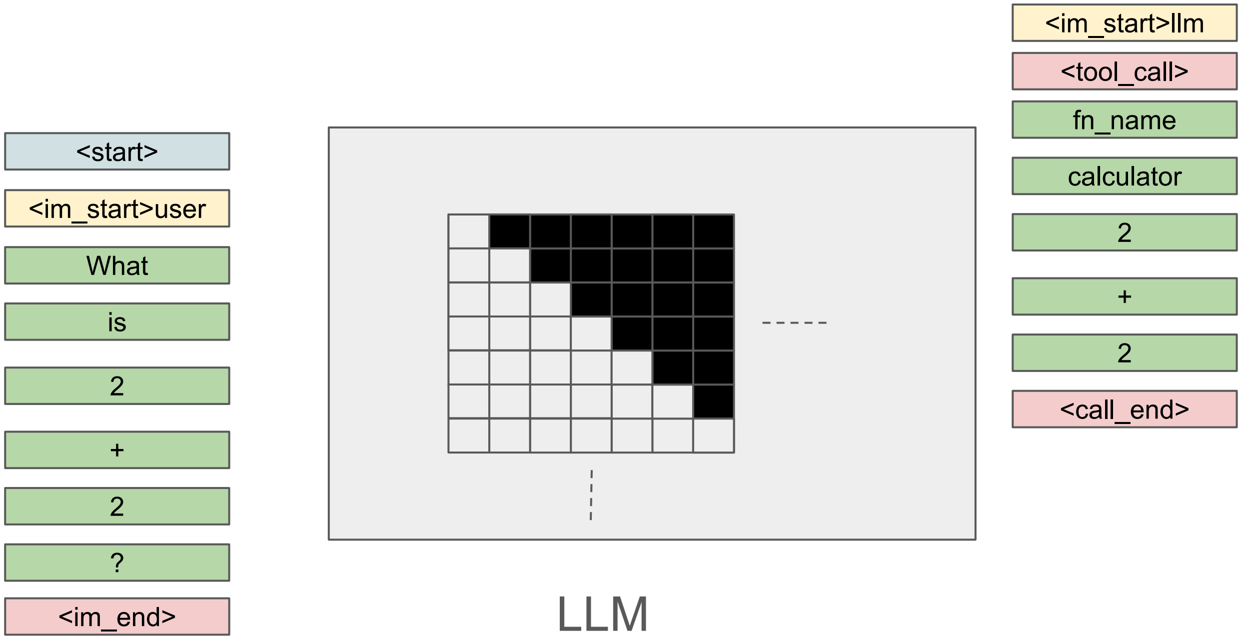
Tool calling - Example



Tool calling - Example

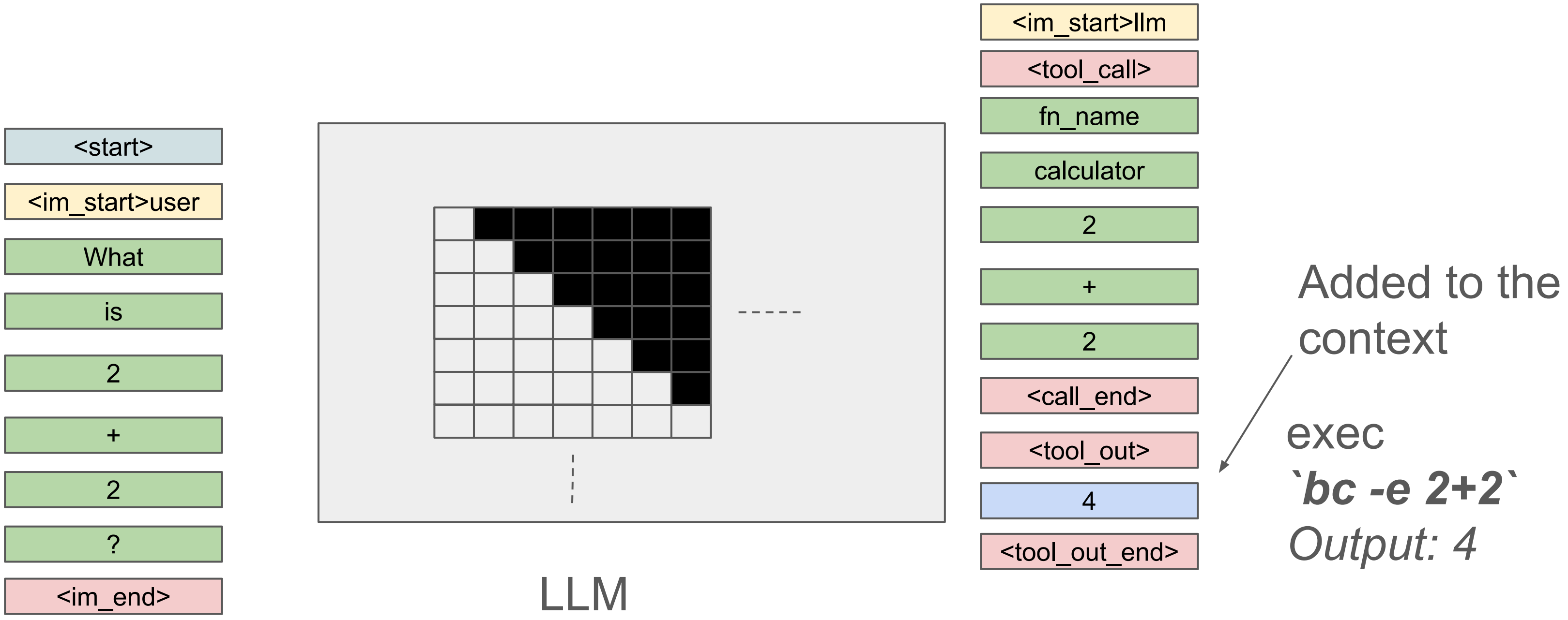


Tool calling - Example



exec
`bc -e 2+2`
Output: 4

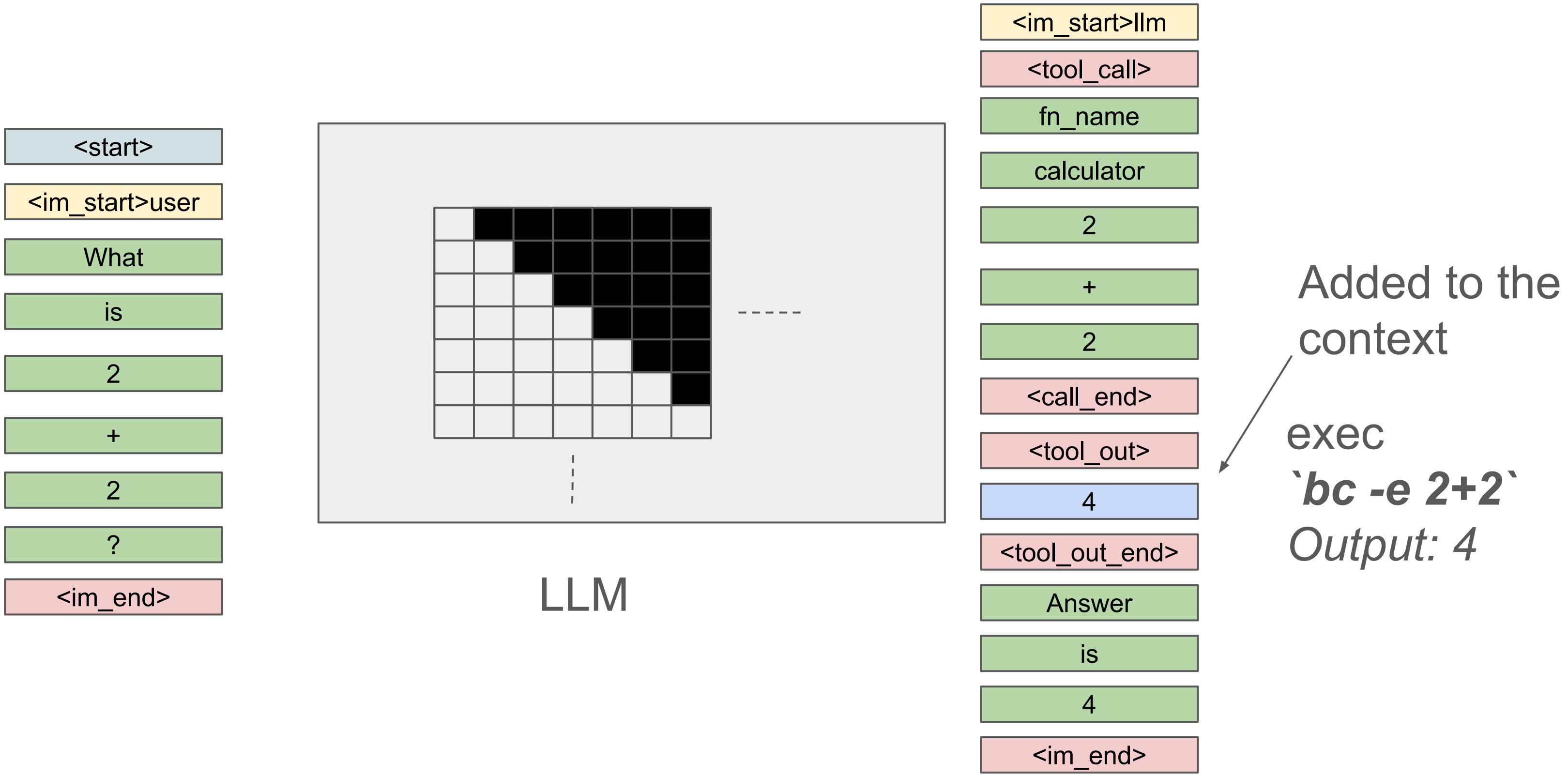
Tool calling - Example



Tool calling - Example

<tool_call> <call_end>
<tool_out> <tool_out_end>

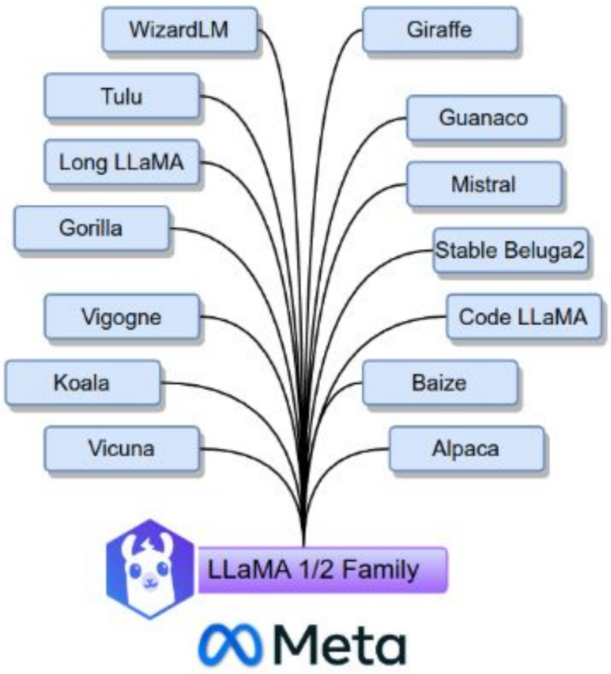
“Special tokens”



Trends for LLMs for edge

- Move towards **open-source models** as opposed to proprietary APIs
  - Privacy preserving
  - E.g., Phi, LLaMa, **Gemma**

- Move towards **domain-specific models** as opposed to general-purpose ones
  - Fine-tune on specific data
  - Reason more, store less knowledge
  - Model ownership: you own the weights!



Small Language Models are the Future of Agentic AI

Peter Belcak[1] **Greg Heinrich**[1] **Shizhe Diao**[1] **Yonggan Fu**[1] **Xin Dong**[1]
Saurav Muralidharan[1] **Yingyan Celine Lin**[1,2] **Pavlo Molchanov**[1]
[1]NVIDIA Research [2]Georgia Institute of Technology
`agents@nvidia.com`

Abstract

Large language models (LLMs) are often praised for exhibiting near-human performance on a wide range of tasks and valued for their ability to hold a general conversation. The rise of agentic AI systems is, however, ushering in a mass of applications in which language models perform a small number of specialized tasks repetitively and with little variation.

