Structure-Guided, Theme-Based Knowledge Discovery with Large Language Models

JIAWEI HAN, MICHAEL AIKEN CHAIR PROFESSOR
SCHOOL OF COMPUTING AND DATA SCIENCE SCIENCE
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

JUNE 29, 2026

Class Information

- **Instructor:**
  - Jiawei Han, Michael Aiken Chair Professor
    School of Computing and Data Science, University of Illinois Urbana-Champaign

- **Prerequisites**
  - General knowledge on LLM, NLP, data mining, AI, machine/deep learning

- **Course website**
  - Lecture Materials – DeepLearn 2026

- **Reference materials**
  - **Recent research papers (listed at the end of each chapter)**

Course Coverage

- A Brief Introduction to Large Language Models (30 min)
- Retrieval and Retrieval Augment Generation (RAG) (70 min)
- Reasoning with Structures for LLMs (70 min)
- LLM Agents and Agent Memory Technology (90 min)
- Look Forward to the Future and Q&A (10 min)

A Brief Introduction to Large Language Models

JIAWEI HAN
SIEBEL SCHOOL OF COMPUTING AND DATA SCIENCE
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

JUNE 29, 2026

Outline

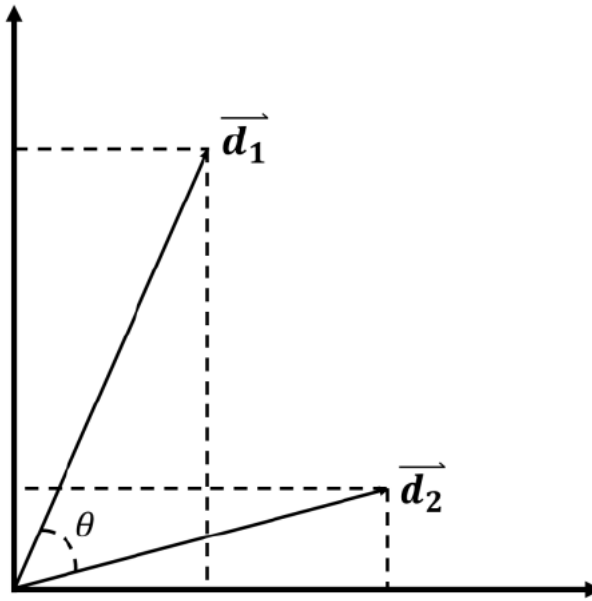
- From Static Text Embeddings to Transformer Architecture
- Large Language Models
- LLM Implementation: Data Preparation, Pretraining, Fine-Tuning, and Optimization

Large Language Models

- Large Language Models (LLMs) are artificial intelligence systems trained on vast amounts of text data to understand and generate human language
  - They *learn statistical patterns* in language and can *perform tasks such as question answering, summarization, translation, coding, reasoning, and content generation*
- Modern LLMs are based primarily on the **Transformer** architecture, introduced by Ashish Vaswani et al. *"Attention Is All You Need."* (2017)
- Rapid advances in **model scaling**, **alignment techniques**, and **multimodal learning** have transformed LLMs into versatile systems capable of performing *a wide range of language and reasoning tasks*, making them a **central technology in modern AI**
- **Current Research Focuses**: (Red ones are the focus of this lecture)
  - Retrieval-Augmented Generation (RAG)
  - Long-context modeling
  - Reasoning and planning capabilities
  - Agent systems and tool use
  - Efficient and smaller models
  - Safety, alignment, and interpretability
  - Multimodal learning

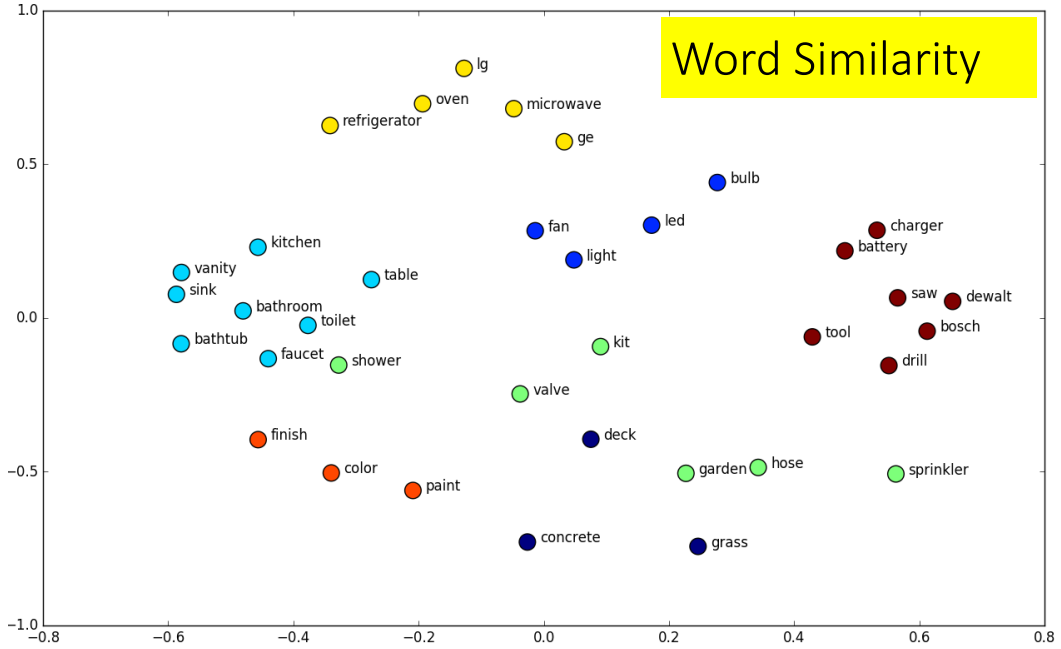
Early Text Representation: Symbolic Representation

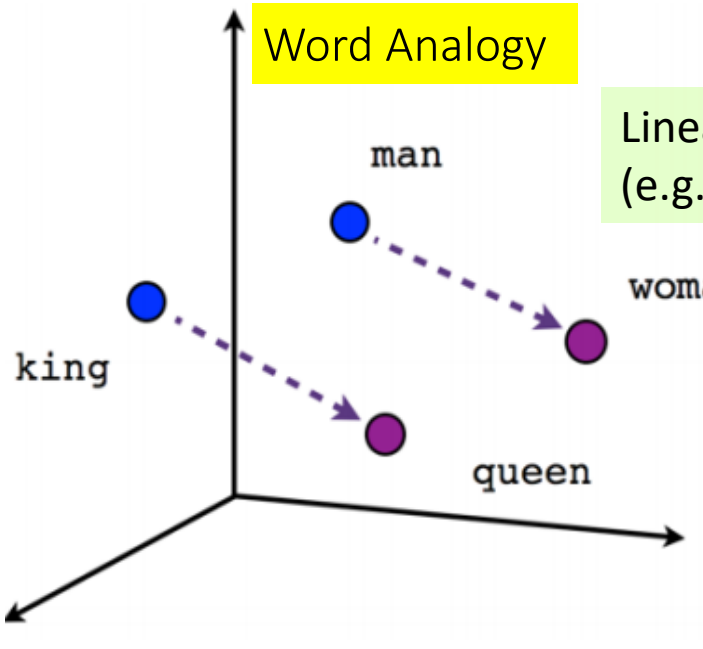
- Symbol-Based Text Representations
  - One-to-one correspondence between text units and representation elements
    - One-hot vector, e.g., “dogs” = [1, 0, 0, 0, 0]; “cats” = [0, 1, 0, 0, 0]; …
  - Bag-of-words: Describe a doc according to which words are present, ignoring word ordering
- Typical measures: TFIDF (and its variant BM25) [still used in many search engines]
  - Term frequency (TF): $tf_i = N(t_i, d)$ or take the log: $f_i = log\ (tf_i + 1)$
  - Inverted document frequency (IDF): $idf_i = \log (N/df_i)$
  - TF-IDF: $tf\_idf_i = tf_i$ X idf_i
    - *Term i appears frequently in the current document but infrequent in other docs*
- Issues:
  - Curse of dimensionality!
  - Vectors do not reflect semantic similarity



Distributed Text Representations: Embedding

- Unsupervised/self-supervised learning of text representations—No annotation needed
- Embed into lower-dimensional space
  - Address "curse of dimensionality" (comparing with one-hot vector representation)
- Word embedding captures useful properties of word semantics
  - Word similarity: Words with similar meanings are embedded closer





Linear relationships between words
(e.g., king − queen = man−woman)

Typical embedding methods:
- Word2Vec (Google)
- GloVe (Stanford)
- fastText (Facebook)
- ……

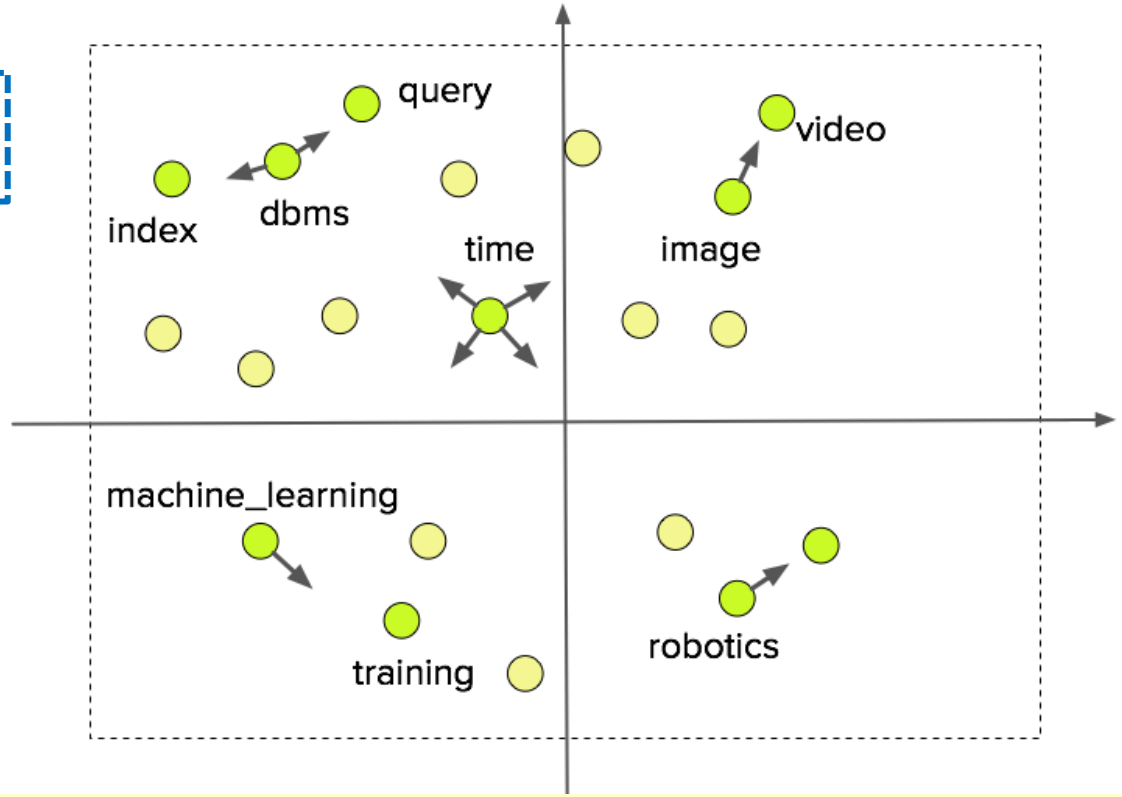
Word2Vec (Mikolov, et al., 2013)

