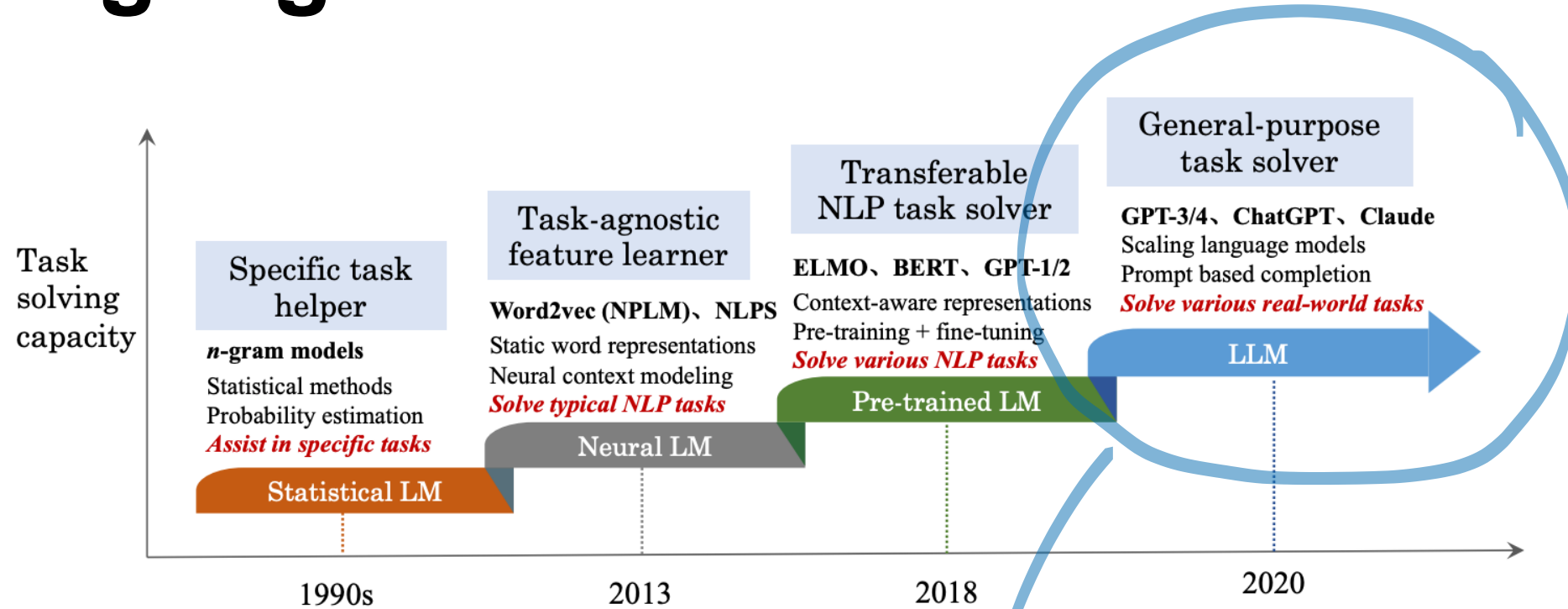


Building and Fine-tuning GenAI Models

SOEN 691: Generative Artificial Intelligence
for Software Engineering

Diego Elias Costa

Language Models



[Source: A Survey of Large Language Models](#)

We will focus on LLMs

Building and Fine-tuning ~~GenAI~~ Models

Large
Language

This is a Road Map class

Explore the reference materials if you want to know the concepts in detail!



Building a Large Language Model

How much does it cost?

- Training the smallest Llama 7b took 180,000 GPU hours

		Time (GPU hours)	Power Consumption (W)	Carbon Emitted (tCO ₂ eq)
LLAMA 2	7B	184320	400	31.22
	13B	368640	400	62.44
	34B	1038336	350	153.90
	70B	1720320	400	291.42
Total		3311616		539.00

- Renting:
 - Nvidia A100: \$1-2 per GPU per hour -> whereabouts of **\$250,000** dollars

[Source: Llama 2: Open Foundation and Fine-Tuned Chat Models](#)

Building a Large Language Model

Theoretically...

A Survey of Large Language Models

Wayne Xin Zhao, Kun Zhou*, Junyi Li*, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie and Ji-Rong Wen

Abstract—Ever since the Turing Test was proposed in the 1950s, humans have explored the mastering of language intelligence by machine. Language is essentially a complex, intricate system of human expressions governed by grammatical rules. It poses a significant challenge to develop capable artificial intelligence (AI) algorithms for comprehending and grasping a language. As a major approach, *language modeling* has been widely studied for language understanding and generation in the past two decades, evolving from statistical language models to neural language models. Recently, pre-trained language models (PLMs) have been proposed by pre-training Transformer models over large-scale corpora, showing strong capabilities in solving various natural language processing (NLP) tasks. Since the researchers have found that model scaling can lead to an improved model capacity, they further investigate the scaling effect by increasing the parameter scale to an even larger size. Interestingly, when the parameter scale exceeds a certain level, these enlarged language models not only achieve a significant performance improvement, but also exhibit some special abilities (*e.g.*, in-context learning) that are not present in small-scale language models (*e.g.*, BERT). To discriminate the language models in different parameter scales, the research community has coined the term *large language models (LLM)* for the PLMs of significant size (*e.g.*, containing tens or hundreds of billions of parameters). Recently, the research on LLMs has been largely advanced by both academia and industry, and a remarkable progress is the launch of ChatGPT (a powerful AI chatbot developed based on LLMs), which has attracted widespread attention from society. The technical evolution of LLMs has been making an important impact on the entire AI community, which would revolutionize the way how we develop and use AI algorithms. Considering this rapid technical progress, in this survey, we review the recent advances of LLMs by introducing the background, key findings, and mainstream techniques. In particular, we focus on four major aspects of LLMs, namely pre-training, adaptation tuning, utilization, and capacity evaluation. Furthermore, we also summarize the available resources for developing LLMs and discuss the remaining issues for future directions. This survey provides an up-to-date review of the literature on LLMs, which can be a useful resource for both researchers and engineers.