Here we lay out the position that small language models (SLMs) are *sufficiently powerful, inherently more suitable, and necessarily more economical for many invocations in agentic systems, and are therefore the future of agentic AI*. Our argumentation is grounded in the current level of capabilities exhibited by SLMs, the common architectures of agentic systems, and the economy of LM deployment. We further argue that in situations where general-purpose conversational abilities

Applications for on-device LLMs

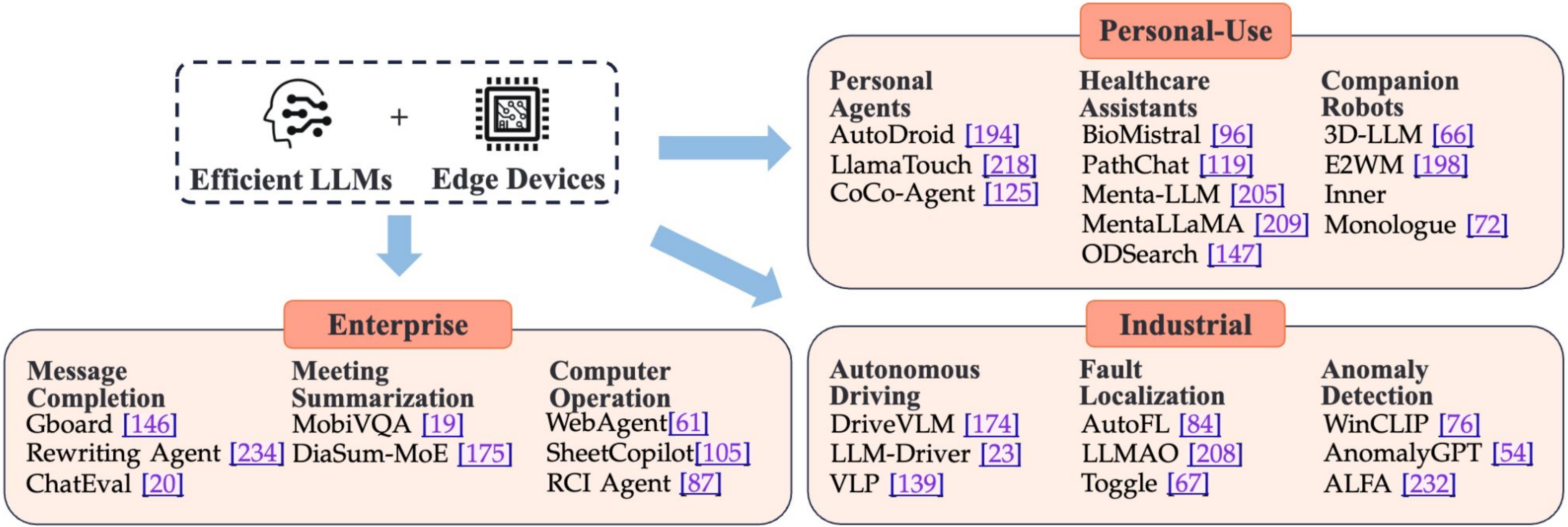


Fig. 11. Illustrations of on-device applications for LLMs.

Zheng, Yue, et al. "A review on edge large language models: Design, execution, and applications." ACM Computing Surveys 57.8 (2025): 1-35.

Llama.cpp

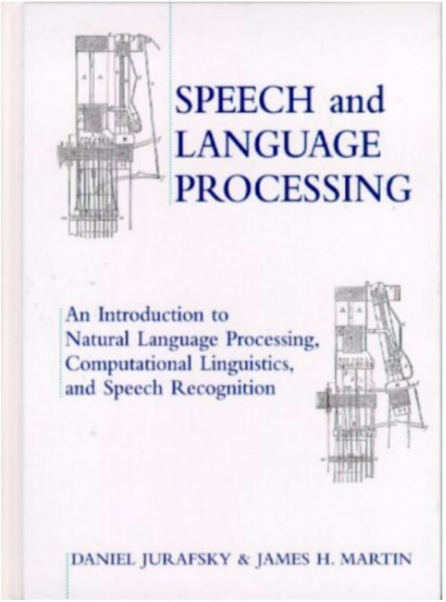
- State-of-the-art performance

- Minimal setup

- Compatible with the majority of .gguf quantized models

A Perspective on the Field

NLP circa 1999



1st edition

Summary of Contents

words

↓

syntax (parsing)

↓

semantics

↓

pragmatics

Comp. linguistics

AI

NLP today has transform(er)ed

Speech and Language Processing

An Introduction to Natural Language Processing,
Computational Linguistics, and Speech Recognition

Third Edition draft

Daniel Jurafsky
Stanford University

James H. Martin
University of Colorado at Boulder

Draft of February 3, 2024. Comments and typos welcome!