- Many text embeddings are learned in the Euclidean space (without constraints on vectors)
- Word2Vec maximizes the probability of observing a word based on its local contexts
- As a result, semantically similar terms are more likely to have close embeddings

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

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

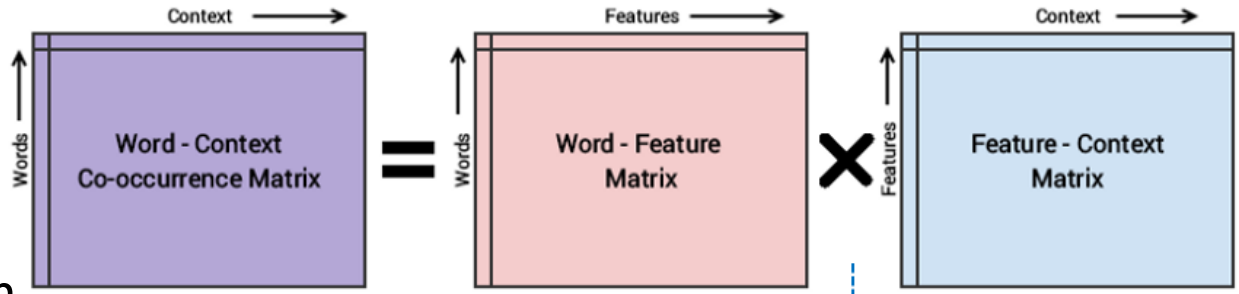
$$p(w_O|w_I)=\frac{\exp\left({v'_{w_O}}^{\top}v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^{\top}v_{w_I}\right)}$$



Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.

Other Word Embedding Models

- GloVe (Stanford, 2014)
  - GloVe factorizes a global co-occurrence matrix derived from the entire corpus
  - Low-dim. reps by solving a least-squares problem to "recover" the co-occurrence matrix
- fastText (Facebook, 2014)
  - Incorporating subword info into word embedding
- Spherical Text Embedding (JoSE, UIUC, 2019)
  - Previous text embeddings are trained in the Euclidean space but used in spherical space (cosine or directional similarity)
  - JoSE: Joint spherical embedding: spherical embedding + integrating local and global contexts



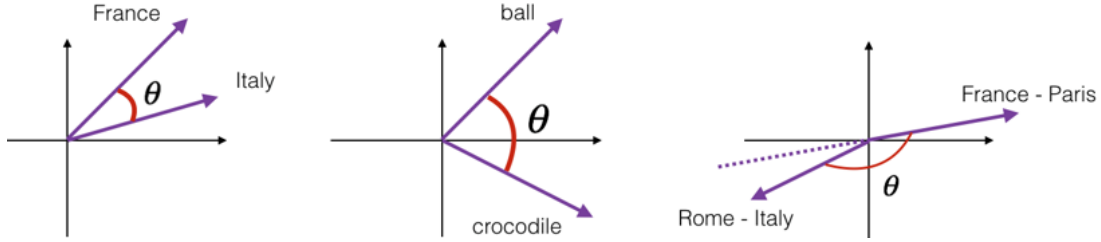
Sparse, high dimensional

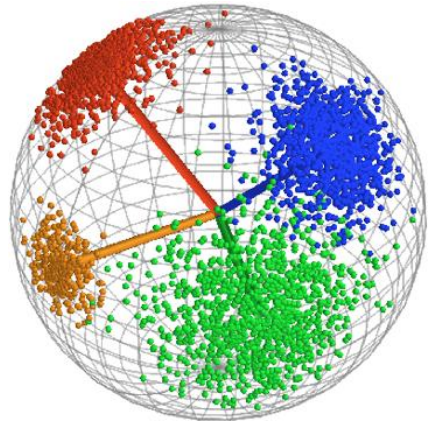
Low dimensional representation

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

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

N-gram embedding



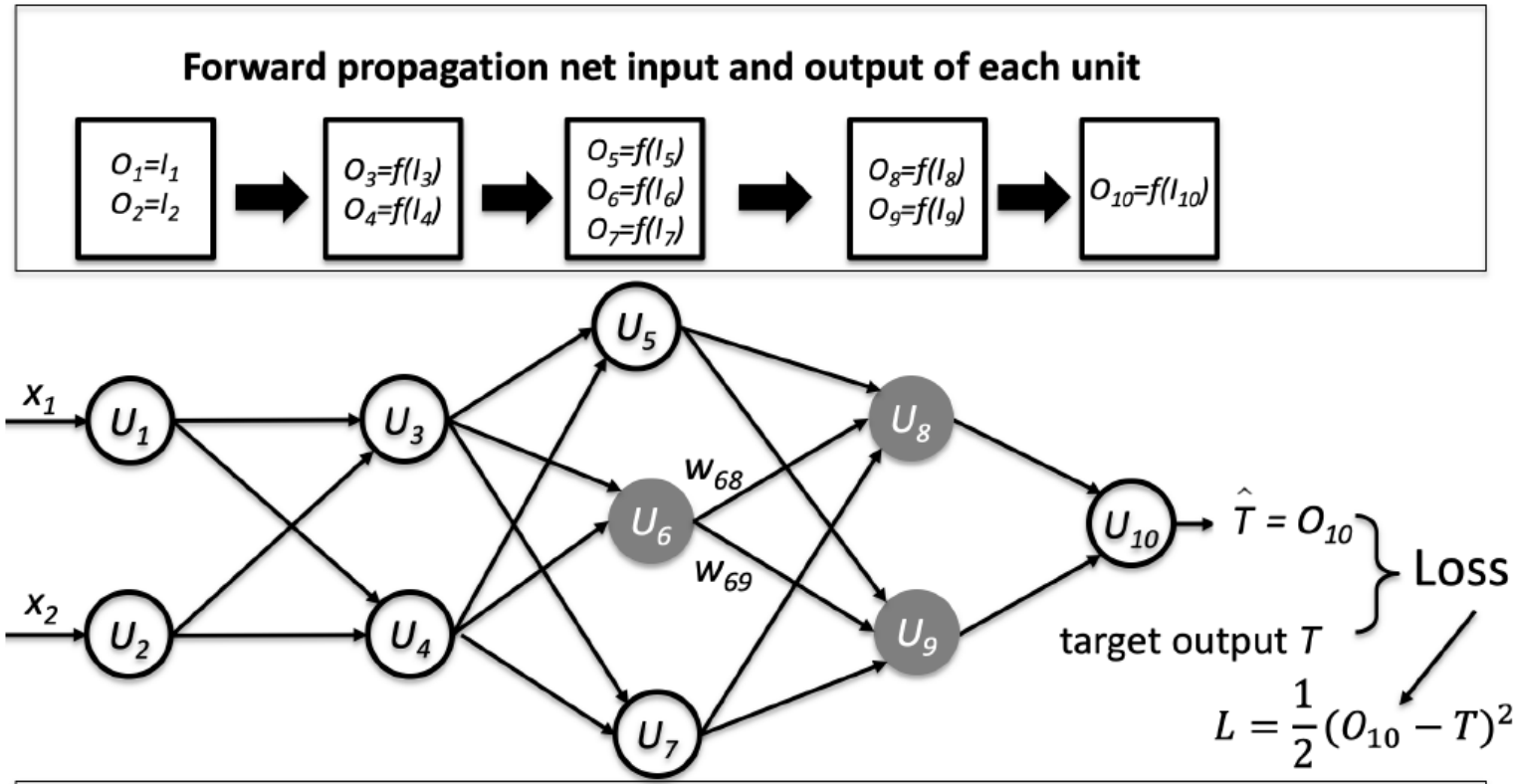


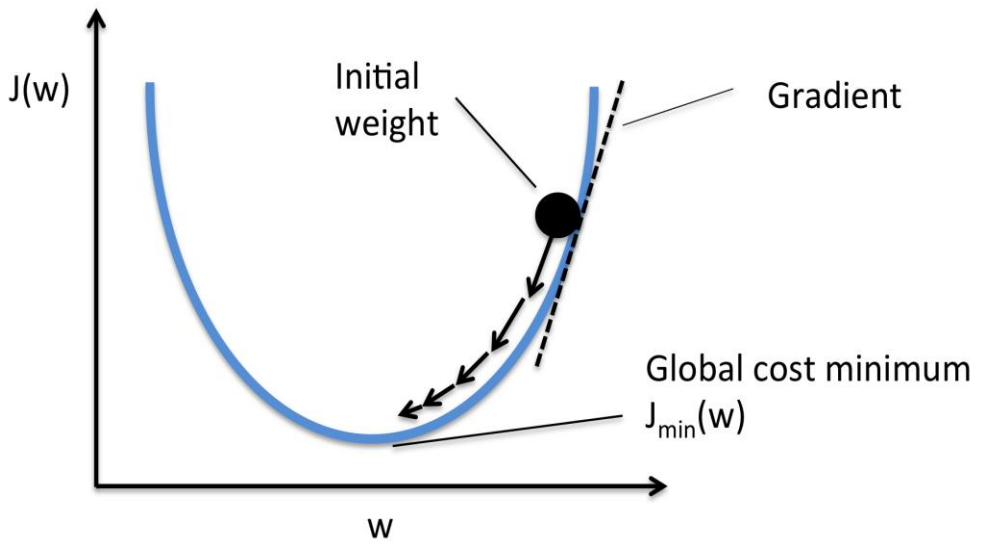
Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. TACL

Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L.M., & Han, J. (2019). Spherical Text Embedding. NeurIPS.

Neural Network: Backpropagation & Gradient Computation





$$\theta_i = \theta_i - \alpha \cdot \frac{\partial}{\partial \theta_i} J_\theta$$

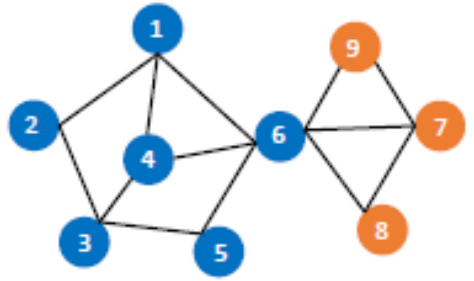
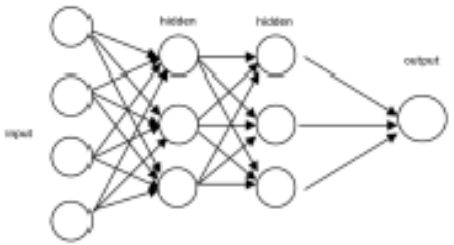
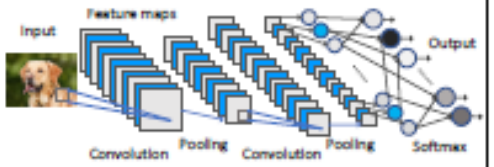
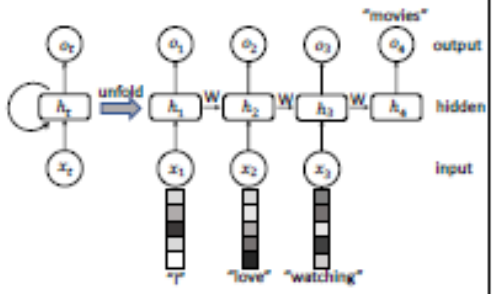
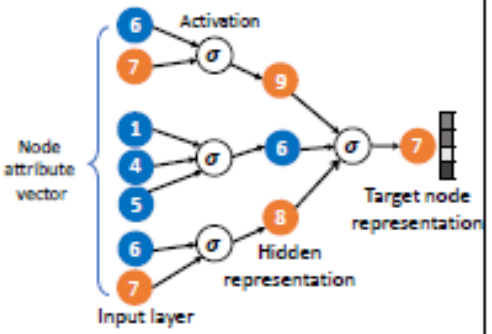
learning rate – "step size" of the optimization

adapted from https://www.upgrad.com/blog/machine-learning-interview-questions-answers-ii/