Index Terms—Large Language Models; Emergent Abilities; Adaptation Tuning; Utilization; Alignment; Capacity Evaluation

[Source: A Survey of Large Language Models](#)

Building a machine learning model

Here are the 4 key steps to build a traditional ML model:

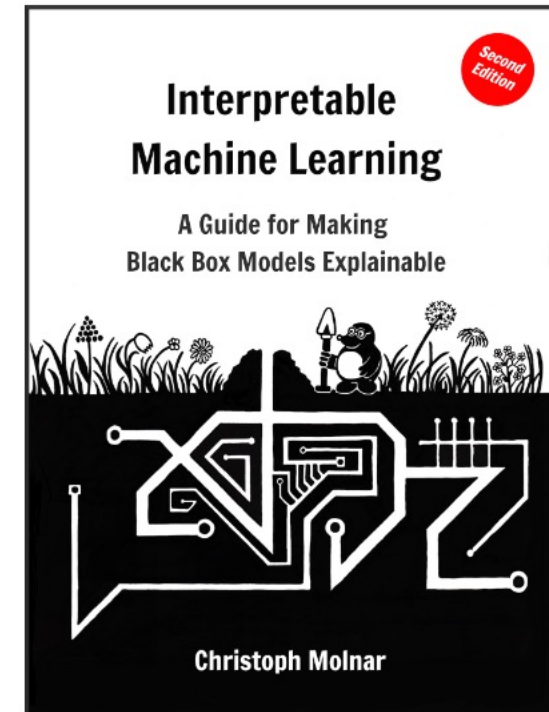
1. Curating the Data
2. Choosing the appropriate model/algorithm
3. Training the model
4. Evaluating the model

What to know more?

Machine Learning

Book freely available

- Covers the basics of classifiers and Regressions
- Emphasis on interpretability
 - How do ML models work?
 - Techniques to explain and interpret ML models



[Source: Interpretable Machine Learning](#)

Want to know more?

Engineering AI-Based Systems

- Course focused on Engineering AI-Based Systems
 - Requirements
 - Architecture and Design
 - Implementation
 - Testing
 - Operations



SOEN 691: Engineering AI-Based Software Systems

[Source: Engineering AI-Based Software Systems](#)

Building large language models

Here are the 4 key steps to build a traditional ML model:

1. Curating/Processing **large amounts** of Data
2. Choosing the appropriate **transformer**
3. Training the model **at scale**
4. Evaluating the **basic** model
5. **Fine-tuning the model to specific cases**

[Source: How to Build an LLM from Scratch | An Overview](#)

Step 1. Data Curation/Processing

Data quality is paramount for the quality of your model

- Garbage in, Garbage out

However, **data quantity** seems to be playing an even bigger role

[Source: Llama 2: Open Foundation and Fine-Tuned Chat Models](#)

Step 1. Data Curation/Processing

Where do we get all these data?

- The Internet
- Public datasets
 - Common Crawl
 - The Pile
 - Hugging Face datasets
- Private data sources
- Use an LLM to generate your dataset
 - Alpaca

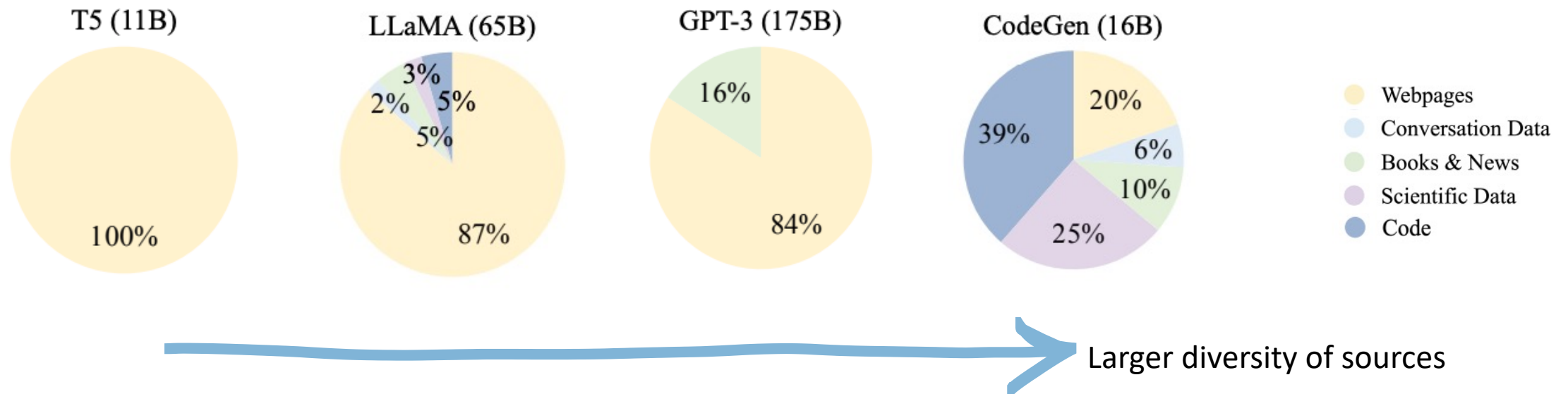
Publicly available datasets

📄 C4 (800G, 2019), 📄 OpenWebText (38G, 2023), 📄 Wikipedia (21G, 2023)
💬 the Pile - StackExchange (41G, 2020)
📖 BookCorpus (5G, 2015), 📖 Gutenberg (-, 2021), 📖 CC-Stories-R (31G, 2019),
🔬 the Pile - ArXiv (72G, 2020), 🔬 the Pile - PubMed Abstracts (25G, 2020)
📄 BigQuery (-, 2023), the Pile - GitHub (61G, 2020)

[Source: A Survey of Large Language Models`](#)

Step 1. Data Curation/Processing

Data diversity is key to generalizability



[Source: A Survey of Large Language Models`](#)

Step 1. Data Curation/Processing

How to prepare the data?