3rd edition

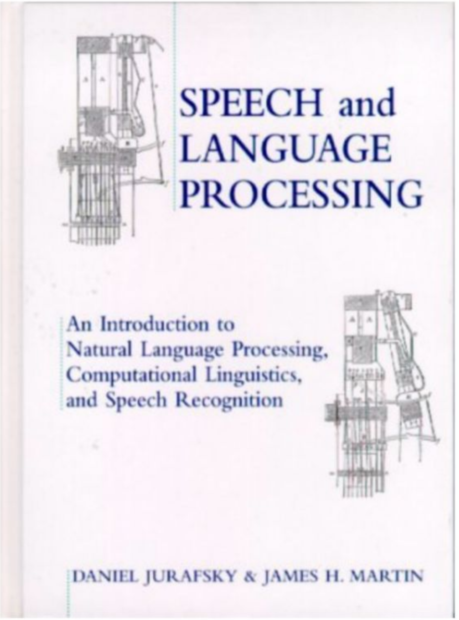
Summary of Contents

Neural networks, LMs, transformers, and LLMs

Misc. other stuff (including parsing)

The book is now 40% shorter

NLP circa 1999



1st edition

Summary of Contents

Fake Tasks

Real Tasks

(sub)words

syntax (parsing)

semantics

pragmatics

LLMs

Workshop Overview

1. **Part 0: Introduction to the Team & Rust for Edge AI**
2. **Part I: Lecture on Computer Vision**
   a. Main problems in Computer Vision
   b. What exactly is a neural network? (CNNs / Transformers)
   c. What exactly is an image embedding?
   d. Computer vision on the Edge
3. **Hands-On I: Air-gapped Face recognition on the Pi**
4. **Part II: Lecture on Natural Language Processing**
   a. A bit of history & development of modern LLMs
   b. How does an LLM work? Tokenizers, pretraining, post-training
   c. Context Engineering: Tool calling, RAG
   d. Libraries: tokenizers-rs, llama.cpp
5. **Hands-On II: Chat with a LLM on Pi**

Workshop Overview

1. **Part 0: Introduction to the Team & Rust for Edge AI**
2. **Part I: Lecture on Computer Vision**
    a. Main problems in Computer Vision
    b. What exactly is a neural network? (CNNs / Transformers)
    c. What exactly is an image embedding?
    d. Computer vision on the Edge
3. **Hands-On I: Air-gapped Face recognition on the Pi**
4. **Part II: Lecture on Natural Language Processing**
    a. A bit of history & development of modern LLMs
    b. How does an LLM work? Tokenizers, pretraining, post-training
    c. Context Engineering: Tool calling, RAG
    d. Libraries: tokenizers-rs, llama.cpp
5. **Hands-On II: Chat with a LLM on Pi**

Workshop Overview

1. **Part 0: Introduction to the Team & Rust for Edge AI**
2. **Part I: Lecture on Computer Vision**
   a. Main problems in Computer Vision
   b. What exactly is a neural network? (CNNs / Transformers)
   c. What exactly is an image embedding?
   d. Computer vision on the Edge
3. **Hands-On I: Air-gapped Face recognition on the Pi**
4. **Part II: Lecture on Natural Language Processing**
   a. A bit of history & development of modern LLMs
   b. How does an LLM work? Tokenizers, pretraining, post-training
   c. Context Engineering: Tool calling, RAG
   d. Libraries: tokenizers-rs, llama.cpp
5. **Hands-On II: Chat with a LLM on Pi**

Hands-on Overview
Chat with an LLM on the PI

- Deploy Gemma 3 using Llama.cpp
- Simple Chat Request
- RAG
- Structured Outputs
- Tool Calling

https://github.com/Wyliodrin/edge-ai-chat-with-llm