Backpropagation in neural networks

Ack. Figures from Han, Pei, Tong, "Data Mining: Concepts and Techniques, Morgan Kaufmann, 2022

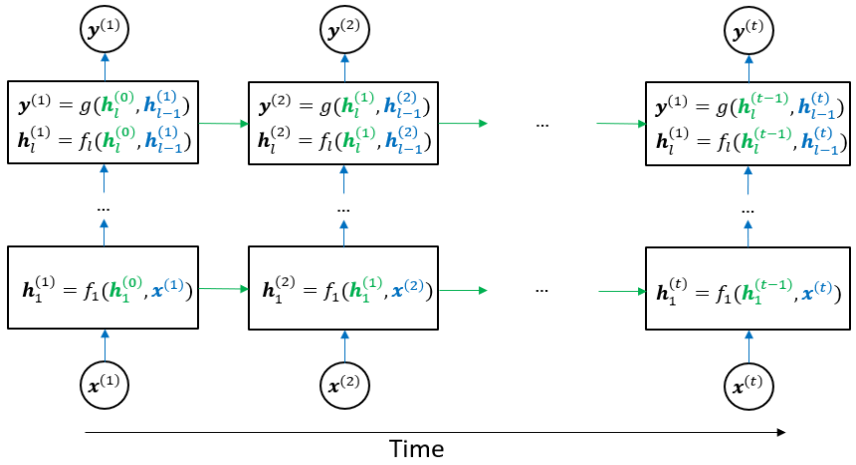
Typical Deep Learning Architectures

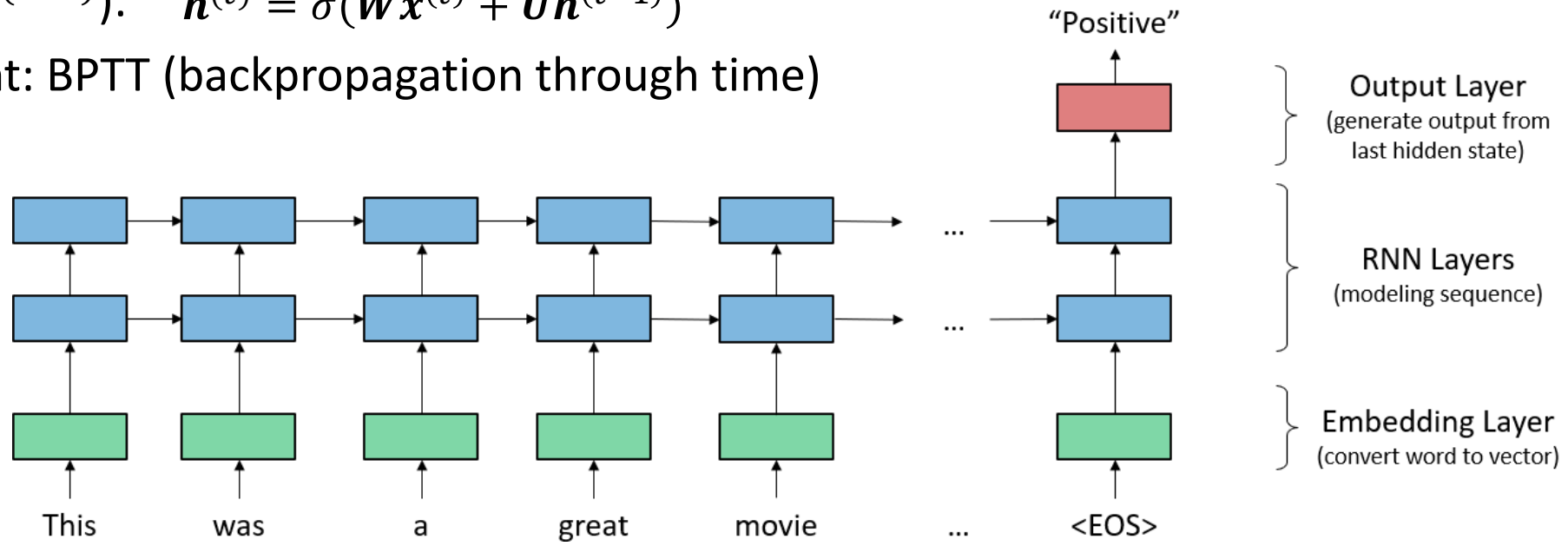
Data type	Multi-dimensional	Grid	Sequence	Graph
	Features: credit rating, account balance $x = (4.5, 500, 3, 5)$ #deposits, #withdraws		x = "I love watching movies."	
DL Architecture	**Feed-forward Network**	**CNN**	**RNN**	**GNN**
				

Ack. Figures from Han, Pei, Tong, "Data Mining: Concepts and Techniques, Morgan Kaufmann, 2022

Recurrent Neural Networks

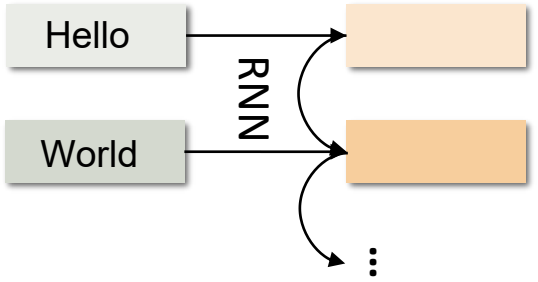
- **Recurrent Neural Networks (RNN) for modeling sequences**
  - At each time step, the input and the previous hidden state are fed into the network
- **Long-term dependencies**: Use hidden states to preserve sequential information
- Output at each time *t* is based on the current input ($\boldsymbol{x}^{(t)}$) and the previous state ($\boldsymbol{h}^{(t-1)}$): $\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{W}\boldsymbol{x}^{(t)} + \boldsymbol{U}\boldsymbol{h}^{(t-1)})$
  - Compute a gradient: BPTT (backpropagation through time)

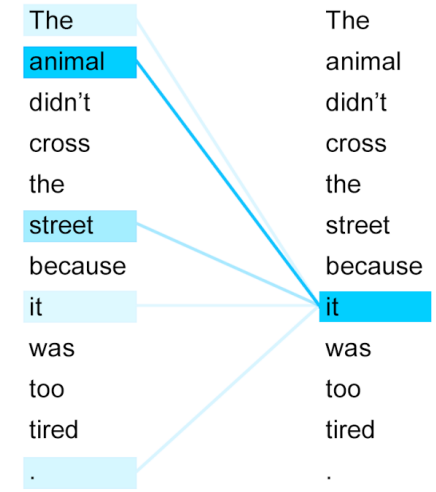




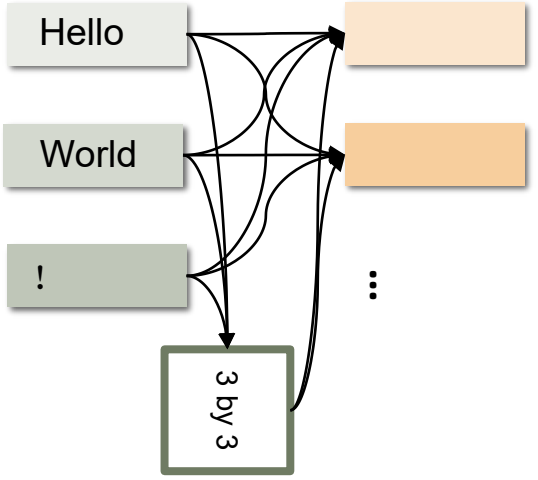
The Transformer Revolution

- Problem with RNNs:
  - The output vector at timestamp *t* depends on the output at timestamp *t – 1*
  - It presumes an order for inputs and prevents parallelization for scalable training
- Solution: "*Attention is all you need*"
  - Explicitly calculate dependencies & handle sequential inputs without "recurrent"
  - The independent computation of outputs allows parallelization
- Heart of Transformer: (self-)attention mechanism: It captures long-term contextual information much more effectively using GPUs than the recurrence and convolution mechanisms

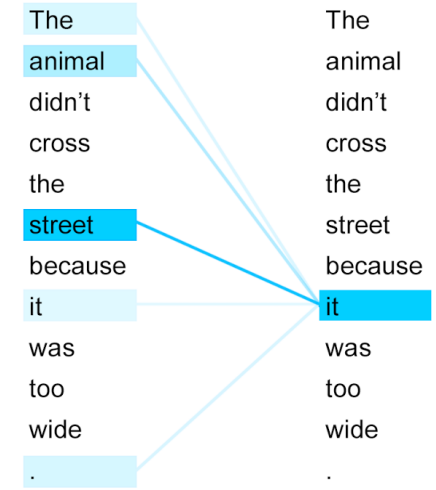




Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "***Attention is All You Need***", NIPS (2017)