- Filter data based on quality
 - Using heuristics or models to put higher weight in better quality text
- De-duplicate the data
 - Reduce the bias toward common text
- Privacy redaction
 - Removal of sensitive and confidential information



Step 1. Data Curation/Processing

Large Language Models do not understand "pure text"

Common processing techniques

1. Punctuation padding
2. Tokenization
3. Text Vectorization
4. Casual masking

Hi, my name is Pierre!

Hi , my name is Pierre !

[12 56 15 56 33 49 2]

[12 56 15 56 33 49 2]

[12 56 15 56 33 49 2]

[12 56 15 56 33 49 2]

[12 56 15 56 33 49 2]

[12 56 15 56 33 49 2]

[12 56 15 56 33 49 2]

Step 2. Model Architecture

Transformers

- Neural Networks that use the **attention** mechanism

Dependency of words based on position and context

Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works.

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

[Source: Attention is All You Need](#)

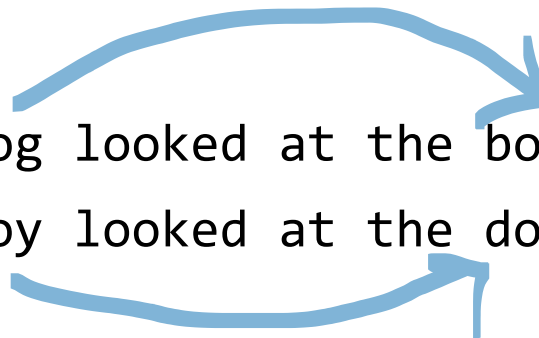
The Attention Mechanism

Learn to pay attention to some key words, depending on the context

The pink elephant tried to get into a car but it was too _____

Positional encoding: the order of the words matter

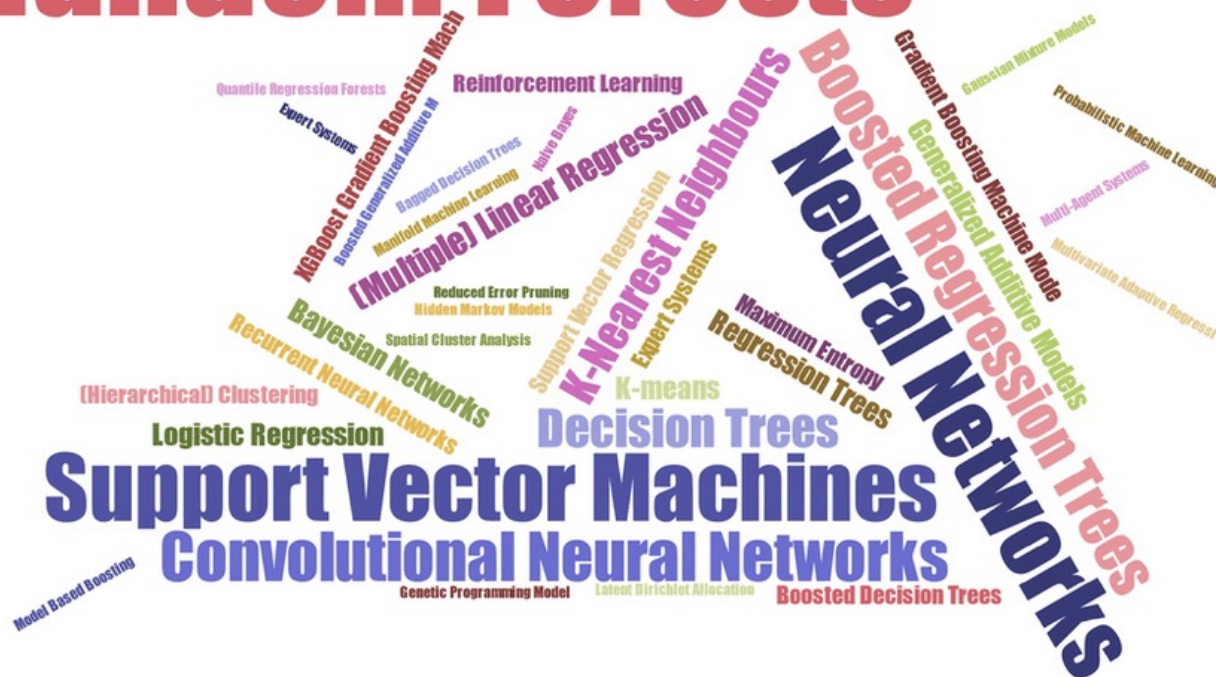
The dog looked at the boy and _____
The boy looked at the dog and _____



Why don't we have the same diversity?

Machine Learning algorithms

Random Forests



[Source: Machine learning in landscape ecological analysis: a review of recent approaches](#)

