

Introduction and Organization

IE685 Large Language Models and Agents



Hello

- About me:
 - Gastprofessor Dr. Ralph Peeters
 - Research Interests
 - Sustainable LLM-Agents
 - Entity Matching using Deep Learning
 - Data Integration
 - Office: B6, 26 - C 1.04
 - eMail: ralph.peeters@uni-mannheim.de
- I will co-teach the lectures and coach the student projects.



Hello

- About me:
 - Prof. Dr. Christian Bizer
 - Professor for Information Systems V
 - Research Interests:
 - Web-based Systems
 - Large-Scale Data Integration
 - Data and Web Mining
 - Room: B6, 26 - B1.15
 - eMail: christian.bizer@uni-mannheim.de
- I will co-teach the lectures and coach the student projects.



Hello

- About me:
 - M.Sc. Aaron Steiner
 - Graduate Research Associate
 - Research Interests
 - Entity Matching using LLMs
 - Data Integration
 - Office: B6, 26 - C 1.04
 - eMail: aaron.steiner@uni-mannheim.de
- I will teach the exercises and coach the student projects.



Outline

1. **Introduction to LLMs and Agents**
2. Course Organization
3. Foundations
 - Language Modelling
 - Language Representations
 - The Transformer Architecture
 - Pre-trained Language Models

Introduction to LLMs



What is a Language Model?

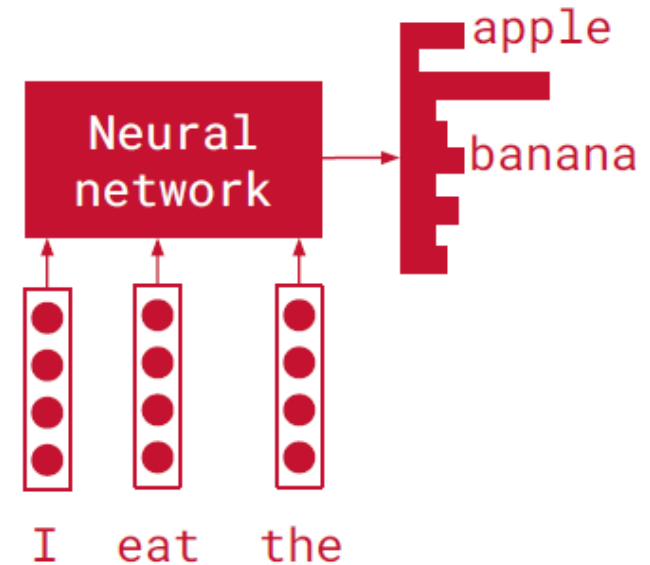
- The classic definition of a language model (LM) is a probability distribution over each possible token sequence $[w_1, w_2, \dots, w_n]$, independent of it making any sense:
 - Sally fed my cat with meat: $P(\text{Sally, fed, my, cat, with, meat}) = 0.03$
 - My cat fed Sally with meat: $P(\text{My, cat, fed, Sally, with, meat}) = 0.005$
 - fed cat meat my my with: $P(\text{fed, cat, meat, my, my, with}) = 0.0001$
- A **good** language model ideally assigns a high probability to sequences that make sense given ...
 - the structure of the actual language (English in this case)
 - any additional context in which a sentence is uttered, if available

Our Focus: Autoregressive LMs

- A type of language model based on the chain rule of probability:

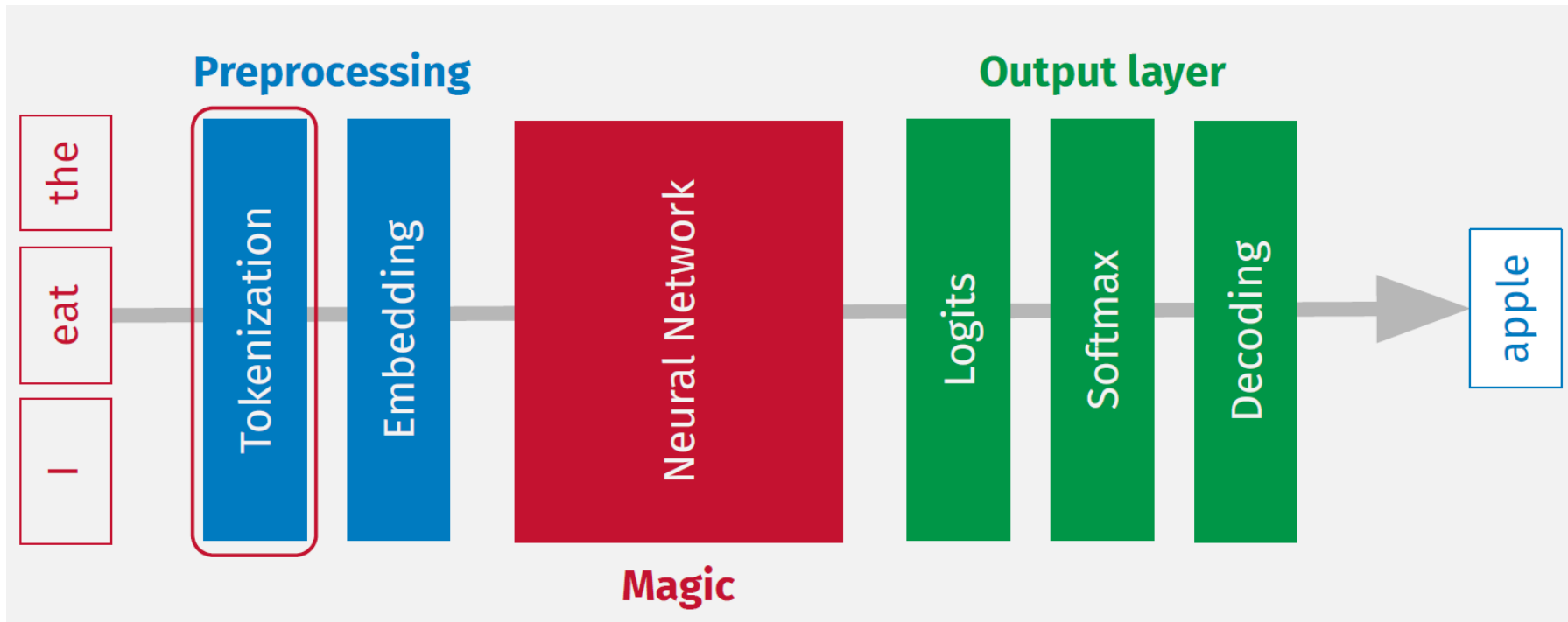
$$P(w_1, w_2, w_3, \dots, w_n) = P(w_1) * p(w_2/w_1) * p(w_3/w_1, w_2) * \dots * p(w_n/w_1, w_2, \dots, w_{n-1})$$

- $P(\text{I, eat, the, chocolate}) =$
 - $P(\text{I})$
 - $* P(\text{ate} \mid \text{I})$
 - $* P(\text{the} \mid \text{I, ate})$
 - $* P(\text{chocolate} \mid \text{I, ate, the})$



- Next Token Prediction:
 - Given some sequence of tokens $w_{[1:n]}$
 - generate the next tokens step by step, starting from w_{n+1}
 - using a large neural net to approximate the distribution $P(w_{n+1}/w_{[1:n]})$

Language Representation within LLMs



Self-supervised Pre-training using Large Training Corpora

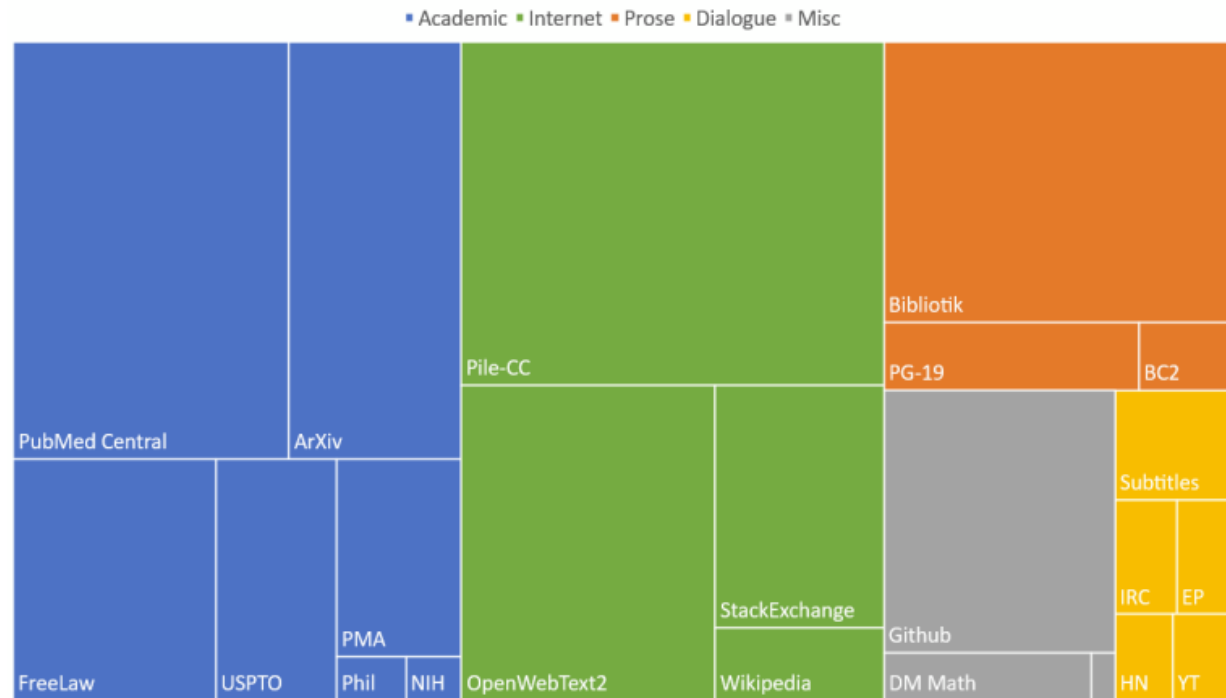
Public Corpora

- The Pile: text, 825B tokens
- The Stack: code, 900B tokens
- LAION-5B: image/caption pairs, 5B
- OpenOrca: instruction/answer pairs, 4M