More reading: https://www.programmersought.com/article/10956032922/

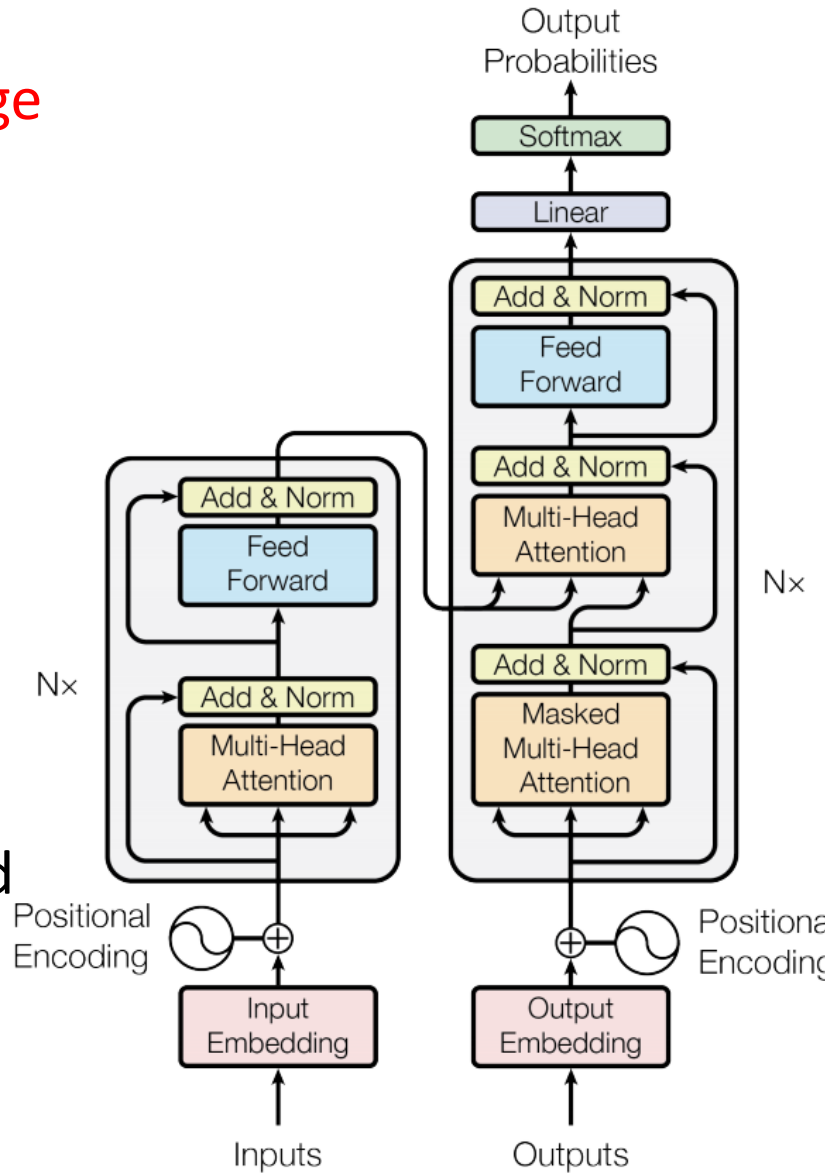


Reading material: The Illustrated Transformer

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformer + Language Model

- The Transformer architecture enables scalable training of large models, but where can we find big data to train the models?
- Language Model: learning from unlabeled text data
  - Task: Given a sentence with randomly masked words, the model is asked to read the seen words (encoder), and predict the masked words (decoder)
    - Ex. "The man [MASK] to the store."
  - All existing text corpora can be our training data!
  - The learned (pretrained) encoder/decoder serves as a good initialization of models for downstream applications
- Reading materials
  - Transformer → BERT → GPT-2



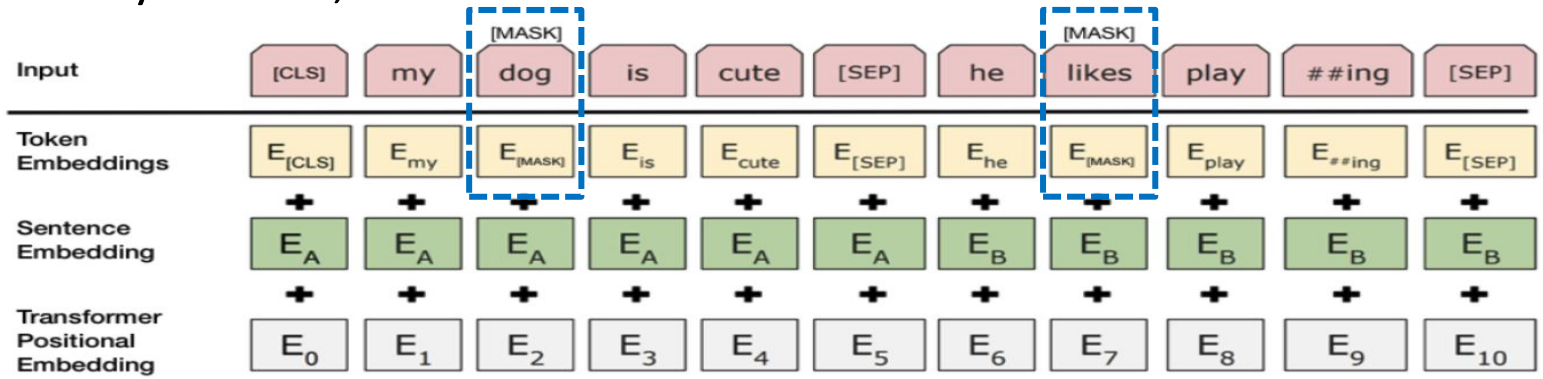
Ack. Figures from Vaswani et al., *NIPS* 2017

Outline

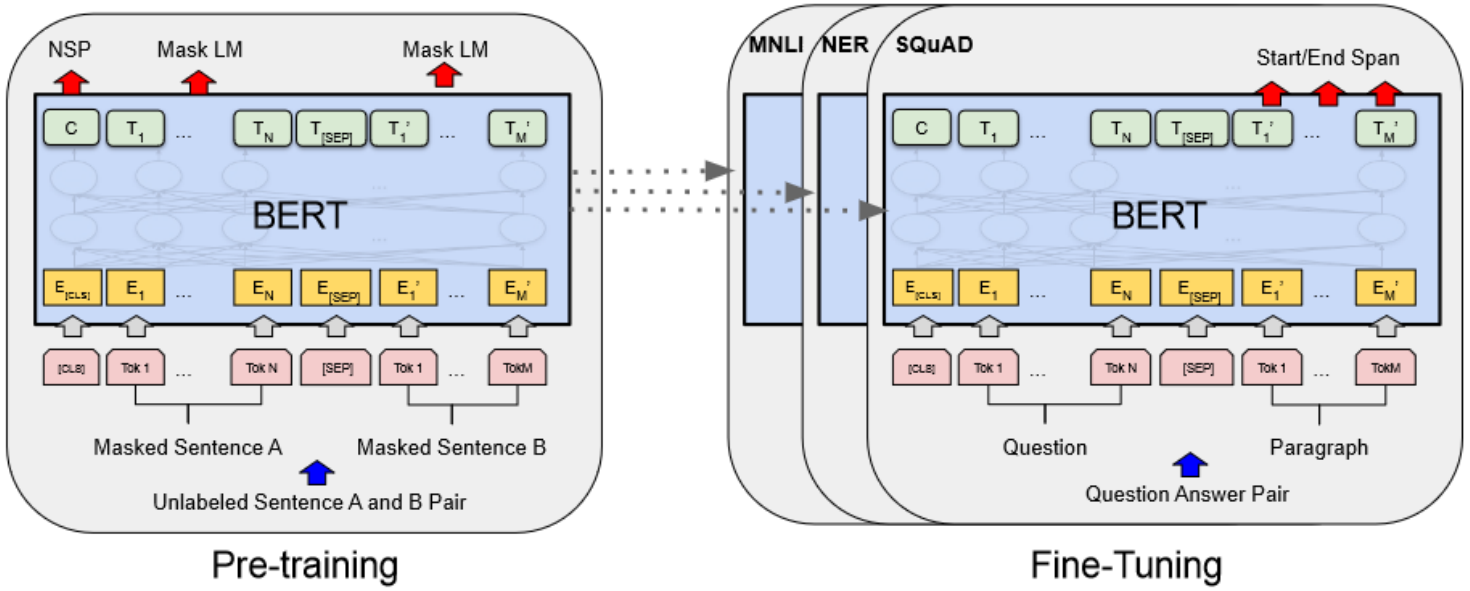
- From Static Text Embeddings to Transformer Architecture
- Large Language Models 
- LLM Implementation: Data Preparation, Pretraining, Fine-Tuning, and Optimization

BERT, Its Pretraining and Fine-Tuning

- Masked LM: With 15% words randomly masked, the model learns bidirectional contextual information to predict the masked words



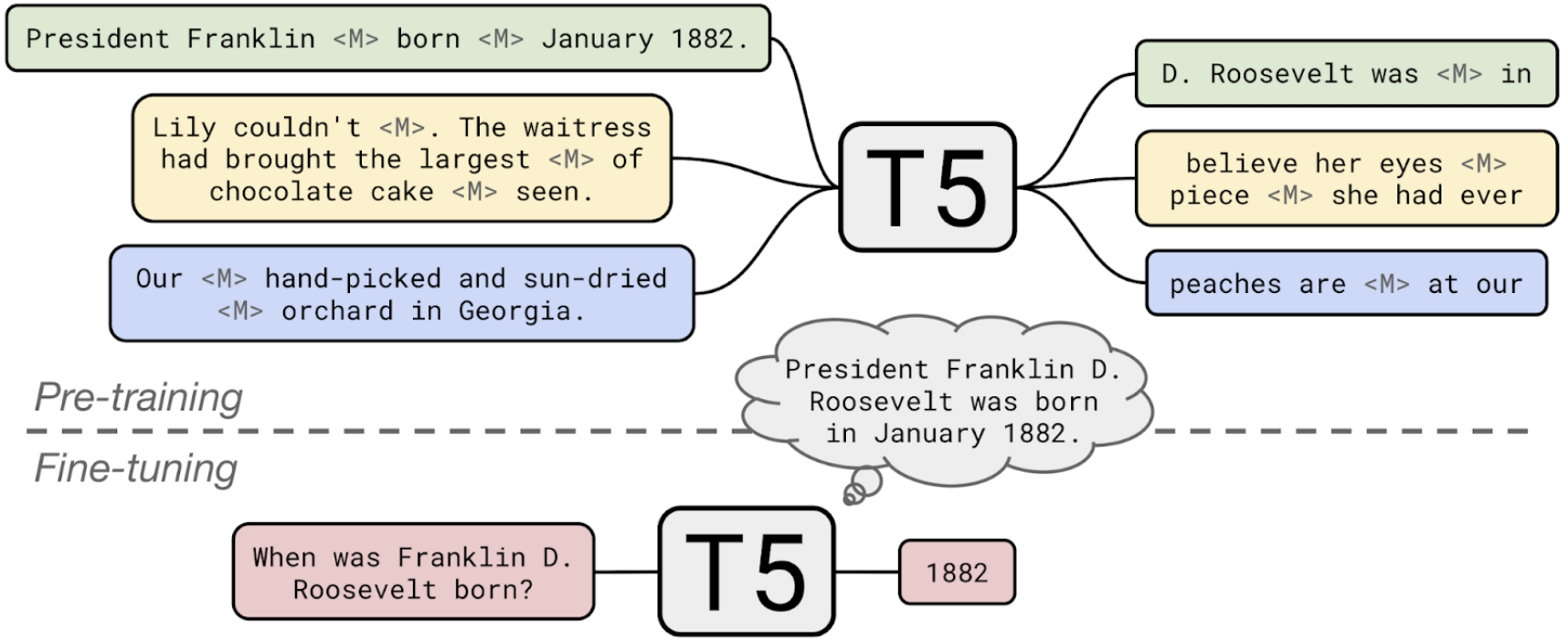
- Next Sentence Prediction: BERT receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document



Jacob Devlin, et al. "BERT: Pre-training of deep bidirectional transformers for language understanding." *NAACL* (2019)

T5

- T5: **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer
- Pretraining: Mask out spans of texts; generate the original spans
- Fine-Tuning: Convert every task into a sequence-to-sequence generation problem



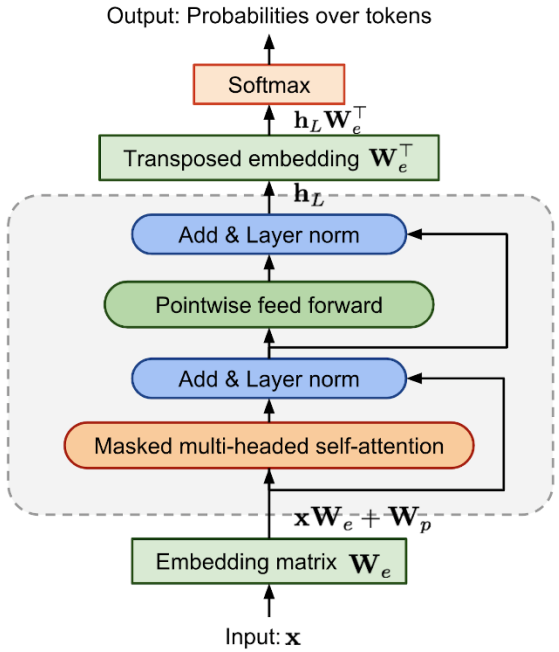
Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.