LLM algorithms

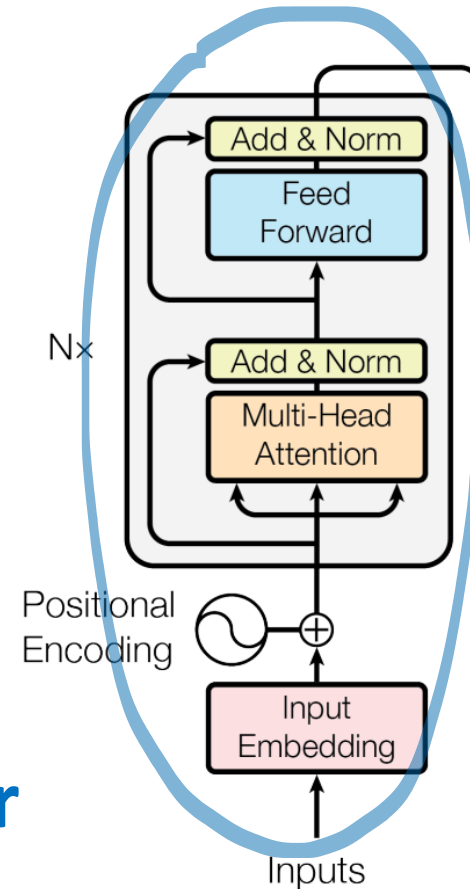


Step 2. Model Architecture

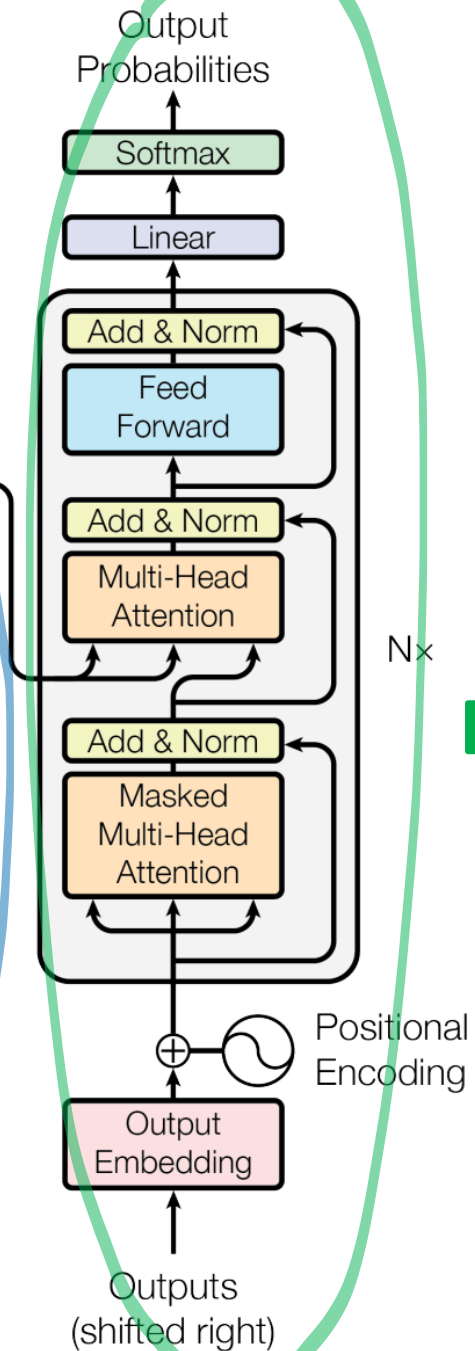
Choose between 3 types of Transformers

- Encoder-only
- Encoder-Decoder
- Decoder-only

Encoder



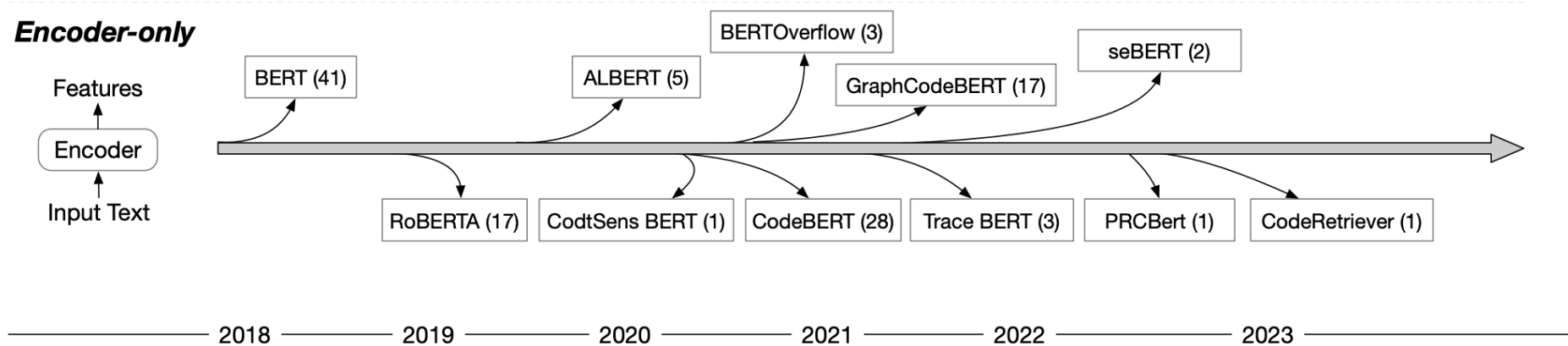
Decoder



Encoder-only Architecture

Good for tasks requiring a nuanced understanding of the entire sentence or code snippet.

- Code review, bug report understanding, and named entity recognition

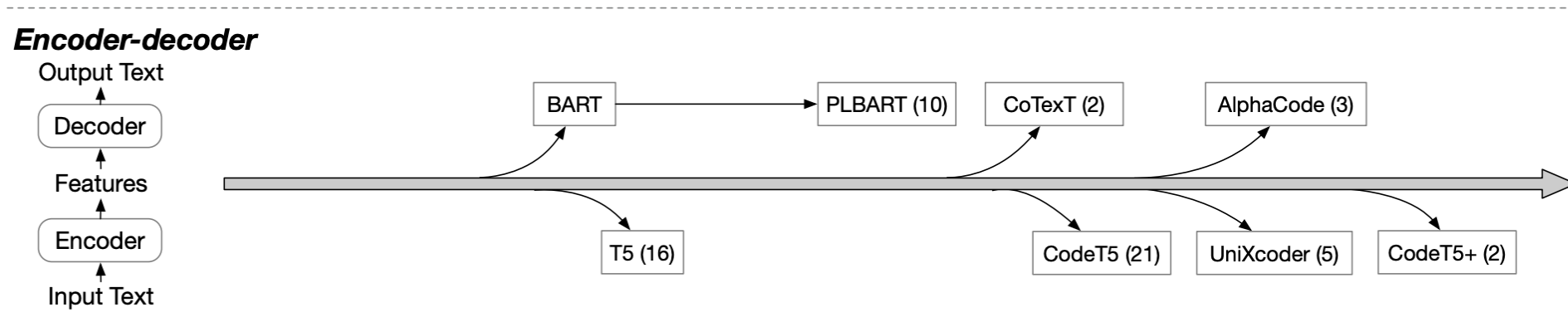


[Source: Large Language Models for Software Engineering: A Systematic Literature Review](#)

Encoder-decoder Architecture

Good for translation or summarization tasks.