OpenAI, Anthropic

- “internet data plus contracted sources”
- Size: not public

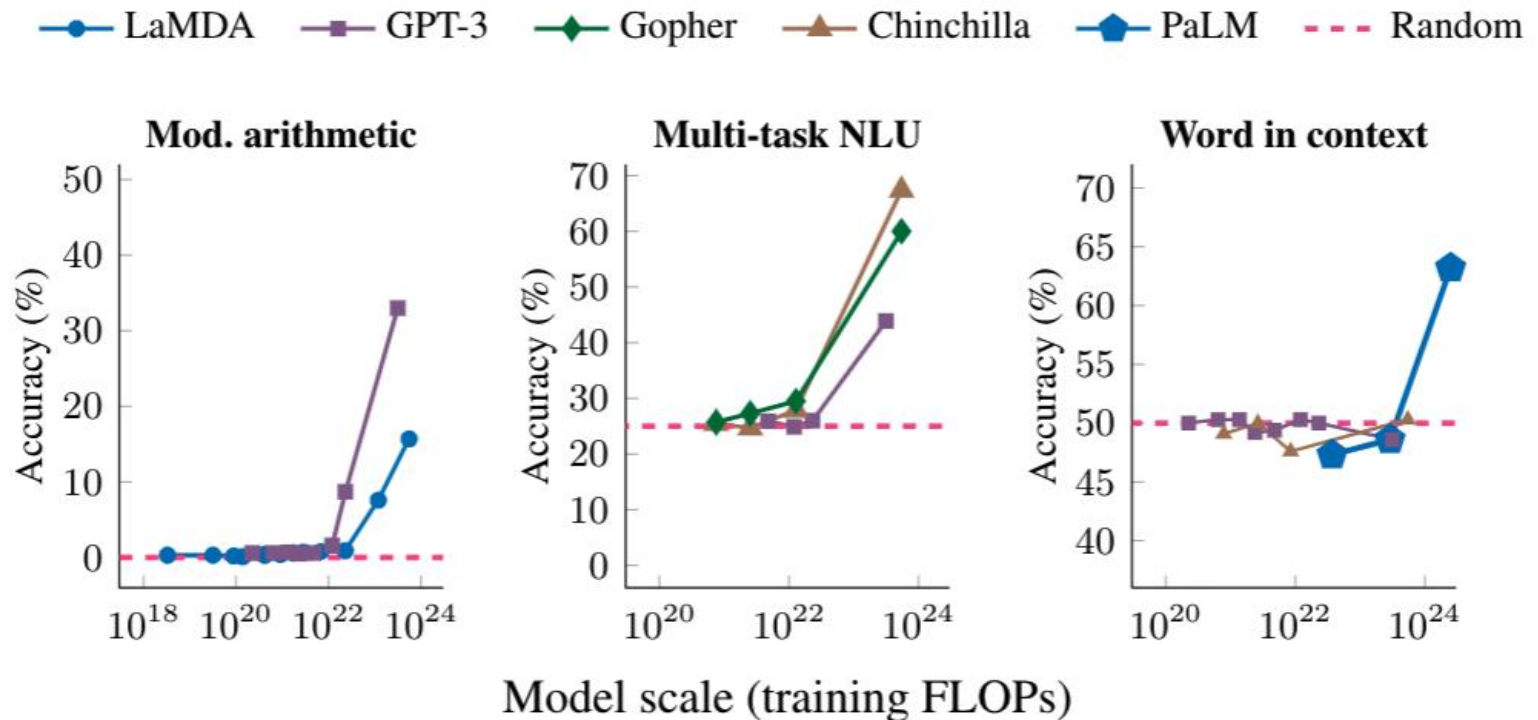
Composition of the Pile by Category



Biderman et al. (2022): Datasheet for the Pile, <https://arxiv.org/pdf/2201.07311>

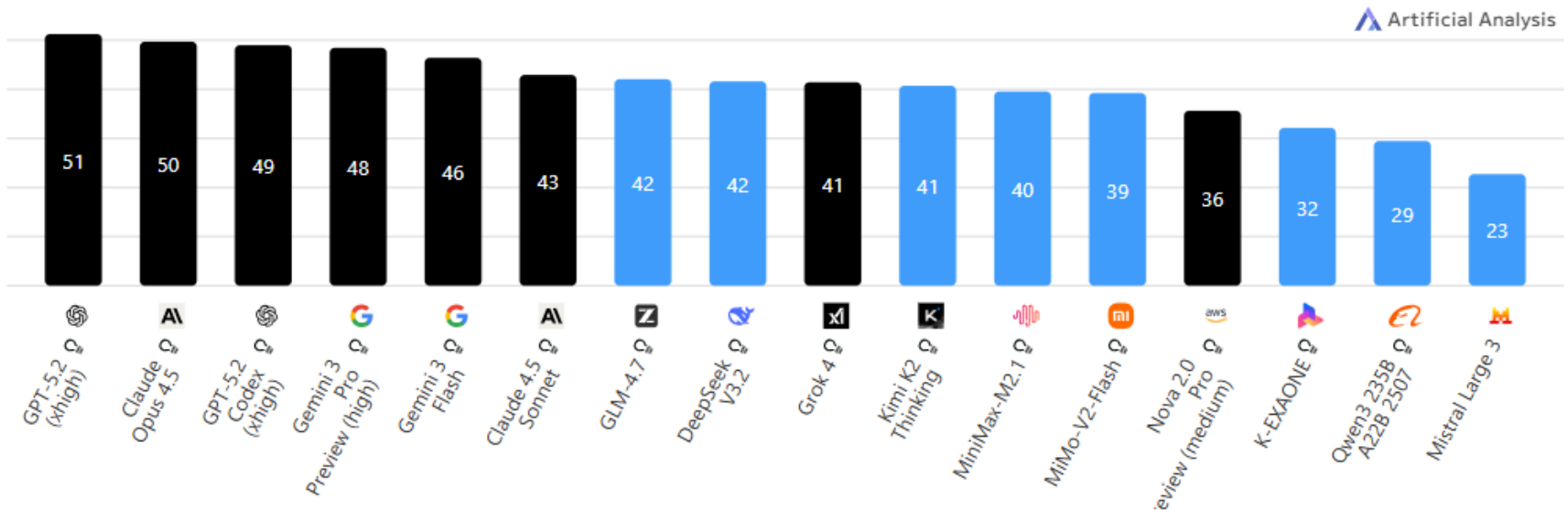
Emergent Abilities of LLMs

“Abilities that are not present in small models but arise in large models”
J. Wei et al., “Emergent Abilities of Large Language Models,” CoRR, vol. abs/2206.07682, 2022



Performance of Current LLMs

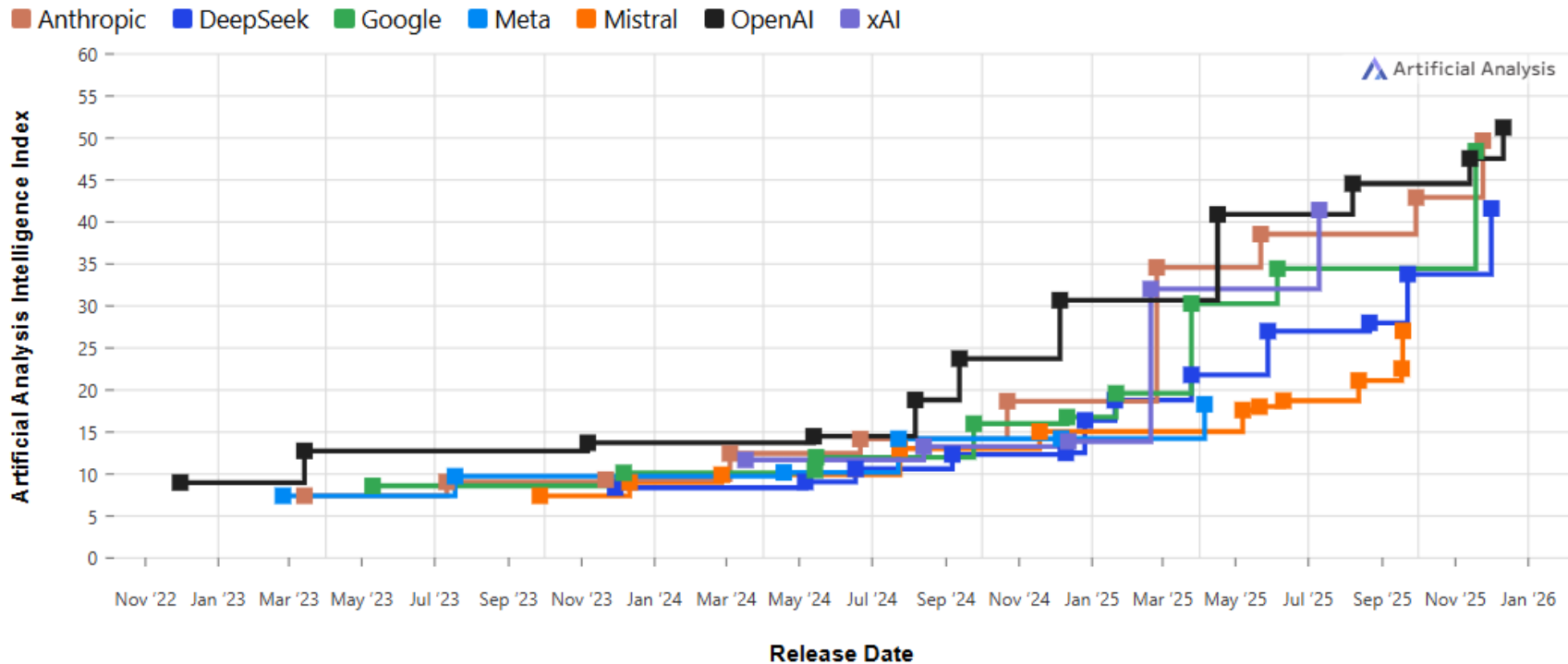
■ Proprietary ■ Open Weights



Average result of 10 benchmarks: GDPval-AA, τ^2 -Bench Telecom, Terminal-Bench Hard, SciCode, AA-LCR, AA-Omniscience, IFBench, Humanity's Last Exam, GPQA Diamond, CritPt.