GPT-Style Pre-Training: Introduction

- GPT (Generative Pre-trained Transformers): GPT [1], GPT-2 [2], GPT-3 [3]
- Leverage unidirectional context (usually left-to-right) for next token prediction (i.e., language modeling)

k previous tokens as context

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

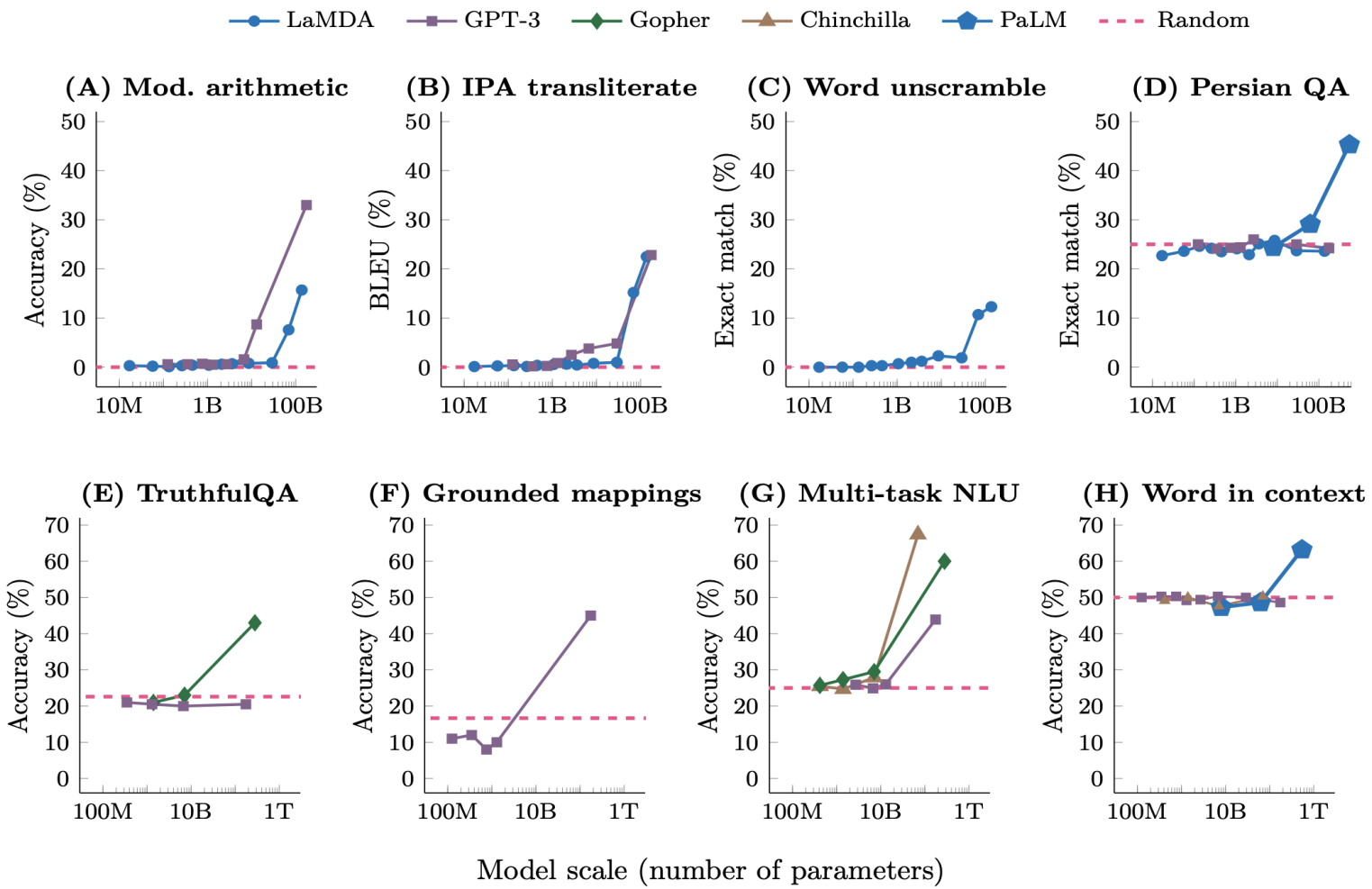


- The Transformer uses **unidirectional** attention masks (i.e., every token can only attend to previous tokens)

[1] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog
[2] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
[3] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.

Why Large Language Models (LLMs)?

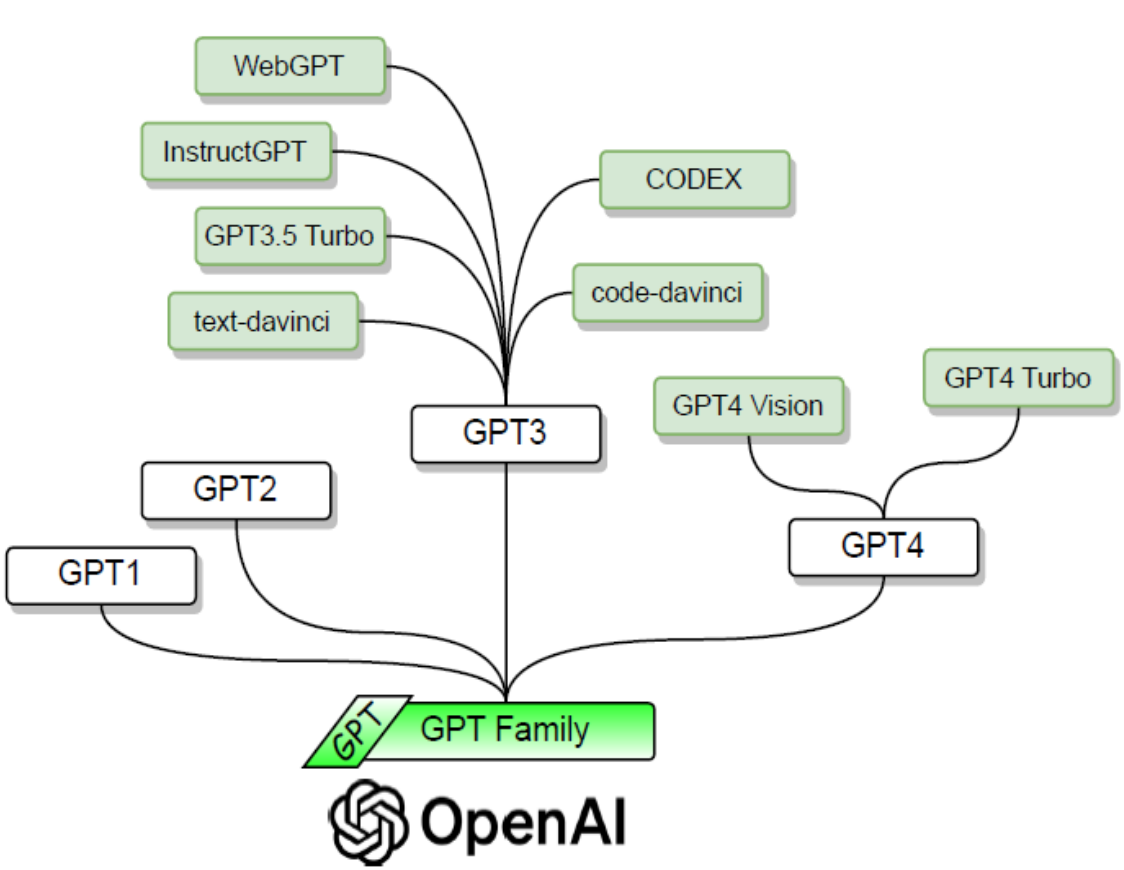
- Scaling up language models induces **emergent abilities**
- "Emergent": not present in smaller models but in larger models

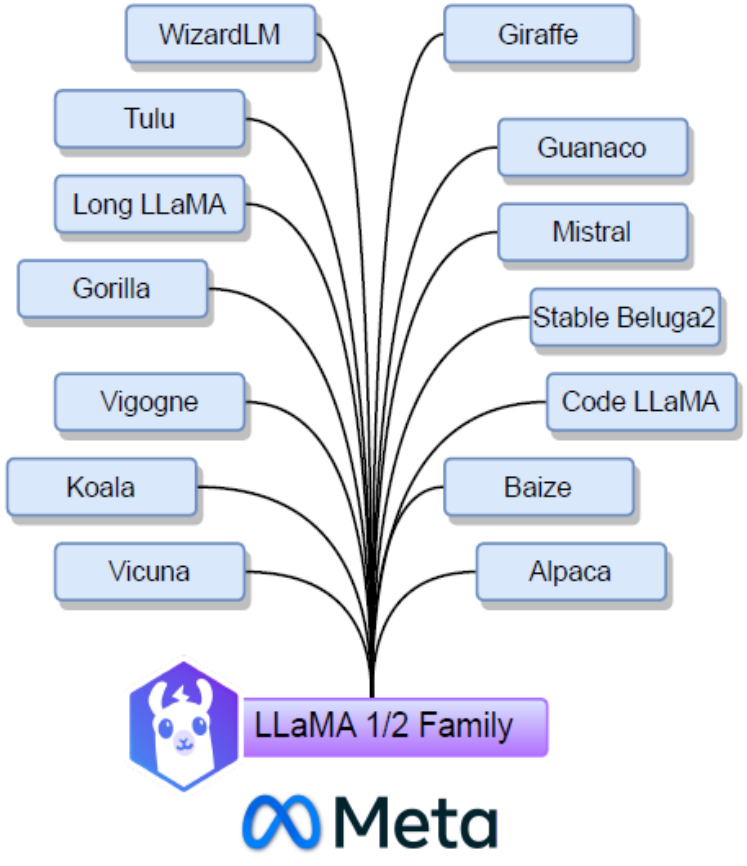


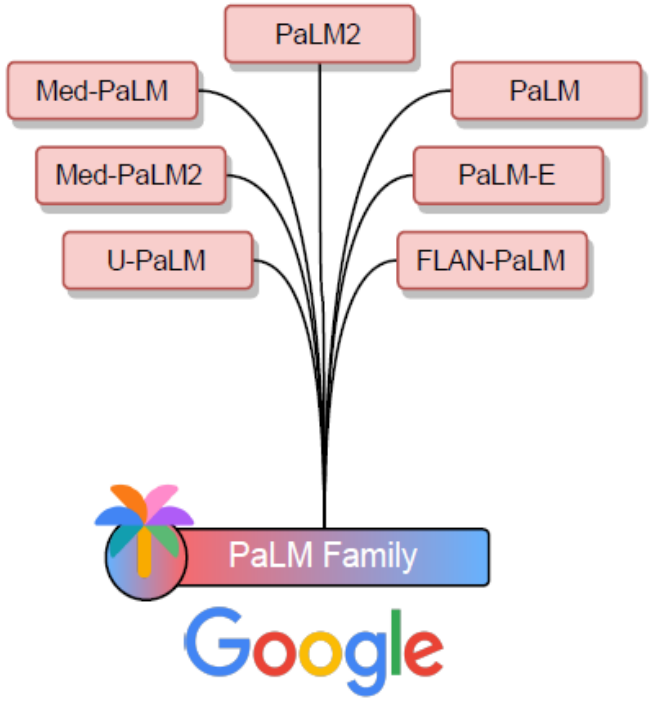
Emergent ability for few-shot prompting: LMs have random performance until a certain scale, after which performance significantly increases well-above random

Wei, J., et al. (2022). Emergent Abilities of Large Language Models. TMLR.