- Code summarization, (programming?) language translation

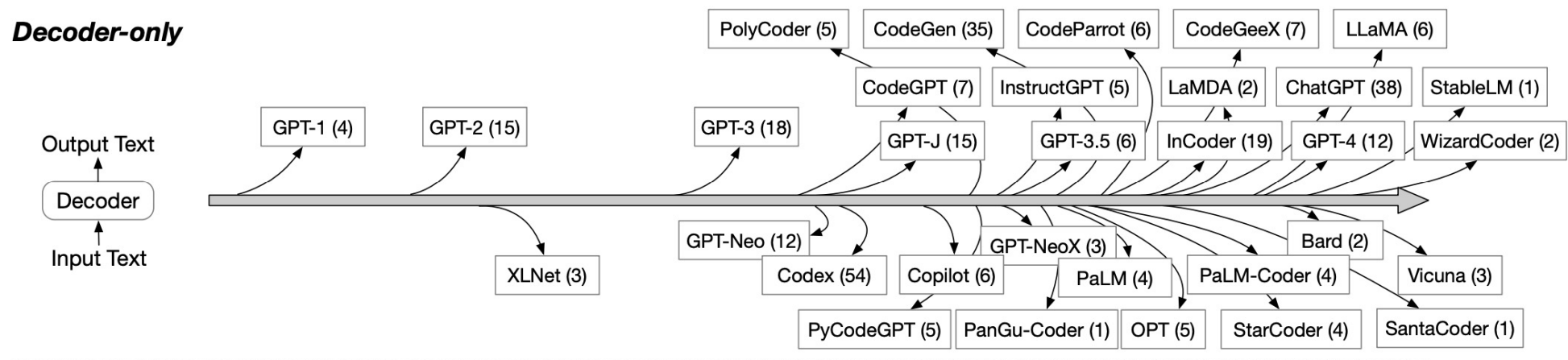


[Source: Large Language Models for Software Engineering: A Systematic Literature Review](#)

Decoder-only Architecture

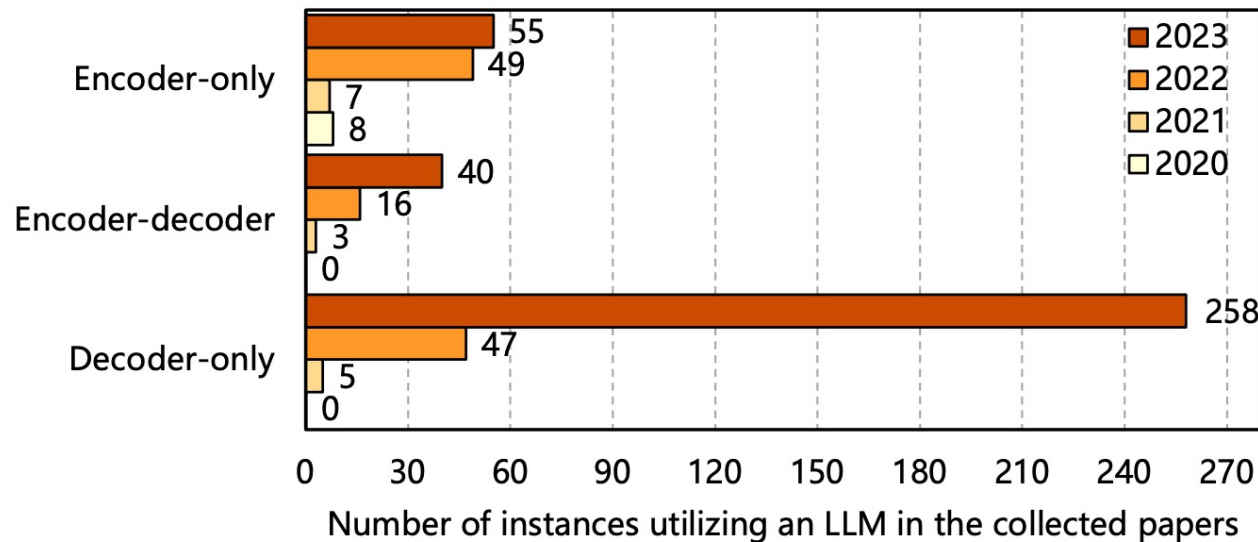
Good for generative tasks

- Basically any SE tasks that requires generation



[Source: Large Language Models for Software Engineering: A Systematic Literature Review](#)

Architectures of LLMs for SE



Decoder-only architecture (e.g., GPT, Llama) are the most popular architectures in SE research

Fig. 5. Trends in the application of LLMs with different architectures in SE tasks over time.