<https://artificialanalysis.ai/> as of January 2026

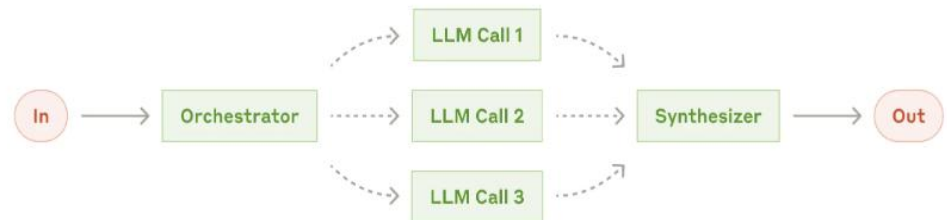
Performance of LLMs Over Time



- <https://artificialanalysis.ai/> as of January 2026

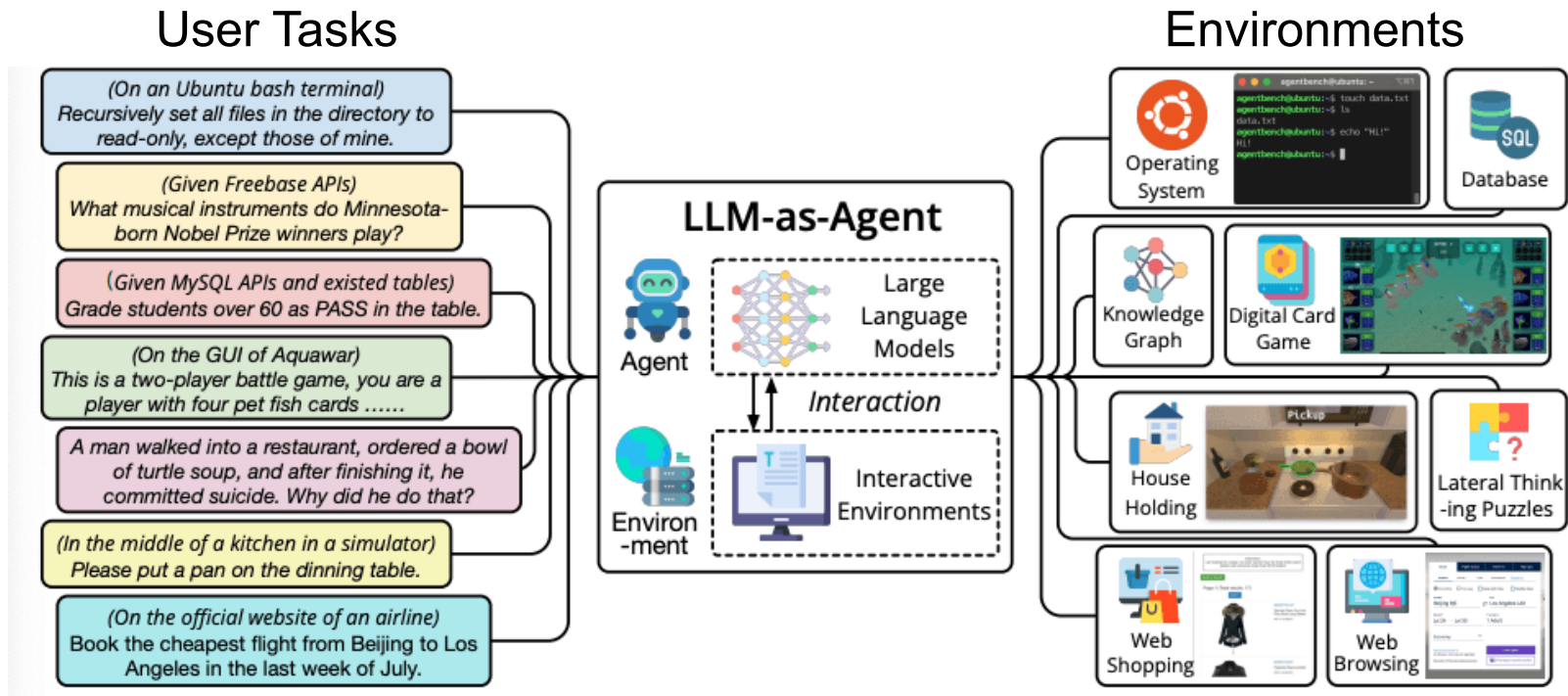
LLM Topics Covered in the Course

- Pre-training and Post-tuning LLMs
 - Instruction Tuning
 - Reinforcement Learning from Human Feedback
 - Efficient Adaptation
- Prompt Engineering
 - Prompting Techniques
 - In-context Learning
 - Chain-of-Thought Prompting
 - Prompt Chains / LLM Workflows
- Evaluating LLMs
 - Benchmarks
 - LLM-as-a-Judge



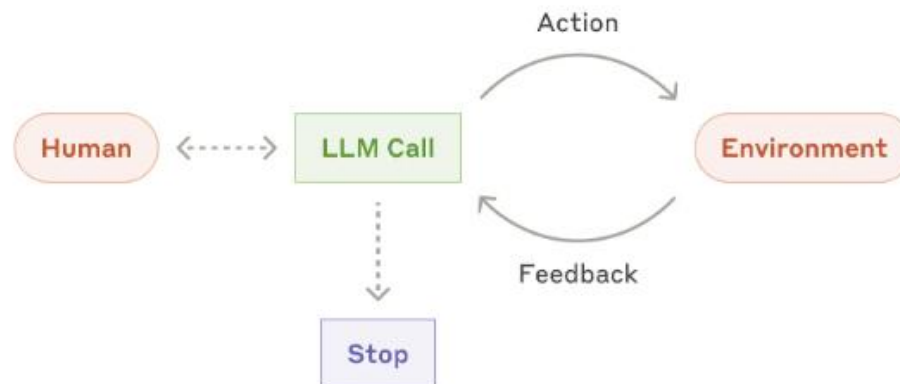
Introduction to LLM Agents

LLM agents are “intelligent” systems that **autonomously** interact with an environment to solve user tasks.



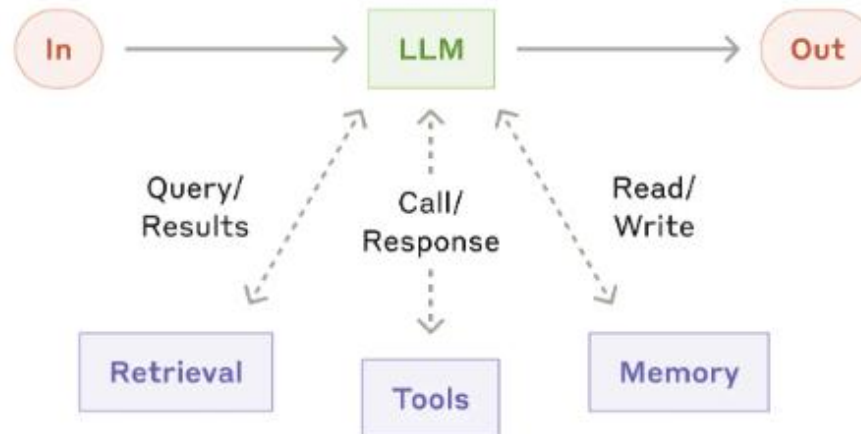
Introduction to LLM Agents

- given a task, agents plan and operate **autonomously**
 - no fixed workflow, but flexible series of actions planned by agent
- agent consists of a **loop** iterating between LLM and tool calls (actions)
- in each iteration
 1. the agent observes the environment
 2. reasons about the task given the environment and its previous actions
 3. choose an action to perform
 4. the agent's runtime environment executes the action



Tool Usage

- agents use **tools** to interact with the environment
 - search the Web or document repositories
 - memorize results of previous interactions
 - move the mouse or type using the keyboards
 - run code in execution environment
 - query databases and ERP systems



Prompt of a Web Browsing Agent

(abbreviated)

SYSTEM: You are an agent trying to solve a web task based on the content of the page and user instructions. You can interact with the page and send messages to the user. Each time you submit an action it will be sent to the browser and you will receive a new page.

USER:

Goal: Find the cheapest offer for an Iphone 13.

Observation of current step:

Currently open tabs:

Tab 0 (active tab):

Title: WebMall

URL: http://localhost:8085/

AXTree: ...

Action space:

15 different types of actions are available.

noop(wait_ms: float = 1000)

scroll(delta_x: float, delta_y: float)

keyboard_press(key: str)

click(bid: str, button: Literal['left', 'middle', 'right'] = 'left')

fill(bid: str, value: str)

hover(bid: str)

tab_focus(index: int)

new_tab() ...

Concrete Example

Here is a concrete example of how to format your answer.

<think>

I tried to set the value of year to "2022", using select_option, but it doesn't appear to be in the form. It may be a dynamic dropdown. I will try using click with the bid "a324" and look at the response from the page.

</think>

<action>

click('a324')

</action>

Agent:

<think>

To find the cheapest offer for an Iphone 13, I need to explore different webshops. I will start by visiting Shop 1.

</think>

<action>

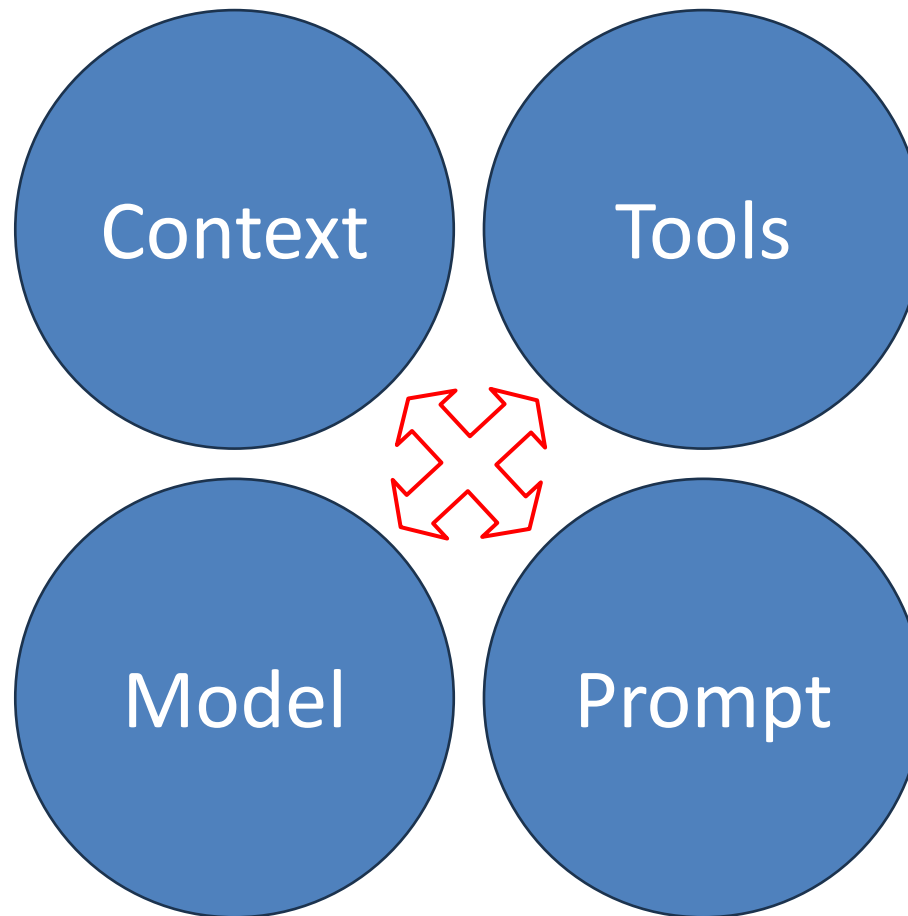
click('9')

</action>

Agent Topics Covered in the Course

- Tool Usage for LLMs
 - Model Context Protocol (MCP)
 - Retrieval Augmented Generation (RAG)
- The ReAct Paradigm
- Unified Framework for LLM Agents
 - Planning and Self-Reflection
 - Short- and Long-Term Memory
 - Multi-Agent Orchestration
- Context Engineering
 - Context Retrieval, Context Summarization
- Evaluating Agents
 - Benchmarks, Trajectory Mining
- Safety and Security of LLMs and LLM Agents

Interaction of the Four Basic Components



Economic Viewpoint

Young will suffer most when AI 'tsunami' hits jobs, says head of IMF

Kristalina Georgieva says research suggests 60% of jobs in advanced economies will be affected, with many entry-level roles wiped out



The Guardian, 23.1.2026

Effizienzsteigerung durch Technik

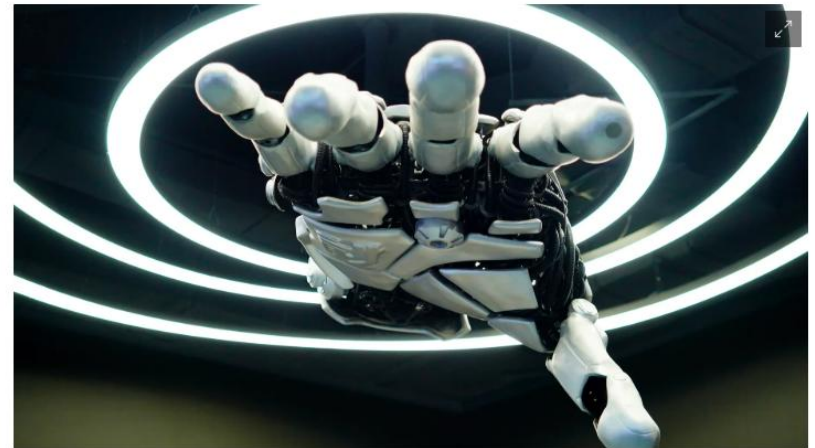
Steht dank KI-Agenten das Ende der bezahlten Arbeit bevor?