Popular LLM Families







Courtesy of *Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, Jianfeng Gao, "Large Language Models: A Survey", https://arxiv.org/abs/2402.06196*

DeepSeek Family Models

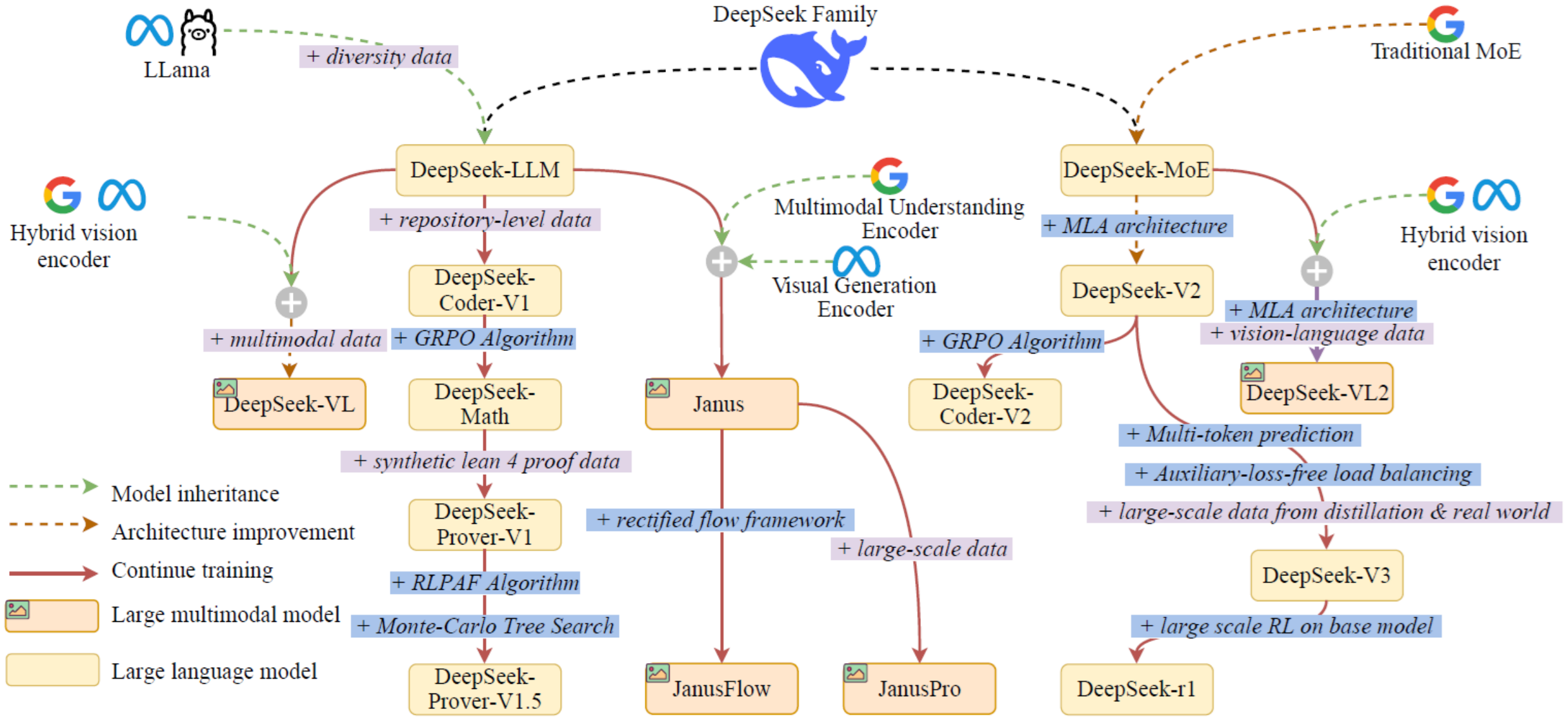


Fig. 1. The Evolution Overview of the DeepSeek family models. represents the data-related improvment on the training datasets. represents the algorithm/architecture-related improvement on the training processes.

Z. Deng, et al, “Exploring Deepseek: A Survey On Advances, Applications, Challenges And Future Directions”, IEEE/CAA J. Automatica Sinica, May 2025

Outline

- LLMs: Basic Concepts and Brief History
- From Static Text Embeddings to Transformer Architecture
- Large Language Models
- LLM Implementation: Data Preparation, Pretraining, Fine-Tuning, and Optimization

Building LLMs: Data Cleaning and Tokenization

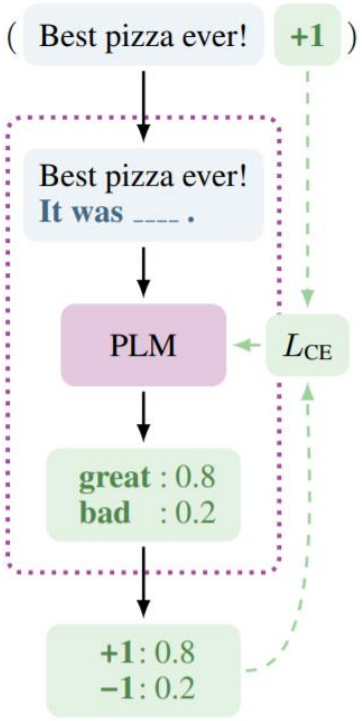
- **Data filtering**
  - *Removing noise*: Eliminate irrelevant or noisy data (e.g., false info) from the training data
  - *Handling outliers/anomalies*: to prevent them from disproportionately influencing the model
  - *Addressing imbalances*: Balancing the distribution of classes or categories to avoid biases
  - *Text preprocessing*: Cleaning & standardizing text data by removing stop words, punctuation, …
  - *Dealing with Ambiguities*: Resolving or excluding ambiguous or contradictory data
- **Deduplication**: Remove duplicate instances or repeated occurrences of the same data in a dataset
  - Methods: Compare entire data points or specific features to identify and eliminate duplicates
- **Tokenization**: Convert a sequence of text into smaller parts, known as tokens
  - Popular tokenizers for LLMs are based on sub-words
  - Methods: BytePairEncoding, WordPieceEncoding, SentencePieceEncoding

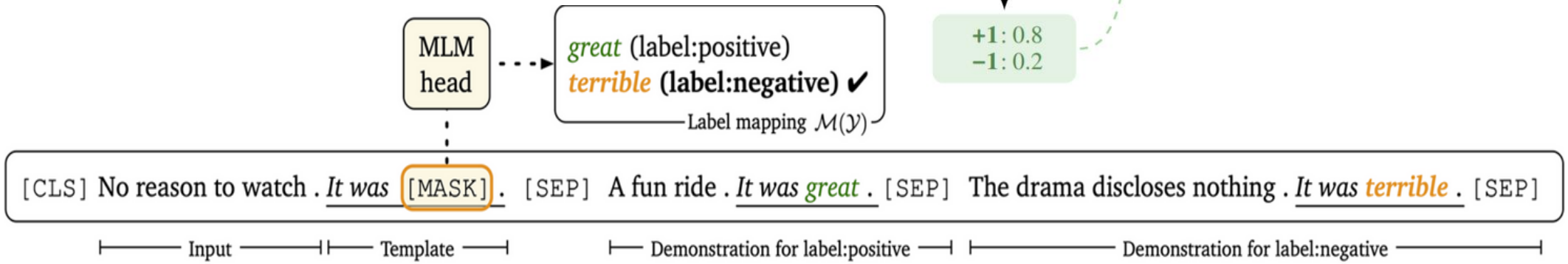
Building LLMs: Fine-Tuning & Instruction Tuning

- **(Supervised) Fine-tuning (SFT):** Adjust the pre-trained parameters to optimize performance for specific text mining and NLP tasks, ranging from classification to generation
  - *Early language models (e.g.,* BERT): Fine-tuned to a specific task with labeled data
  - *More recent LLMs* no longer require fine-tuning, but they can still benefit from task or data-specific fine-tuning
    - Ex.: GPT-3.5 Turbo model can outperform GPT-4 when fine-tuned with task-specific data
- **Multi-task fine-tuning**: Improve the model's ability to generalize *across different types of data*
  - Fine-tuning to one or more tasks to improve results and reduce the complexity of prompt engineering, adapt to *new or proprietary data*
- **Instruction tuning**: Align the responses to *human expectations when given instructions (prompts)*
  - Natural Instructions include not only the task definition but other components (e.g., *positive/negative examples or things to avoid*)
  - **Self-Instruct**: Generate instructions, input, and output samples *from a language model*, then filter invalid or similar ones before using them to fine tune the original model

Prompt-Based Fine-Tuning

- Task descriptions are created to convert training examples to cloze questions
- Highly resemble the pre-training tasks (MLM) so that pre-training knowledge could be better leveraged
- Better than standard fine-tuning especially for few-shot settings





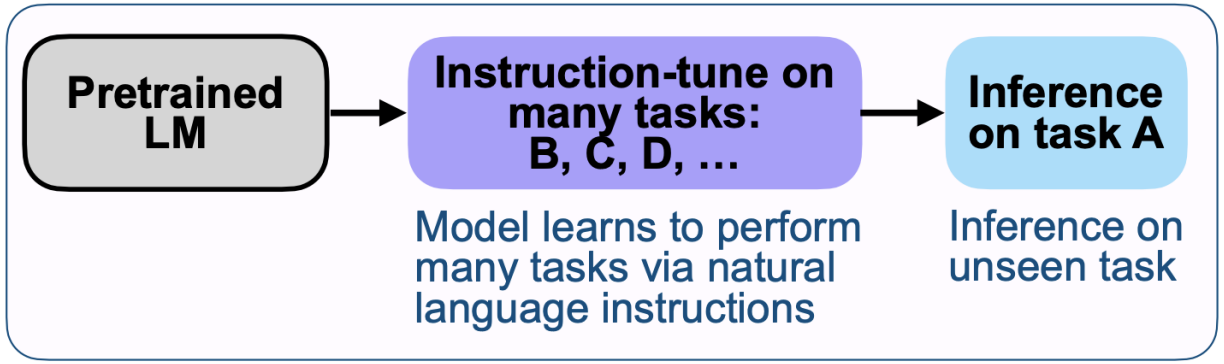
Schick, T., & Schütze, H. (2021). Exploiting cloze questions for few shot text classification and natural language inference. EACL.
Le Scao, T., & Rush, A. M. (2021). How many data points is a prompt worth? NAACL.