[Source: Large Language Models for Software Engineering: A Systematic Literature Review](#)

Step 2. Model Architecture

Other design choices and hyper-parameters to tune:

- Activation functions, Layer normalization, Position Embeddings

Model Size

Rule of Thumb:
1 parameter for
~20 tokens

Table 3 | **Estimated optimal training FLOPs and training tokens for various model sizes.** For various model sizes, we show the projections from Approach 1 of how many FLOPs and training tokens would be needed to train compute-optimal models. The estimates for Approach 2 & 3 are similar (shown in [Section D.3](#))

Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

Step 3. Training at Scale

- Self-supervision
 - We can train large models using huge amounts of data, only if the label is already part of the input
- Efficiency is key when training billion-parameter models
 - 3D parallelism (data parallelism, pipeline parallelism, tensor parallelism)
 - Mixed-precision training
 - Use 16-bit floating numbers instead of 32-bit to reduce memory usage and communication overhead

[Source: A Survey of Large Language Models](#)

Step 4. Evaluation

How to assess the quality of the trained LLM?

- Benchmarks ([Open LLM Leaderboard](#))
- Multiple-choice Tasks
 - ARC,
 - [Hellaswag](#),
 - MMLU
- Open-ended Tasks
 - [TruthfulQA](#)

Example: Hellaswag

- Can a Machine really finish your sentence?

How to catch dragonflies. Use a long-handled aerial net with a wide opening. Select an aerial net that is 18 inches (46 cm) in diameter or larger. Look for one with a nice long handle.

a) Loop 1 piece of ribbon over the handle. Place the hose or hose on your net and tie the string securely.


b) Reach up into the net with your feet. Move your body and head forward when you lift up your feet.

c) If possible, choose a dark-colored net over a light one. Darker nets are more difficult for dragonflies to see, making the net more difficult to avoid.

d) If it's not strong enough for you to handle, use a hand held net with one end shorter than the other. The net should have holes in the bottom of the net.

Building large language models

Here are the 4 key steps to build a traditional ML model:

1. Curating **large amounts** of Data
 2. Choosing the appropriate **transformer**
 3. Training the model **at scale**
 4. Evaluating the model
 5. **Fine-tuning the model to specific cases**
- 
- Base LLM
Foundational Model

DEMO

SOEN 691: Building your own GPT

In this notebook, we'll walk through the steps required to train your own GPT model on the wine review dataset.

- The code is adapted from the excellent [GPT tutorial](#) created by Apoorv Nandan available on the Keras website.
- The code is also adapted from the notebook provided in the [Generative Deep Learning](#)

Setting up the environment for the code, ensuring that necessary libraries and modules are loaded and ready for use in the subsequent cells.

+ Code

+ Markdown

[34]:

```
%load_ext autoreload
%autoreload 2
import numpy as np
import json
import re
import string
from IPython.display import display, HTML

import tensorflow as tf
from tensorflow.keras import layers, models, losses, callbacks
```

The autoreload extension is already loaded. To reload it, use:

<https://www.kaggle.com/code/diegoeliascosta/soen691-building-your-own-gpt>

From Base Models to SE-related Solutions

Large Language Models for Software Engineering: A Systematic Literature Review

XINYI HOU*, Huazhong University of Science and Technology, China
YANJIE ZHAO*, Monash University, Australia
YUE LIU, Monash University, Australia
ZHOU YANG, Singapore Management University, Singapore
KAILONG WANG, Huazhong University of Science and Technology, China
LI LI, Beihang University, China
XIAPU LUO, The Hong Kong Polytechnic University, China
DAVID LO, Singapore Management University, Singapore
JOHN GRUNDY, Monash University, Australia
HAOYU WANG[†], Huazhong University of Science and Technology, China

Large Language Models (LLMs) have significantly impacted numerous domains, including Software Engineering (SE). Many recent publications have explored LLMs applied to various SE tasks. Nevertheless, a comprehensive understanding of the application, effects, and possible limitations of LLMs on SE is still in its early stages. To bridge this gap, we conducted a systematic literature review on LLM4SE, with a particular focus on understanding how LLMs can be exploited to optimize processes and outcomes. We collect and analyze 229 research papers from 2017 to 2023 to answer four key research questions (RQs). In RQ1, we categorize different LLMs that have been employed in SE tasks, characterizing their distinctive features and uses. In RQ2, we analyze the methods used in data collection, preprocessing, and application highlighting the role of well-curated datasets for successful LLM for SE implementation. RQ3 investigates the strategies employed to optimize and evaluate the performance of LLMs in SE. Finally, RQ4 examines the specific SE tasks where LLMs have shown success to date, illustrating their practical contributions to the field. From the answers to these RQs, we discuss the current state-of-the-art and trends, identifying gaps in existing research, and flagging promising areas for future study.

[Source: Large Language Models for SE](#)

Fine-tuning in a nutshell

1. Define a SE-related task/problem where LLM has the potential to assist
2. Find a reasonably sized dataset for the model to fine-tune
3. Choose a pre-trained LLM
 - Open source or closed source + API
4. Choose the method for fine-tuning
5. Evaluate the fine-tuned model in the SE task

What SE Task is commonly selected?

SE Activity	SE Task	Total
Requirements engineering	Anaphoric ambiguity treatment (3)	11
	Requirements classification (3)	
	Requirement analysis and evaluation (2)	
Software design	GUI retrieval (1)	3
	Rapid prototyping (1)	
Software development	Code generation (62)	136
	Code completion (16)	
	Code summarization (10)	
	Code understanding (7)	
	Code search (5)	
	Program synthesis (5)	
	API recommendation (2)	
	API synthesis (2)	
	Code comment generation (2)	
	Code representation (2)	
	Method name generation (2)	
	Others (11)	
Software quality assurance	Test generation (8)	24
	Vulnerability detection (7)	
	Test automation (4)	
	Verification (2)	
Software maintenance	Program repair (23)	58
	Code review (6)	
	Debugging (4)	
	Bug report analysis (3)	
	Code clone detection (3)	
	Logging (2)	
	Bug prediction (1)	
	Bug triage (1)	
	Bug report replay (1)	
Software management	Effort estimation (1)	1

Most common SE Tasks

- Code generation (62)
- Program Repair (23)
- Code completion (16)
- Code summarization (10)

SE Related Datasets

Challenge:
Which ones are ****really****
open source?