Shoppen, Urlaube buchen, Papierkram: Künstliche Intelligenz übernimmt immer mehr Aufgaben selbstständig. Was das für Branchen, Unternehmen und Jobs bedeutet. Und für Sie.

Von **Simon Book**

31.12.2025, 06:38 Uhr • aus [SPIEGEL Chronik 1/2025](#)

12 Min



Der Spiegel, 31.12.2025

- The course lays the foundation for better informed discussions.

Outline

1. Introduction to LLMs and Agents
- 2. Course Organization**
3. Foundations
 - What is a Language Model?
 - Language Representations
 - The Transformer Architecture
 - Pre-trained Language Models

The First Half of the Course

- Lectures (IE685)
 - Foundational knowledge for understanding LLMs
 - Neural architectures, pre-training and post-tuning, prompt engineering, ReAct agents, tool usage, evaluation, retrieval-augmented generation, context engineering, safety and security, ...
 - **Goal:** Learn and understand concepts and methods
- Exercises
 - Practical applications of the lecture concepts
 - Introduce tools: Huggingface Transformers, LangChain, Tavily
 - **Goal:** Learn to apply concepts in practice and prepare you for the group projects

The Second Half of the Course


- Group Projects (IE686)
 - Work in teams of five students on the practical application of LLM workflows or LLM agents to a problem
 - **Goal:** Understand a problem domain and apply the learned concepts and tools to solve the problem!
 - Teams write a 12-page report about their project and present their results during a presentation at the end of the semester
- Course Grading
 - **IE685 (3 ECTS): Written Exam (100%)**
 - **IE686 (3 ECTS): Group Projects (70% written project report, 30% presentation of results)**

Course Schedule

Week	Tuesday	Wednesday
10.02	Lecture: Introduction to LLMs and LLM Agents	-
17.02	Lecture: Post-Training and Efficient Adaptation	Exercise: Huggingface Transformers
24.02	Lecture: Prompt Engineering	Exercise: LLM Workflows with LangChain
03.03	Lecture: LLM Agents and Tool Use	Exercise: Agent Programming using LangChain
10.03	Lecture: Retrieval Augmented Generation	Exercise: RAG Workflows using Tavily and LangChain
17.03	Lecture: Context Engineering	Exercise: Coding with LLM Agents in VS Code
24.03	Project: Introduction to Student Projects	Project: Preparation of Project Outline
	Easter Break	Easter Break
14.04	Project: Feedback about Outlines	Project: Coaching
21.04	Lecture: Agent Safety and Security	Project: Coaching
28.04	Project: Work	Project: Coaching
05.05	Project: Work	Project: Coaching
12.05	Project: Work	Project: Coaching
19.05	Project: Presentation of Project Results	Project: Presentation of Project Results
XX.05	Final Exam	

Course Organization

- Course Webpage
 - The lecture slides and exercises tasks are published on this webpage: <https://www.uni-mannheim.de/dws/teaching/course-details/courses-for-master-candidates/ie-685-large-language-models-and-agents/>
- Time and Location
 - **Lecture:** Every Tuesday, 10:15 to 11:45
Room: B6 A101
 - **Exercise:** Every Wednesday, 12:00 to 13:30
Room: B6 A101
 - **Starting 10.02.2025**



UNIVERSITY OF MANNHEIM
School of Business, Economics
and Mathematics

Home Research People Career Teaching

University of Mannheim | Data and Web Science Group | Teaching | Course Details | Courses for Master Candidates | IE 685 Large Language Models and Agents

Large Language Models and Agents (FSS2026)

Large language models (LLMs) such as ChatGPT, Claude, Llama, Gemini, and Mistral have the potential to enable a wide range of new applications and to significantly improve the performance of existing systems. The course introduces students to LLMs and LLM agents and teaches them how to employ language models and LLM agents within different applications.

The course covers the following topics:

1. Introduction to LLMs
2. Prompt engineering
3. LLM-based agents
4. Tool use and environment interaction
5. Retrieval augmented generation
6. Context engineering for LLM agents
7. Safety and security of LLM agents
8. Evaluation of LLM agents

The course participants gain knowledge about the principles of training LLMs. They will be able to identify opportunities for employing LLMs in business applications and will learn to apply prompt engineering techniques as well as agent frameworks for solving complex tasks. In the second half of the course, the participants apply their knowledge in team projects and will report about the results of the projects in the form of a written report as well as an oral presentation.

Course Organization

- Course in ILIAS
 - Lecture: <https://ilias.uni-mannheim.de/goto.php/grp/1728875>
 - Projects: <https://ilias.uni-mannheim.de/goto.php/grp/1726394>
 - Need to be accepted for the course to access
- Contain:
 - **Forums** (one for lecture, one for projects)
 - For online communication/discussion during the course
 - Not just with us, but also amongst each other!
 - Don't be shy to post questions and bringing your unique knowledge to discussions, so we can all learn from each other!
 - **Project Administration** (just for projects)
 - Team creation / communication
 - Submission of deliverables

Literature and Credits

- Some supporting literature for the course
 - Zhao, et al.: [A Survey of Large Language Models](#), arXiv:2303.18223, 2024.
 - Wang, et al.: [A Survey on Large Language Model Based Autonomous Agents](#). Frontiers of Computer Science, 2024.
 - Sager et al.: AI Agents for Computer Use: [A Review of Instruction-based Computer Control, GUI Automation, and Operator Assistants](#). arXiv:2501.16150, 2025.
 - Mohammadi, et al.: [Evaluation and benchmarking of LLM agents: A survey](#). SIGKDD Conference on Knowledge Discovery and Data Mining, 2025.
 - Wang, et al.: [A Comprehensive Survey in LLM\(-Agent\) Full Stack Safety: Data, Training and Deployment](#). arXiv:2504.15585, 2025.
 - Zhou, et al.: [A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT](#), 2024, International Journal of Machine Learning and Cybernetics.
 - Daniel Jurafsky & James H. Martin: [Speech and Language Processing](#). (3rd edition)
- The slide set of this lecture builds on slides from:
 - Jiaxin Huang, Mrinmaya Sachan, Danqi Chen, Daniel Jurafsky & James H. Martin
 - Many thanks to all of you!

Tools

- Transformers

- Library for fine-tuning and inference with pre-trained and large language models
- Streamlined interface



- LangChain

- Library for interaction with hosted and local LLMs
- Supports LLM workflows, tool usage, retrieval augmented generation
- Provides ReAct agent loop



Project API and Hardware Usage

- bwUniCluster 2.0
 - All students can register
 - Provide compute servers with modern GPUs (up to 8 per machine)
 - Uses a job queuing system for distributing resources
 - **Good and free option** for locally hosting open-source LLMs
- APIs for hosted LLMs
 - **We cannot reimburse you** for API costs, e.g. OpenAI, Anthropic, ...
 - You may consider using cheap but powerful models like GPT5-mini
 - Check for free tier offers from different providers (e.g. Groq API)
 - We will give you more tips in the exercises...
- Google Colab
 - Could be useful for **prototyping** with small open-source models

Outline

1. Introduction to LLMs and Agents
2. Course Organization
- 3. Foundations**
 - **Language Modelling**
 - Language Representations
 - The Transformer Architecture
 - Pre-trained Language Models

What is a Language Model?

- The classic definition of a language model (LM) is a probability distribution over each possible token sequence $[w_1, w_2, \dots, w_n]$, independent of it making any sense:
 - Sally fed my cat with meat: $P(\text{Sally, fed, my, cat, with, meat}) = 0.03$
 - My cat fed Sally with meat: $P(\text{My, cat, fed, Sally, with, meat}) = 0.005$
 - fed cat meat my my with: $P(\text{fed, cat, meat, my, my, with}) = 0.0001$
- A **good** language model ideally assigns a high probability to sequences that make sense given ...
 - the structure of the actual language (English in this case)
 - any additional context in which a sentence is uttered, if available

Our Focus: Autoregressive LMs

- A type of language model based on the chain rule of probability:

$$P(w_1, w_2, w_3, \dots, w_n) = P(w_1) * p(w_2/w_1) * p(w_3/w_1, w_2) * \dots * p(w_n/w_1, w_2, \dots, w_{n-1})$$

- $P(\text{Sally, fed, my, cat, with, meat}) = P(\text{Sally})$
 - * $P(\text{fed} \mid \text{Sally})$
 - * $P(\text{my} \mid \text{Sally, fed})$
 - * $P(\text{cat} \mid \text{Sally, fed, my})$
 - * $P(\text{with} \mid \text{Sally, fed, my, cat})$
 - * $P(\text{meat} \mid \text{Sally, fed, my, cat, with})$

➔ Conditional probability

- We will also see some other types of language models later

Tokenization

A cute teddy bear is reading.

arbitrary A cute teddy bear is reading .

word A cute teddy bear is reading .

sub-word A cute ted ##dy bear is read ##ing .

A _ c u t e _ t e d d y _ b e a r _ i s _ r e a d i n g .

➔ for more info also see course
Advanced Methods in Text Analytics (IE 696)

Language Generation

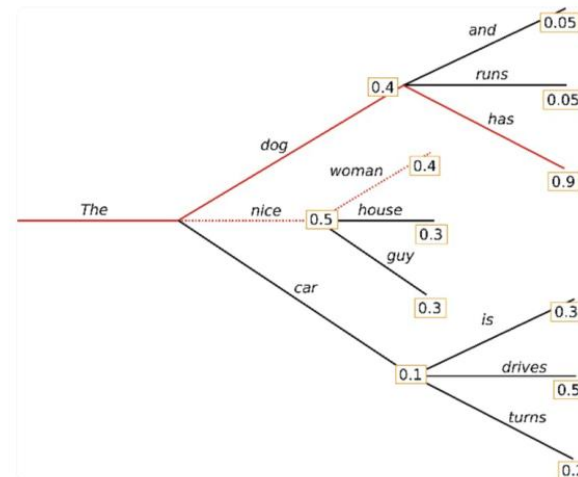
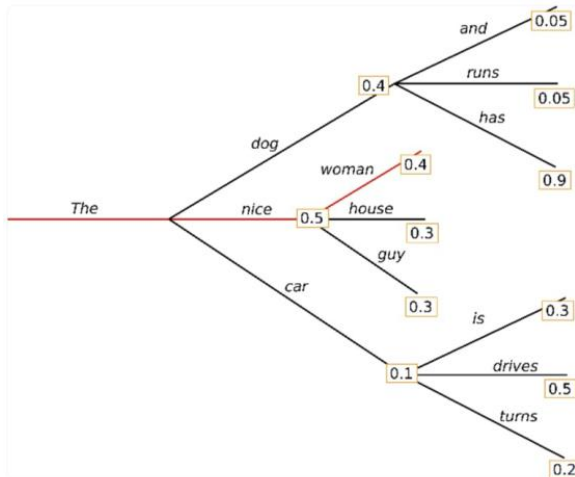
- Assume we have a good language model and a given text prompt $w_{[1:n]}$
 - Now we want to generate a good completion for this prompt with some length L
 - How to find $w_{[n+1:n+L]}$ with the highest probability?
 - Enumerate over all possible combinations? 😞