Building LLMs: Human Alignment

- Alignment: Steering AI systems towards human goals, preferences, and principles
  - A pretrained LLM may generate contents that are toxic, harmful, misleading and biased
- To align LLM outputs with human values, the refinement stage begins with *instruction tuning*, a specialized form of supervised fine-tuning
- Two popular approaches: RLHF (human feedback) and RLAIF (AI feedback)
  - **RLHF** (reinforcement learning from human feedback): Learn alignment from human feedback, using a *reward model*, which, *after being tuned, rate different outputs and score them according to their alignment preferences given by humans*
  - **RLAIF** (reinforcement learning from AI feedback): directly connect a pretrained and well-aligned model to the LLM and help it to learn from larger and more aligned models
- **Direct Preference Optimization** (DPO) (Rafailov et al. 2023): directly optimize for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL
- **KTO (Kahneman-Tversky Optimization)** (Ethayarajh et al. 2024): Use a far more abundant kind of data, making it much easier to use in the real world

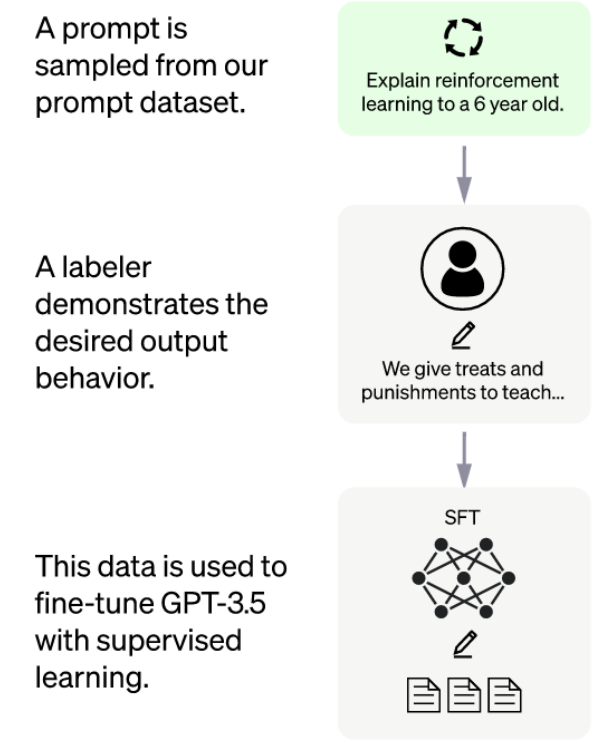
Human Alignment: Instruction Tuning

- Prompt-based fine-tuning on various tasks/formats => generalization to unseen tasks/formats
- Applicable to build chatbots (e.g., ChatGPT) by tuning language models on dialogue input-response pairs
- Following instruction tuning, Reinforcement Learning from Human Feedback (RLHF) tailors the models' behavior



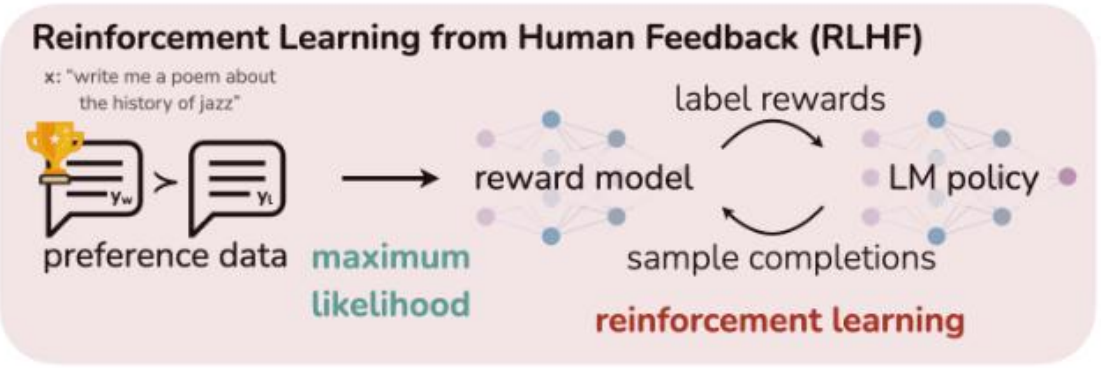
Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., & Le, Q.V. (2022). Finetuned Language Models Are Zero-Shot Learners. ICLR

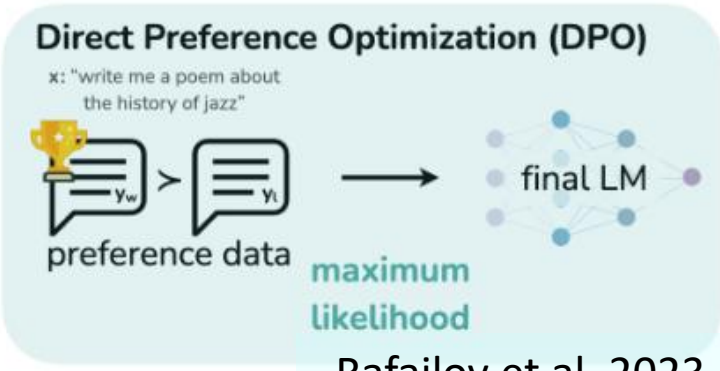
Through RLHF, employing algorithms like Proximal Policy Optimization (PPO) and Direct Preference Optimization (DPO), models are trained based on human preferences, steering them toward generating responses that are ethical, relevant, and contextually appropriate, thereby ensuring their outputs are closely aligned with human expectations



Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., & Levy, O. (2023). LIMA: Less Is More for Alignment.

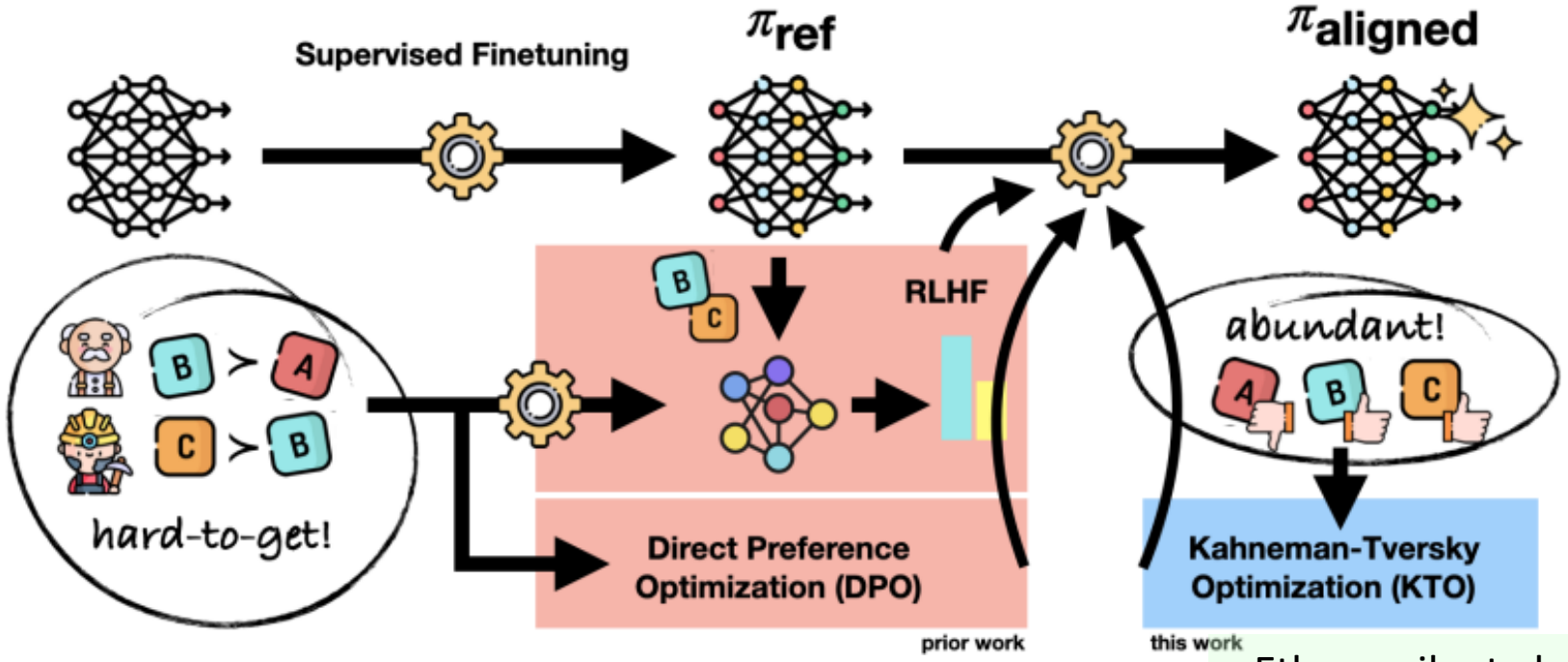
Alternative LLM Alignment Methods: DPO and KTO





Rafailov et al. 2023

DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL



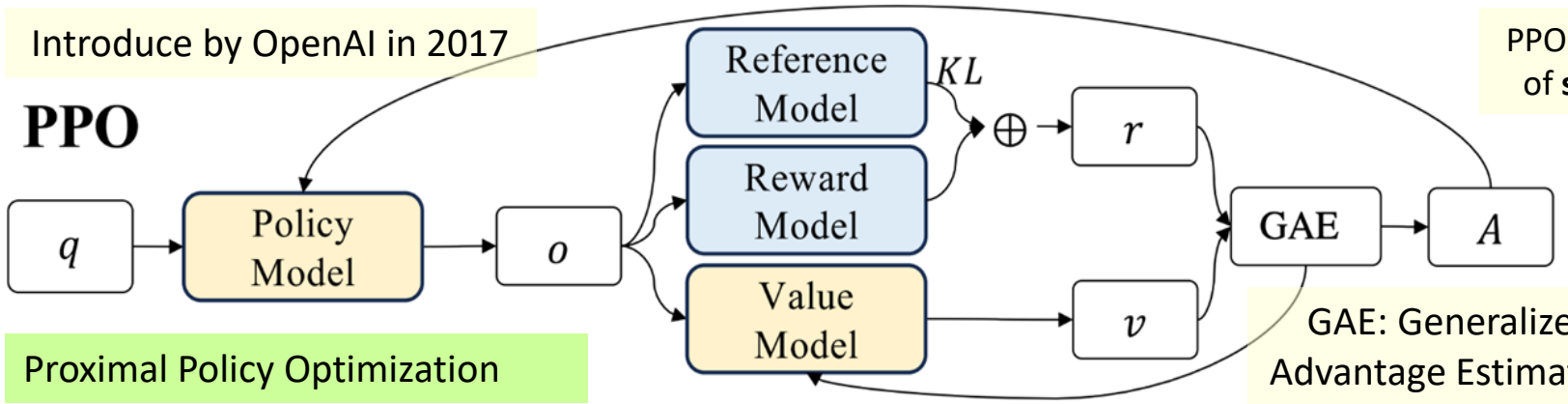
Ethayarajh et al. 2024

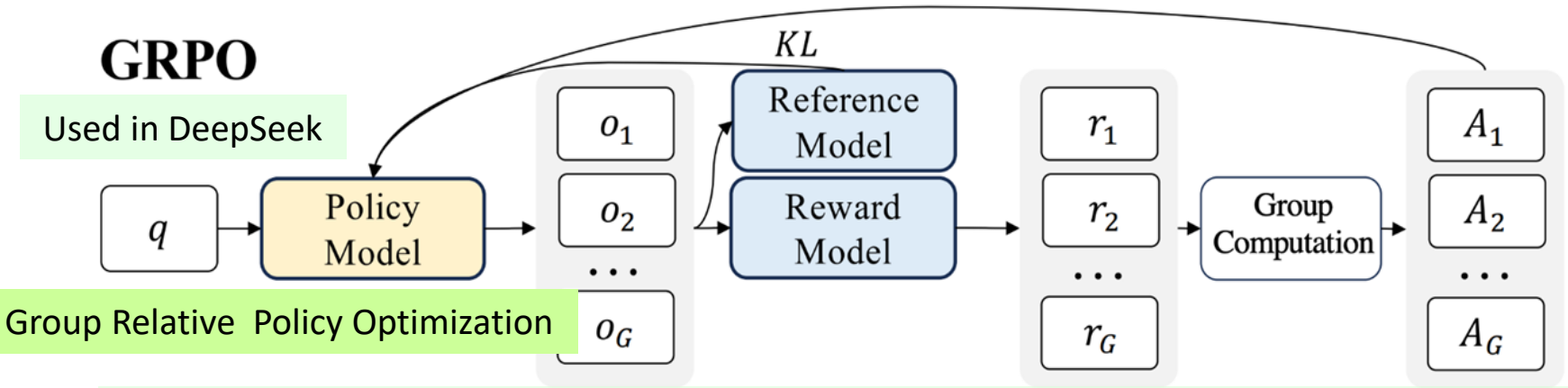
The paired preferences that existing approaches need are hard-to-obtain. KTO uses a far more abundant kind of data (e.g., whether y is desirable or undesirable), making it much easier to use in the real world.