Dominance of code-based and text-based datasets

Category	Data type	# Studies	Total
Code-based datasets	Source code	44	70
	Bugs	4	
	Patches	4	
	Code changes	4	
	Test suites/cases	3	
	Error code	3	
	Vulnerable source code	2	
	Bug-fix pairs	2	
	Labeled clone pairs	1	
	Buggy programs	1	
	Packages	1	
	Flaky test cases	1	

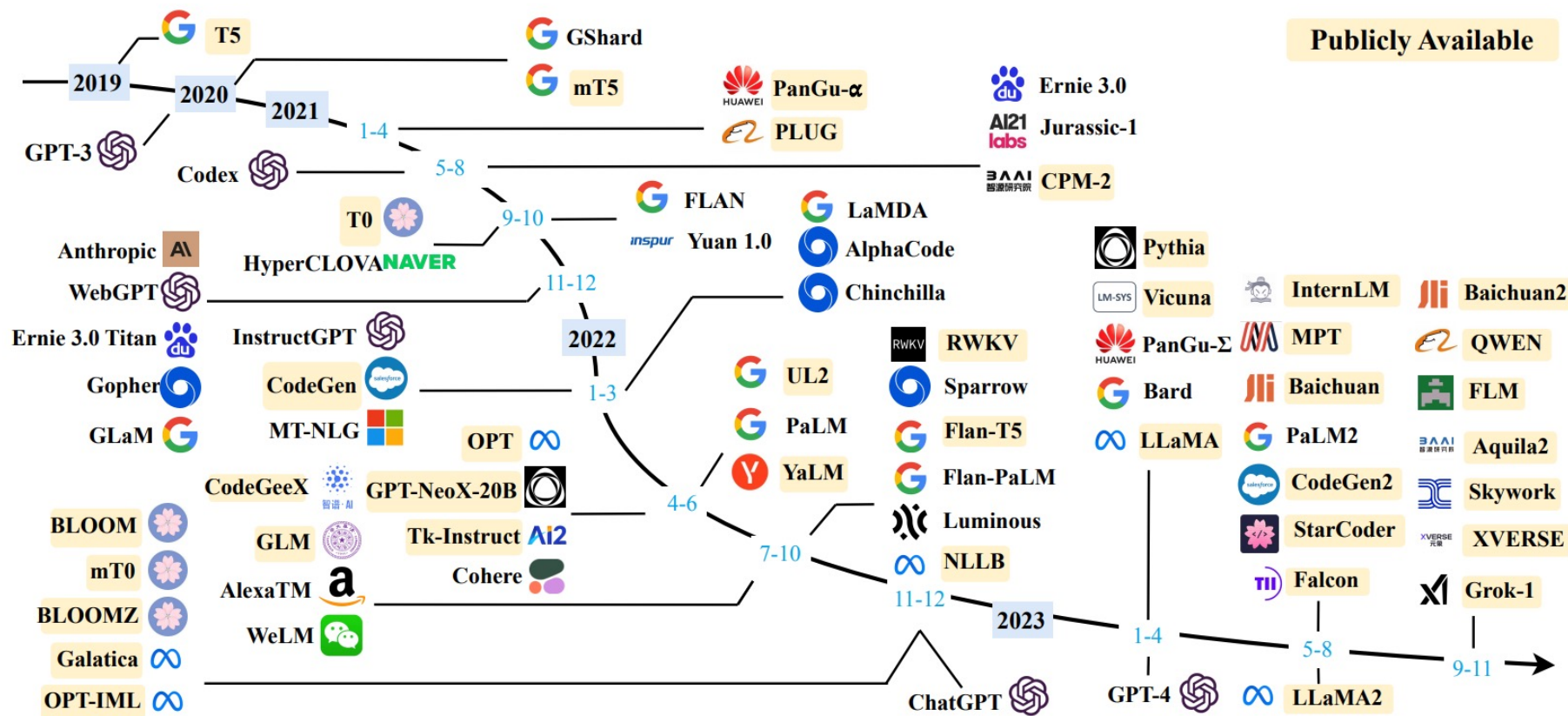
Category	Data type	# Studies	Total
Text-based datasets	Prompts	28	104
	Programming problems	14	
	SO (i.e., Stack Overflow) posts	9	
	Bug reports	9	
	Programming tasks (and solutions)	7	
	Requirements documentation	6	
	APIs/API documentation	5	
	Q&A pairs	5	
	Vulnerability descriptions	4	
	Bug reports with changesets	4	
	Methods	3	
	Code comments	2	

[Source: Large Language Models for SE](#)

There are plenty of base LLM models out there...

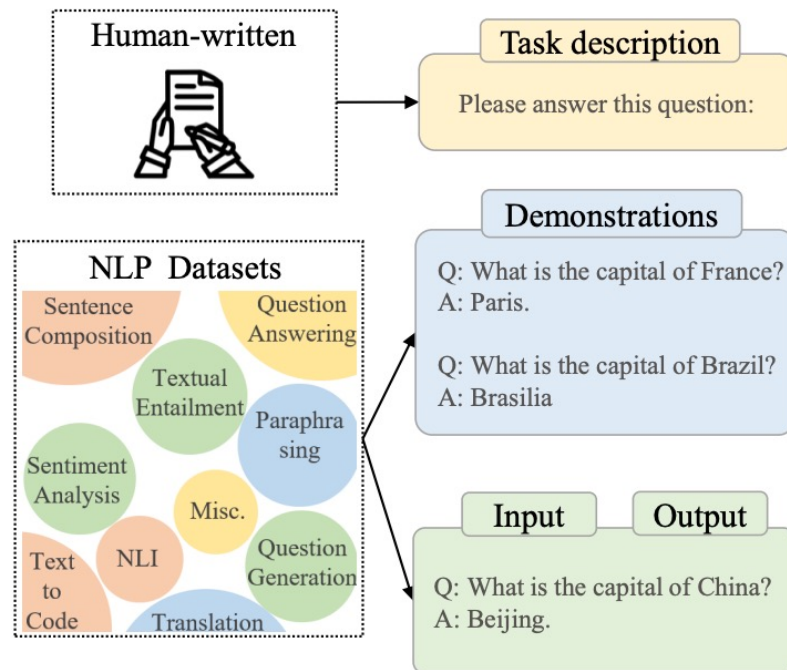
Challenge:

Which ones can be used in conventional hardware?