➔ Next token prediction

- Generate tokens step by step starting from w_{n+1} using $p(w_{n+1}/w_{[1:n]})$
- For selecting the next token with $p(w_{n+1}/w_{[1:n]})$, there are different decoding approaches

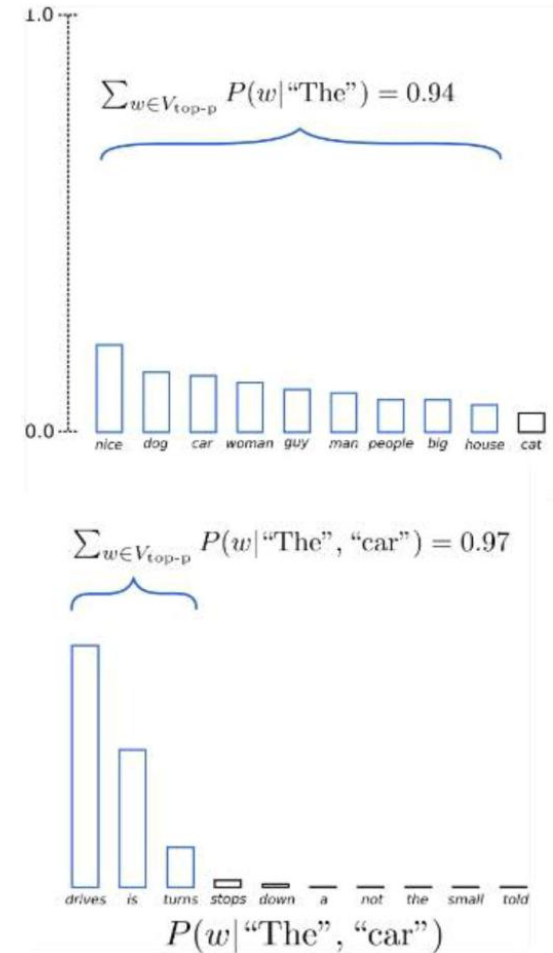
Decoding Approaches

- **Greedy decoding:** At each step, always select w_t with the highest $p(w_t/w_{[1:t-1]})$.
- **Beam Search:** Keep track of k possible paths at each step instead of just a single one. Reasonable beam size k: 5-10



Decoding Approaches

- **Top-k sampling:** At each step, randomly sample the next token from $p(w_t/w_{[1:t-1]})$, but restrict to only the k most probable tokens.
 - ➔ Allows to **control diversity:**
 - Increasing k leads to more creative outputs with an increasing risk of getting bad outputs
 - Decreasing k gives “safer” but less creative outputs
 - Problem: fixed k can cover wildly different amount of probability mass
- **Top-p sampling:** At each step, randomly sample the next token from $p(w_t/w_{[1:t-1]})$, but restrict to the set of tokens with a cumulative probability of p
 - ➔ Throw away long-tail tokens
- Top-k and Top-p can be used together!



Decoding Approaches

- **Temperature sampling**

- Reshape the distribution instead of truncating it
- Inspired by thermodynamics
 - System at high temperature is flexible and can explore many possible states
 - System at lower temperature is likely to explore a subset of lower energy (better) states

- **Low-temperature sampling ($\tau \leq 1$)**

- increases the probability of the most probable words
- decreases the probability of the rare words

$$\text{~~y = softmax}(u)~~$$

$$\mathbf{y} = \text{softmax}(u/\tau)$$

Temperature sampling

$$\mathbf{y} = \text{softmax}(u/\tau) \quad 0 \leq \tau \leq 1$$

- Why does it work?
 - When τ is close to 1 the distribution does not change much
 - The lower τ is, the larger the scores being passed to the softmax
 - Softmax pushes high values toward 1 and low values toward 0
 - Large inputs pushes high-probability words higher and low probability words lower, making the distribution more greedy
 - As τ approaches 0, the probability of the most likely word approaches 1

Sounds great but...

Question: How do we actually train a good language model?

Sounds great but...

Question: How do we actually train a good language model?

Answer: By maximizing the language model probability over an observed large corpus of text.

N-gram Language Models

- For example: Bi-gram language models based on co-occurrence of two words
- $P(w_N | w_{N-1}) = C(w_{N-1}, w_N) / C(w_{N-1})$
 - $P(\text{to} | \text{want}) = 608/927 = 65.59\%$
 - $P(\text{spend} | \text{want}) = 1/927 = 0.11\%$

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Curse of Dimensionality

- Limitations of N-gram models
 - **Limited context length:** N-grams have a finite context window of length N, which means they cannot capture long-range dependencies or context beyond the previous N-1 words
 - **Sparsity:** As N increases, the number of possible N-grams grows exponentially, leading to sparse data and increased computational demands
 - Suppose the vocabulary size is V, the number of possible N-grams increases to V^N
 - Usually V is larger than ten thousand. Representing each word as a one-hot vector is **inefficient** and **ignores word semantics**
 - “Dogs” and “cats” are more similar than “dogs” and “rectangular”

Outline

1. Introduction to LLMs and Agents
2. Course Organization
- 3. Foundations**
 - Language Modelling
 - **Language Representations**
 - The Transformer Architecture
 - Pre-trained Language Models

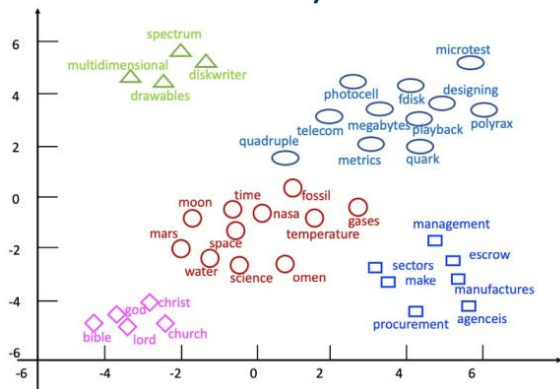
Language Representations

- Ludwig Wittgenstein: *“The meaning of a word is its use in the language”*
 - How to represent words while accounting for their meaning?
- 1. Words are defined by their **environment** (the words around them)
 - *“If A and B have almost identical environments, we say that they are synonyms”* – Zellig Harris (1954)
- 2. Words are defined by **different dimensions**
 - Which can be represented as a point in a multi-dimensional space
 - E.g. 3 affective dimensions of words (Osgood et al. 1957)

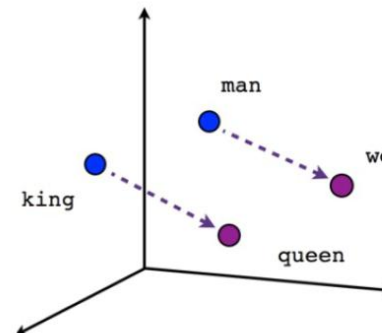
	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

Distributed Representations

- A milestone in NLP and ML:
 - Unsupervised learning of text representations – no supervision needed
 - Embed previously one-hot vectors into lower dimensional space
 - **Word embeddings** have fixed dimensions
 - Addresses sparsity of bag-of-words model (curse of dimensionality)
- Embeddings capture relevant properties of word semantics
 - Word similarity: Words with similar meaning are embedded closer
 - Word analogy: Linear relationships between words (e.g. king - queen = man - woman)



Word Similarity



Word Analogy

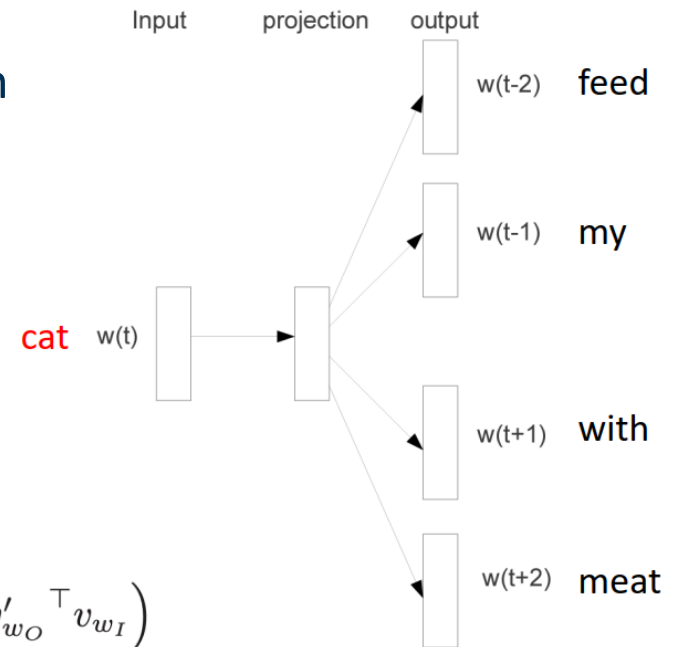
Distributed Representations

Intuition: Why embeddings?

- With **words**, a feature is a word identity (e.g. early bag-of-words models, see Data Mining lecture videos)
 - For example Feature 5: “terrible” with no information about context
 - Requires exact same word to be in training and test sets
- With **embeddings**:
 - Feature is a vector in a semantic space
 - Feature 5: [35,22,17,...]
 - In the test set there might be a similar vector [34,21,14,...]
 - It is possible to generalize to **similar but unseen** words!

Word2Vec

- Assumption: If two words have similar contexts, then they have similar semantic meanings!
- Training objective: Learn to predict word(s) in nearby context
 - Skip-gram: predict context from center word
 - CBOW: predict center word from context



Co-occurring words in a **local context window**

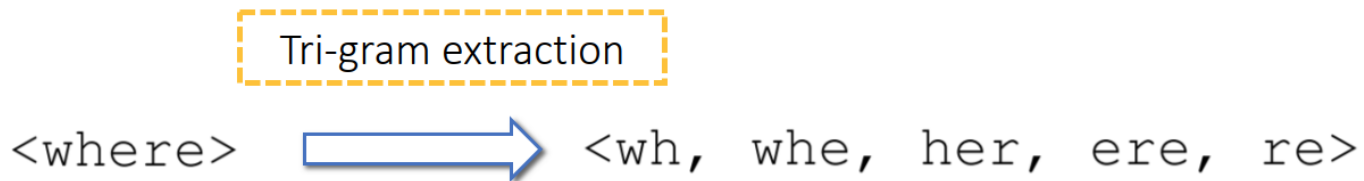
$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

$$p(w_O | w_I) = \frac{\exp(v'_{w_O} \top v_{w_I})}{\sum_{w=1}^W \exp(v'_w \top v_{w_I})}$$

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed Representations of words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 26.

Extension to Subwords: fastText

- fastText improves on Word2Vec by incorporating subword information into word embeddings



- Words are represented by aggregating their n-gram embeddings
- Allows to also embed words unseen during training

Word2Vec probability expression

$$p(w_O | w_I) = \frac{\exp\left(v'_{w_O} \top v_{w_I}\right)}{\sum_{w=1}^W \exp\left(v'_w \top v_{w_I}\right)}$$

$$\sum_{g \in \mathcal{G}_w} \mathbf{z}_g \top \mathbf{v}_c$$

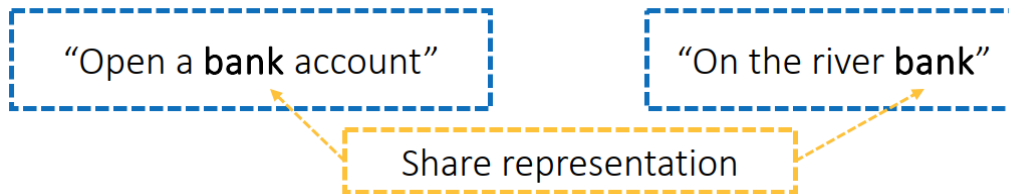
Represent a word by the sum of the vector representations of its n-grams

N-gram embedding

Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, pp.135-146.