RL Algorithms: PPO vs. GRPO

Introduce by OpenAI in 2017

PPO offers a compelling mix of **stability** and **simplicity**





Policy Model is the LLM being trained

Reference Model is the untrained LLM itself

Reward Model is the reward function to compute the (rule-based) reward

GRPO avoids the challenges of learning an accurate value model and saves roughly half the memory/computation since we don't maintain extra model parameters for the critic. This makes RL training simpler and more feasible in memory-constrained settings.

Assign higher probability to the better-than-average actions and lower prob. to the worse ones

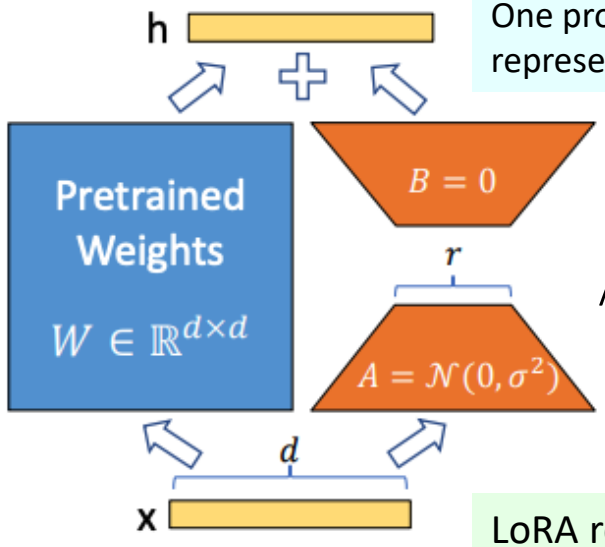
(PPO) Schulman, et al. "Proximal policy optimization algorithms," arXiv 2017
(GRPO) Shao, et al. "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," arXiv:2402.03300

Parameter-Efficient Fine-Tuning

- **Parameter-efficient fine-tuning** (e.g., Low-Rank Adaptation (LoRA)): Minimizes computational costs while maintaining the model performance
  - **Why Parameter-efficient fine-tuning?**
    - Fine-tuning updates all PLM parameters at the same time
    - But large PLMs can have an enormous number of parameters that are costly to optimize
  - **Parameter-efficient fine-tuning:** Optimize only a small set of parameters in PLMs while still achieving comparable performance to fine-tuning
  - A few strategies:
    - **Adapter**: Insert small bottleneck modules and only update adapter + layer norm parameters
    - **Prefix Tuning**: Prepend tunable prefix vectors to every Transformer layer and keep other parameters unchanged
    - **Low-Rank Adaptation (LoRA)**: Use trainable low-rank matrices to approximate weight updates

Low-Rank Adaptation (LoRA)

- **Low-Rank Adaptation (LoR):** A popular and lightweight training technique that significantly reduces the number of trainable parameters
- **Insight:** The difference between the fine-tuned weights for a specialized task and the initial pre-trained weights often exhibits "low intrinsic rank" (***it can be approximated well by a low rank matrix***)
- **Inject trainable low-rank matrices into transformer layers to approximate the weight updates**
  - Since low-rank matrices have far less parameters than full-rank ones, training them is much more efficient than standard fine-tuning



One property of low-rank matrices: They can be represented as the product of two smaller matrices

$$W_0 + \Delta W = W_0 + BA$$

A and B are low-rank matrices

$W_0 \in R^{d \times k}$, $B \in R^{d \times r}$, $A \in R^{r \times k}$, and $\text{rank}(r) \ll \min(d, k)$.

Training with LoRA is much faster, memory-efficient, and produces smaller model weights (a few hundred MBs). By focusing on updating these two smaller matrices rather than the entire original weight matrix, computational efficiency can be substantially improved.

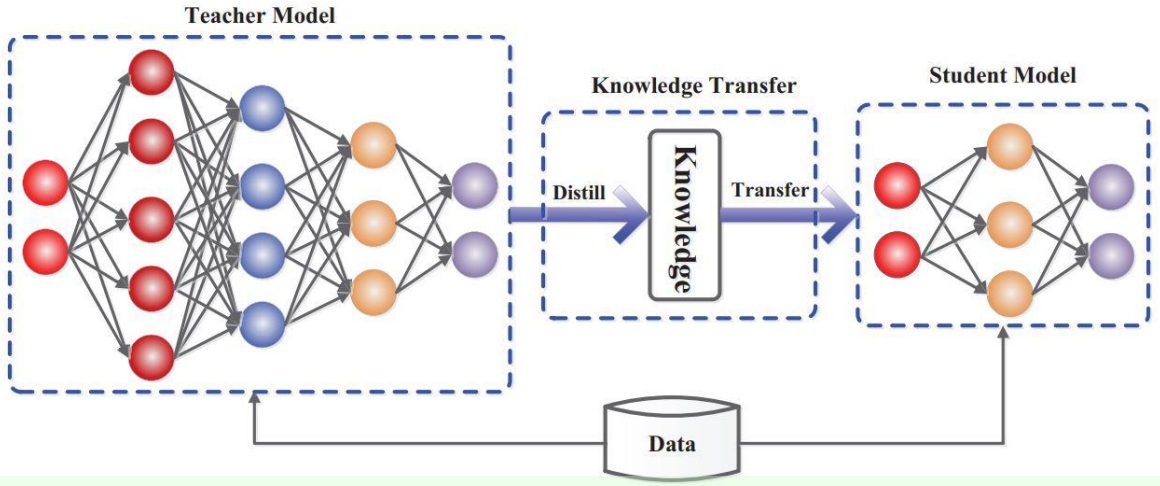
LoRA can be applied to any a subset of weight matrices in a neural network to reduce the number of trainable parameters.
In the Transformer architecture, there are four weight matrices in the self-attention module (W_q , W_k, W_v , W_o), and two in the MLP module. Most of the time, LoRA is focused on adapting the attention weights only for downstream tasks, and freezes the MLP modules, so they are not trained in downstream tasks both for simplicity and parameter-efficiency.

LoRA re-parameterization: Only A and B trained during this process

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. ICLR.

Building LLMs: Knowledge Distillation

- Knowledge distillation: The process of learning from a larger model
  - *Distill the knowledge of one or multiple models into a smaller one*
- Different forms of knowl. transfer: response distillation, feature distillation, and API distillation
  - **Response distillation**: Concerned only with the outputs of the teacher model, teach the student model how to exactly or at least similarly perform (in the sense of prediction) as the teacher
  - **Feature distillation**: not only uses the last layer but also intermediate layers to create a better inner representation for the student model: having a similar representation as the teacher model
  - **API distillation:** Uses an API to train smaller models
    - Note: OpenAI prohibits the usage of its API to create LLMs that later will be used to compete with it



A generic knowledge distillation framework with student and teacher

References (I)

- Abu-El-Haija, S., Perozzi, B., Al-Rfou', R., & Alemi, A.A. (2018). Watch Your Step: Learning Node Embeddings via Graph Attention. NeurIPS.
- Anil, R. et al. (2023). PaLM 2 Technical Report.
- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. Transactions of the Association for Computational Linguistics, 5, 135-146.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. NeurIPS.
- Chowdhery, A. et al. (2022) PaLM: Scaling Language Modeling with Pathways.
- Clark, K., Luong, M. T., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. ICLR.
- Zehang Deng, Wanlun Ma, Qing-Long Han, Wei Zhou, Xiaogang Zhu, Sheng Wen, and Yang Xiang, “Exploring Deepseek: A Survey On Advances, Applications, Challenges And Future Directions”, IEEE/CAA J. Automatica Sinica, May 2025
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.
- Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. ACL
- Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., ... & Gelly, S. (2019). Parameter-efficient transfer learning for NLP. ICML
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations. ICLR.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. ACL.

References (II)

- T. Lin, Y. Wang, Y. Wang, X. Liu, X. Qiu, A Survey of Transformers, https://arxiv.org/abs/2106.04554v2, 2021
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv:1907.11692.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. NIPS.
- Mikolov, T., Chen, K., Corrado, G.S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv: 1301.3781
- Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L.M., & Han, J. (2019). Spherical Text Embedding. NeurIPS.
- Meng, Y., Xiong, C., Bajaj, P., Bennett, P., Han, J., & Song, X. (2021). COCO-LM: Correcting and contrasting text sequences for language model pretraining. NeurIPS.
- Meng, Y., Xiong, C., Bajaj, P., Bennett, P. N., Han, J., & Song, X. (2022). Pretraining Text Encoders with Adversarial Mixture of Training Signal Generators. ICLR.
- Meng, Y., Huang, J., Zhang, Y., & Han, J. (2022) “Generating Training Data with Language Models: Towards Zero-Shot Language Understanding”, NeurIPS’22
- **Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, Jianfeng Gao, "Large Language Models: A Survey", https://arxiv.org/abs/2402.06196**
- Nickel, M., & Kiela, D. (2017). Poincaré Embeddings for Learning Hierarchical Representations. NIPS
- Nickel, M., & Kiela, D. (2018). Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. ICML.
- OpenAI (2023). GPT-4 Technical Report.

References (III)

- Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.
- Peters, M.E., Neumann, M., Iyyer, M., Gardner, M.P., Clark, C., Lee, K., & Zettlemoyer, L.S. (2018). Deep contextualized word representations. NAACL.
- Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., & Riedel, S. (2019). Language models as knowledge bases? EMNLP.
- Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. OpenAI blog.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.
- Schick, T., & Schütze, H. (2021). Exploiting cloze questions for few shot text classification and natural language inference. EACL.
- Touvron, H et al. LLaMA: Open and Efficient Foundation Language Models/LLaMA 2: Open Foundation and Fine-Tuned Chat Models
- Tifrea, A., Bécigneul, G., & Ganea, O. (2019). Poincare Glove: Hyperbolic Word Embeddings. ICLR.
- Turian, J.P., Ratinov, L., & Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-Supervised Learning. ACL.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "***Attention is All You Need***", NIPS (2017)
- Wei, J., et al. (2022). Emergent Abilities of Large Language Models. TMLR.
- Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., Ma, X., Efrat, A., Yu, P., Yu, L., Zhang, S., Ghosh, G., Lewis, M., Zettlemoyer, L., & Levy, O. (2023). LIMA: Less Is More for Alignment.