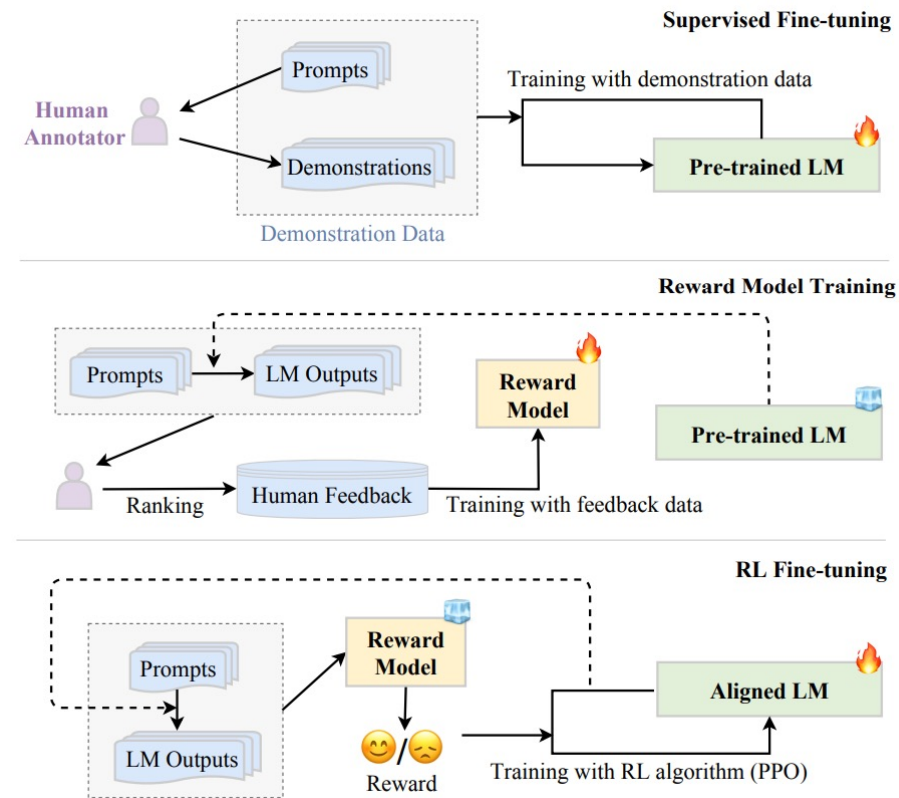
Fine-tuning strategies

Instruction tuning



(a) Formatting Task Datasets

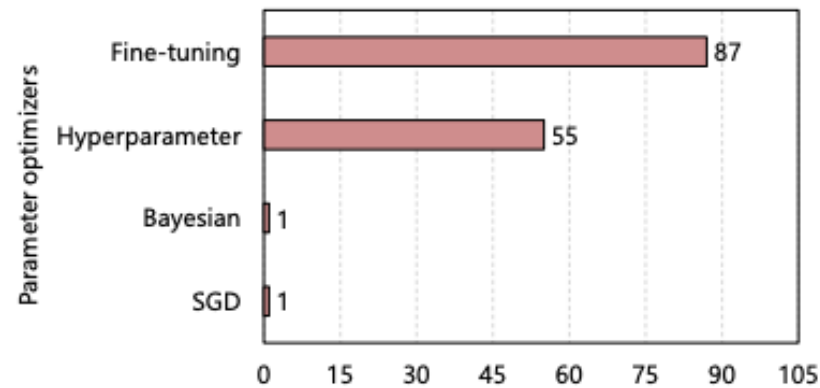
Alignment tuning



Reinforcement Learning with Human Feedback

Fine-tuning for SE Tasks

- Fine-tuning is commonly used to optimize LLMs




However, due to data scarcity and computing restrictions, prompt engineering is still frequently used as a method for adapting LLMs to solve SE tasks

[Source: Large Language Models for SE](#)

Evaluating the fine-tuned model

The evaluation metric depends on the SE task:

- Classification-tasks
 - F1 –score, precision, recall and accuracy
- Recommendation tasks
 - Mean Reciprocal Ranking (MRR), Precision@k, F1-score@k
- Generation tasks
 - BLEU
 - Pass@k
 - Accuracy@k



The correct answer was given in the first K-ranked responses.

[Source: Large Language Models for SE](#)

Technology and Libraries

- [Transformers](#)
 - Open-source Python library for building models using the Transformer architecture
- Want to use a LLM on your laptop?
 - [Llama2cpp](#) with bindings for most programming languages

Course Project

- Careful about the scope of the project
 - You should be discussing a Software Engineering problem/application
 - Using LLMs for other topics is not part of this course's scope
- Start diving deep into the type of **problem** you want to address
- **Do you need to train or fine-tune an LLM in your project?**
 - Not necessarily.
 - Survey papers, replication papers, and empirical studies are part of the course's scope

Paper Critiques

- Next week's class will be a research discussion class!
- 2 papers to read:
 - 1 paper to summarize (half a page)
 - 1 paper to critique (1 to 2 pages max)
 - Summary + Positive points (3+) + Negative Points (3+)
- Never read or critiqued a paper before?
 - Check Complementary Materials in Moodle