Limitations of Word2Vec

- The embeddings are context-free
 - Each (sub-)word is mapped to only one vector
 - Polysemous words with wildly different meaning have same vector



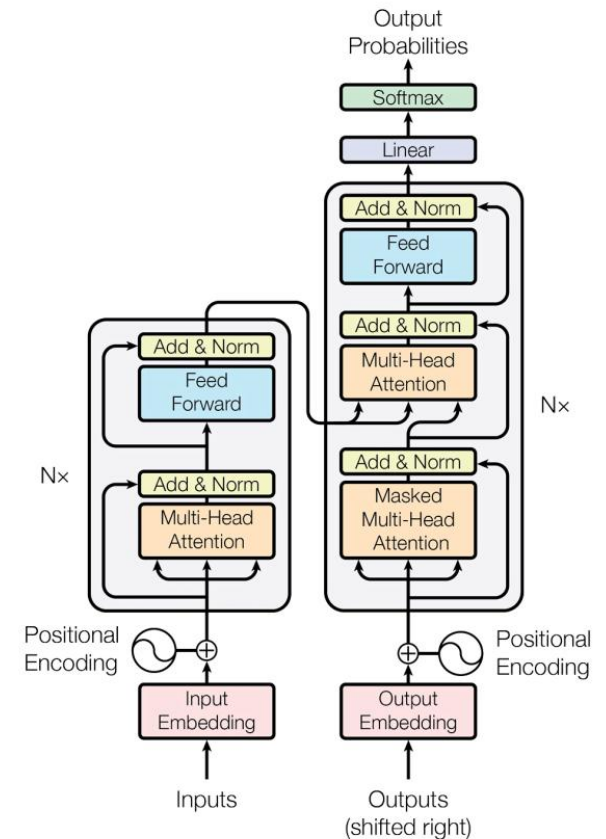
- The embeddings do not consider the order of words
- Every word is weighted the same, regardless of importance

Outline

1. Introduction to LLMs and Agents
2. Course Organization
- 3. Foundations**
 - Language Modelling
 - Language Representations
 - **The Transformer Architecture**
 - Pre-trained Language Models

The Transformer Architecture

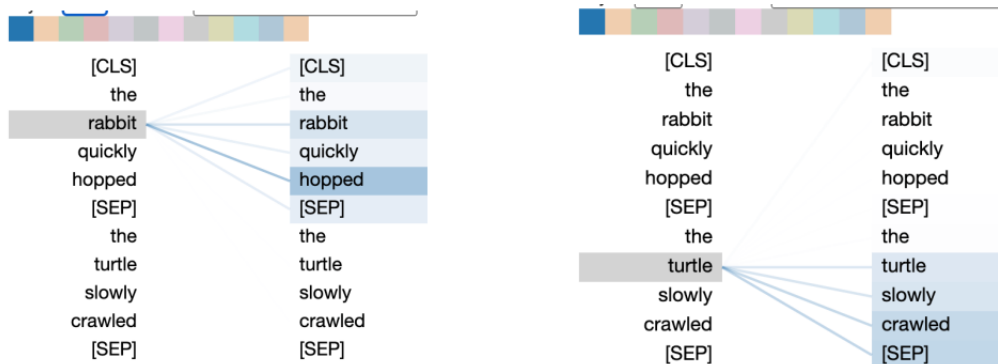
- Input Embedding
- Positional Encoding
- 12 Transformer layers
 - 6 encoder layers
 - 6 decoder layers
- Linear + Softmax layer for next word prediction



Vaswani, A., et al., 2017. Attention is All You Need. *Advances in Neural Information Processing Systems*.

Self-Attention

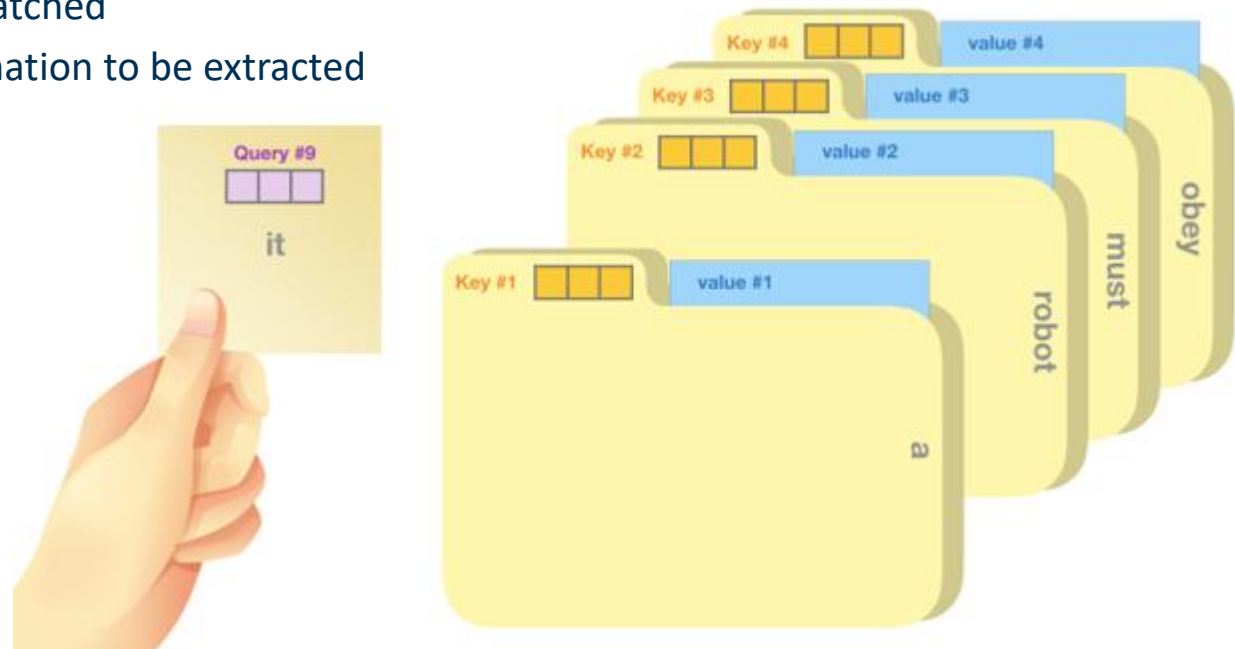
- Self-Attention: Each token attends to every other token in the sequence with differing weights
- Embeddings are **contextualized** based on surrounding words
- Demo for the BERT Transformer:
<https://github.com/jessevig/bertviz>



➔ for more info see also see course
Advanced Methods in Text Analytics (IE 696)

Self-Attention

- Terminology:
 - Query: to match others
 - Key: to be matched
 - Value: information to be extracted

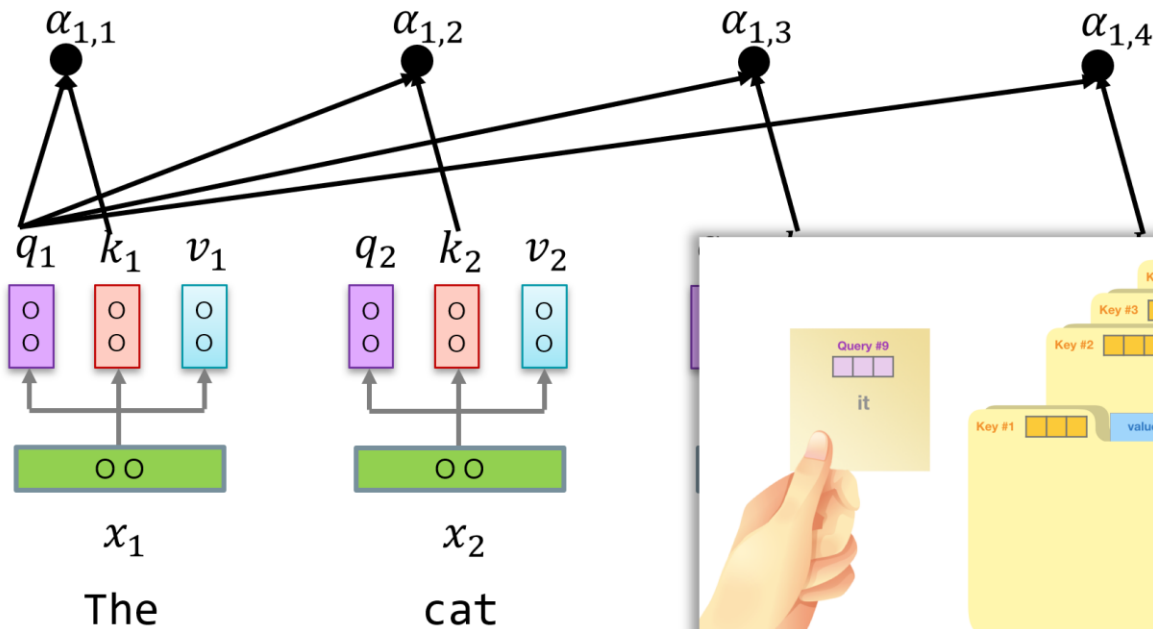


Self-Attention

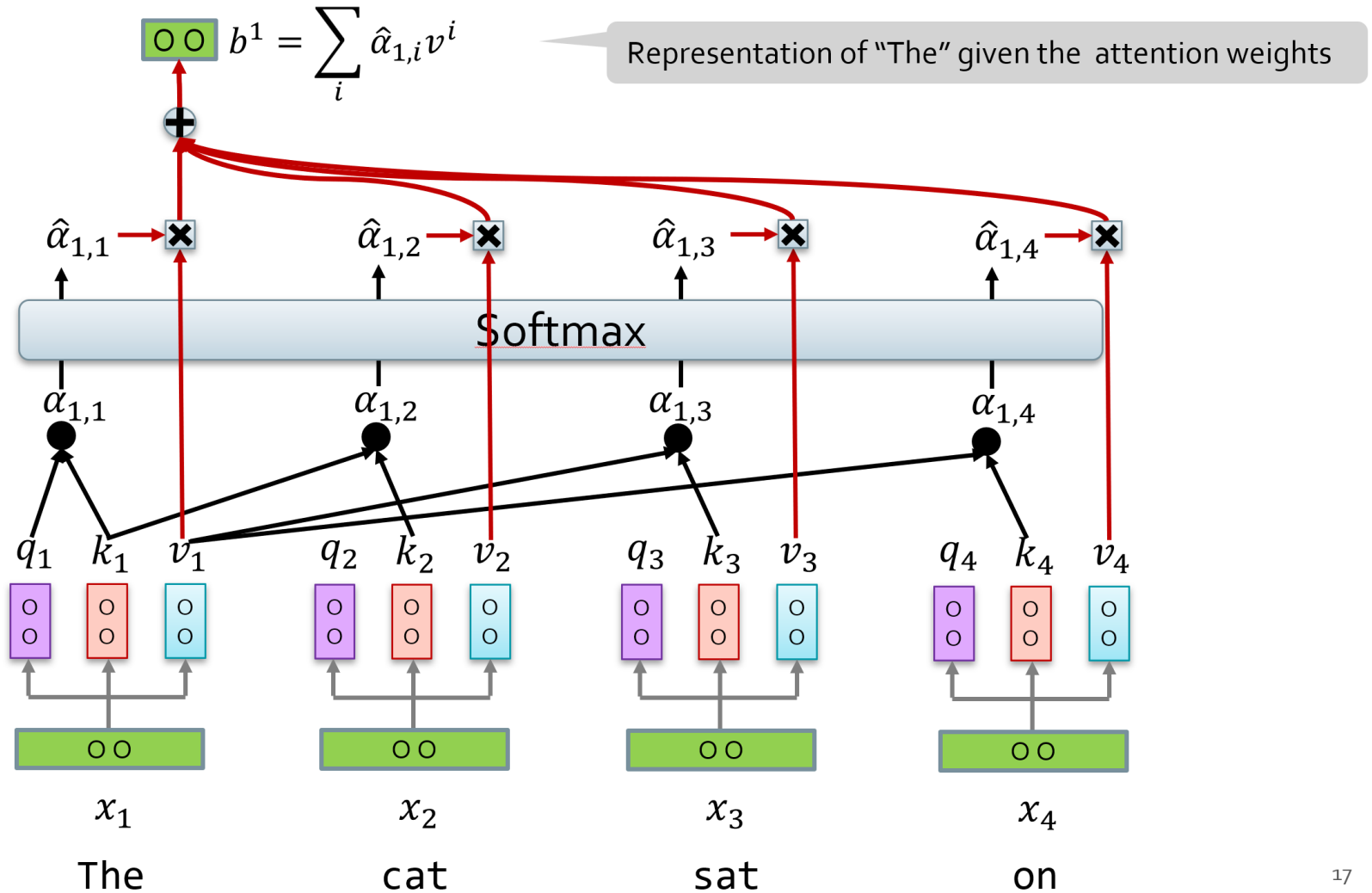
$$\alpha_{1,i} = \underbrace{q^1 \cdot k^i}_{\text{Scaled dot product}} / \sqrt{d}$$

q: query (to match others)
k: key (to be matched)
v: value (information to be extracted)

How much should "The" attend to other positions?



Self-Attention

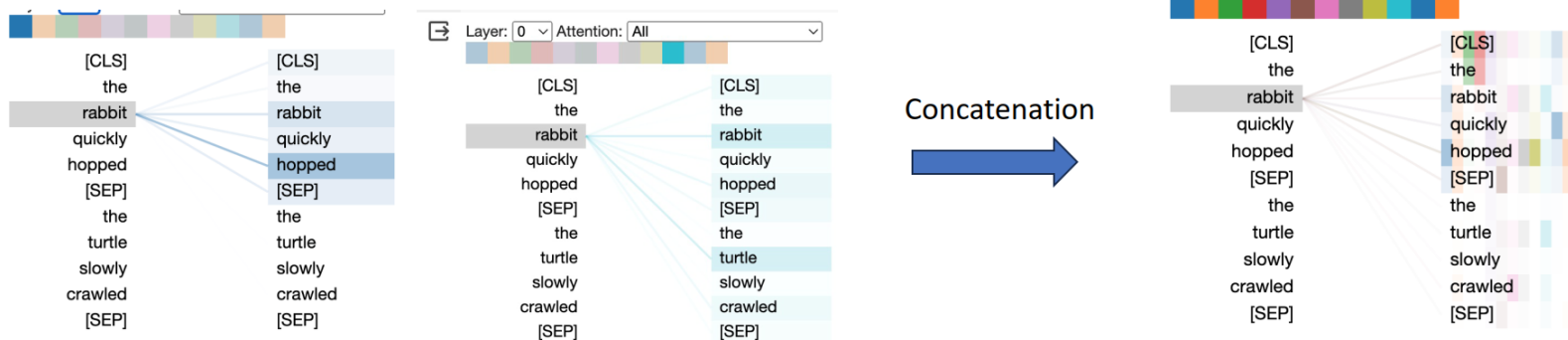


Multi-Head Attention

- Input: Multiple independent sets of query, key, value matrices
- Output: Concatenated outputs of all attention heads
- Advantage: Each attention head can focus on different patterns of the data

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

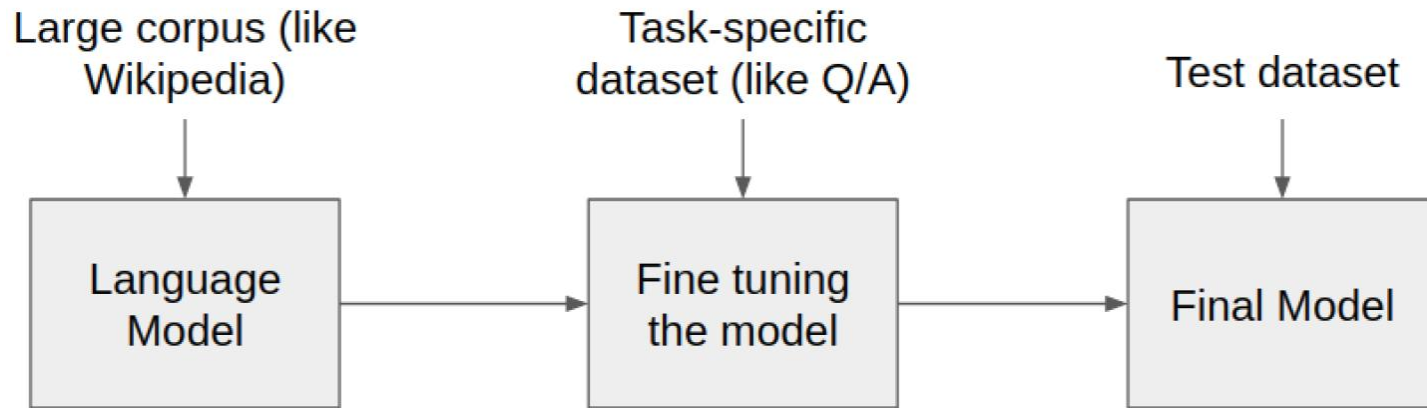


Outline

1. Introduction to LLMs and Agents
2. Course Organization
- 3. Foundations**
 - Language Modelling
 - Language Representations
 - The Transformer Architecture
 - **Pre-trained Language Models**

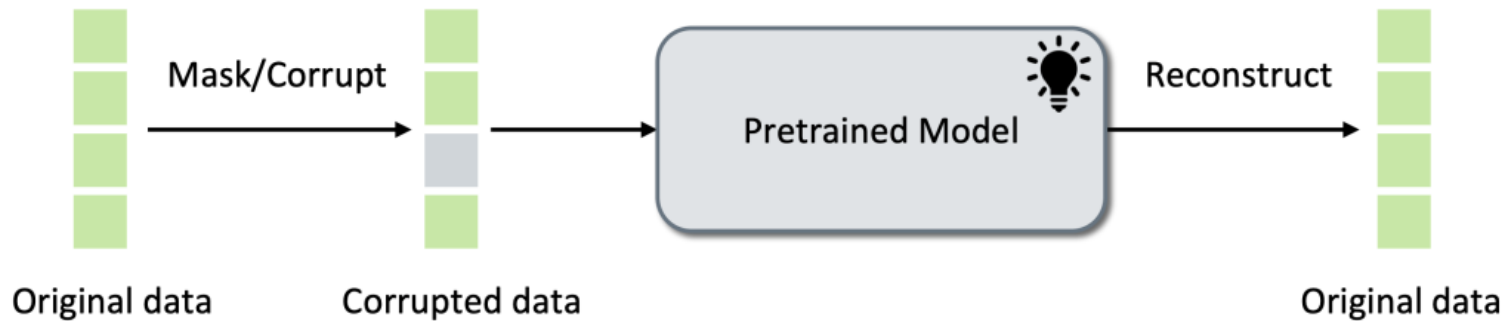
Pre-trained Language Models

- **Pre-training:** Train deep language models (usually Transformer-based) via **self-supervised** objectives on **large-scale general-domain corpora**
- **Fine-tuning:** Adapt the pre-trained language models (PLMs) to downstream tasks using task-specific data
- **Transfer Learning - The Power of PLMs:** Encode generic linguistic features and knowledge learned through large-scale pre-training, which can be effectively transferred to the target applications



General Pre-training Idea

- Self-supervised learning
- Make a part of the input unknown to the model
- Let the model predict that unknown part based on the known part



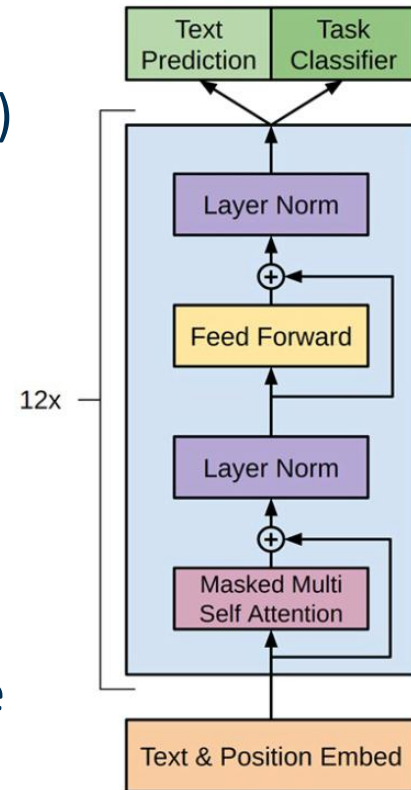
Generative Pre-training (GPT)

- Architecture: multi-layer transformer decoder
- Leverages unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

$$\mathcal{L}_{\text{LM}} = - \sum_i \log p(x_i | \boxed{x_{i-k}, \dots, x_{i-1}})$$

k previous tokens as context

- The Transformer uses unidirectional attention masks (every token can only attend to previous tokens)
- Decoder architecture is the prominent choice in large language models



Radford et al., 2018. [Improving Language Understanding by Generative Pre-Training](#)

Radford et al., 2019. [Language Models are Unsupervised Multitask Learners](#)

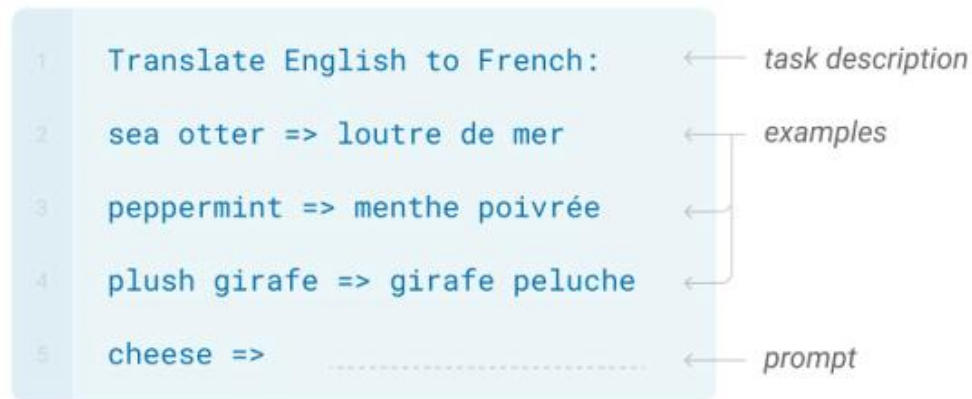
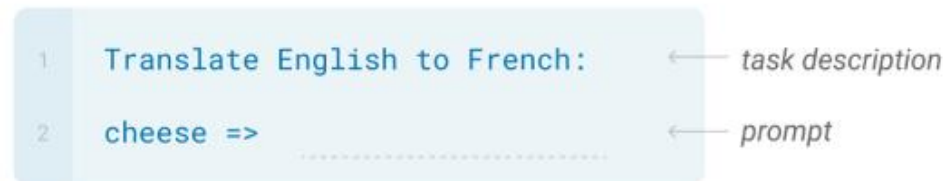
Brown et al., 2020. Language Models are Few-shot Learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems (pp. 1877-1901).

Decoder Pre-training

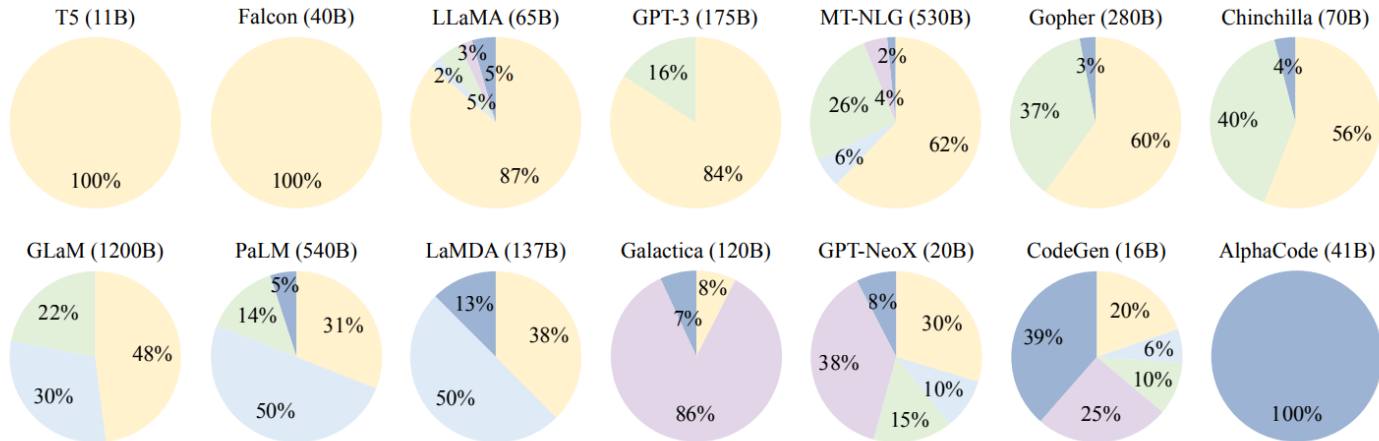
$$\mathcal{L}_{LM} = - \sum_i \log p(x_i | x_{i-k}, \dots, x_{i-1})$$



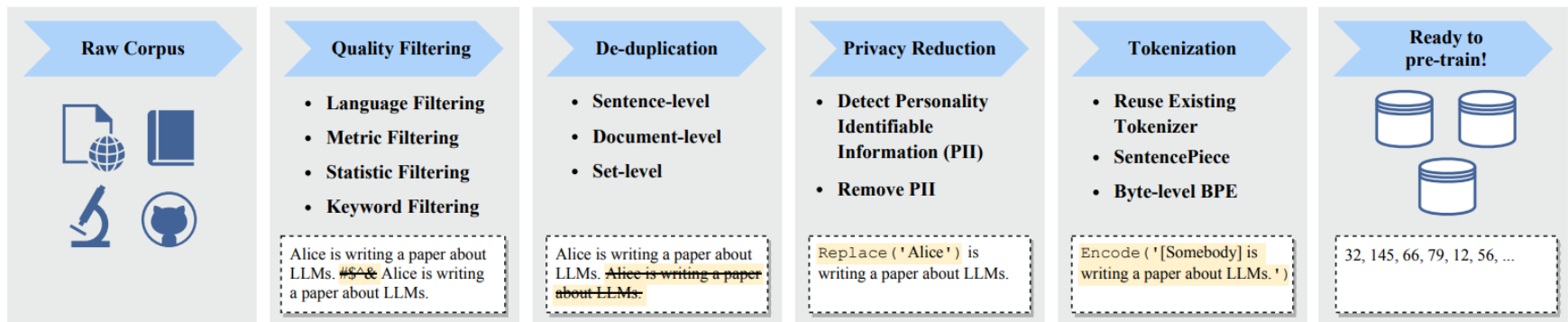
Usage of Decoder Models



Pre-training Data

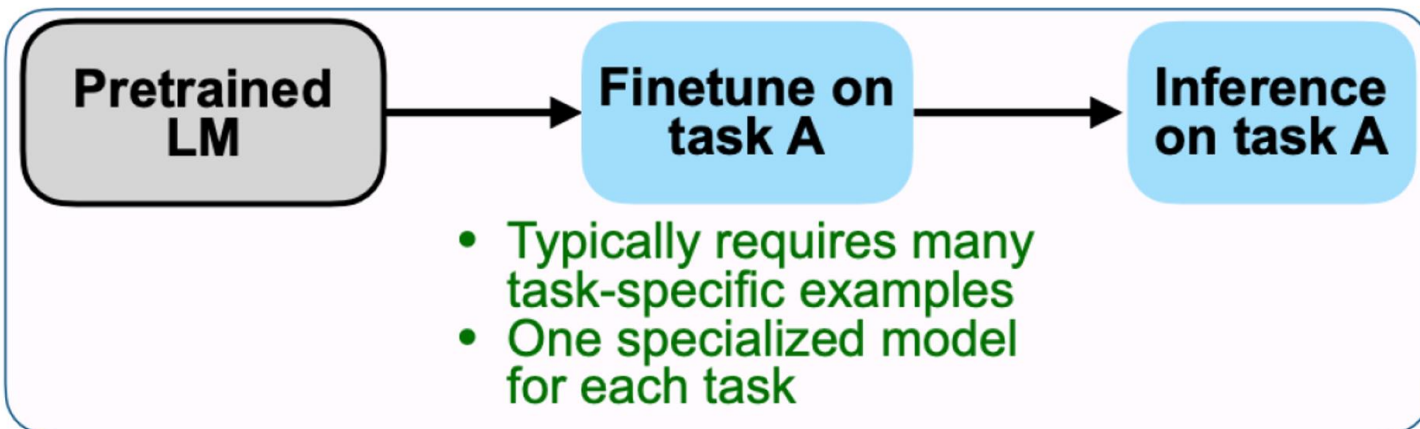


- Webpages
- Conversation Data
- Books & News
- Scientific Data
- Code
- C4 (800G, 2019), ■ OpenWebText (38G, 2023), ■ Wikipedia (21G, 2023)
- ☰ the Pile - StackExchange (41G, 2020)
- 📖 BookCorpus (5G, 2015), 📖 Gutenberg (-, 2021), 📖 CC-Stories-R (31G, 2019), 📖 CC-NEWS (78G, 2019), 📖 REALNEWS (120G, 2019)
- 📄 the Pile - ArXiv (72G, 2020), 📄 the Pile - PubMed Abstracts (25G, 2020)
- 🗄️ BigQuery (-, 2023), the Pile - GitHub (61G, 2020)



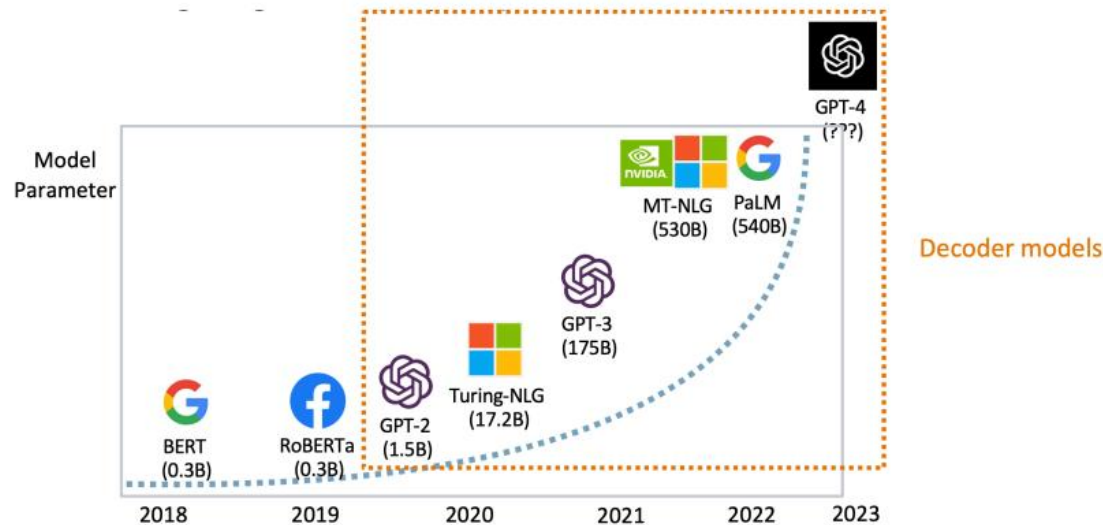
Fine-tuning PLMs

- The pre-training stage lets language models learn generic representations and knowledge from the corpus, but they are not fine-tuned on any form of user tasks.
- To adapt language models to a specific downstream task, we usually use task-specific datasets for fine-tuning



Scaling up Language Models

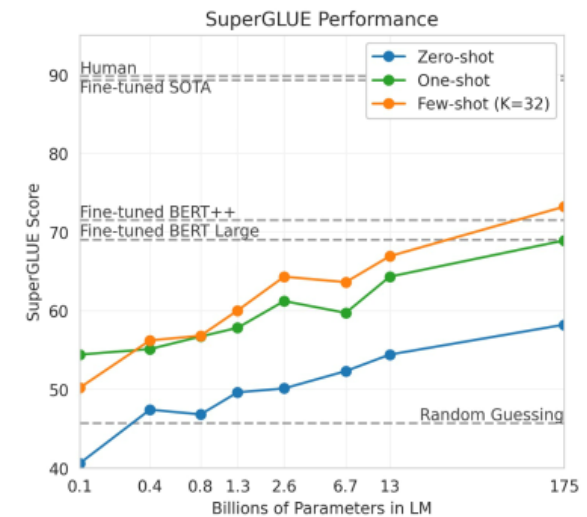
- Model size in terms of trainable parameters has increased significantly over the years, constantly increasing performance, especially for text generation models.
- The name **large language model** is *usually* applied to language models with more than 1B parameters (mostly decoder-only)



Performance of Zero-/Few-shot GPT 3

	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	89.0	91.0	96.9	93.9	94.8	92.5
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	76.1	93.8	62.3	88.2	92.5	93.3
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1



See you next week!

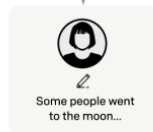
- Next time: Post-training and Efficient Adaptation
 - Instruction tuning
 - Reinforcement learning from human feedback
 - Efficient fine-tuning of LLMs

Step 1
Collect demonstration data, and train a supervised policy.

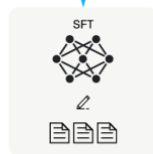
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.

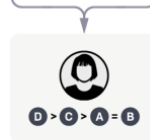


Step 2
Collect comparison data, and train a reward model.

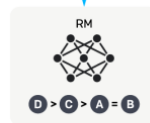
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

