Escuela AI HUB de verano
Sobre IA y Educación

INTRODUCTION TO TRANSFORMERS

Jesús Cerquides (IIIA-CSIC)
(2/07/2024)

CSIC Consejo Superior de Investigaciones Científicas | CSIC HUB Inteligencia Artificial | Fundación "la Caixa" | IFIC Institut de Física Corpuscular | Universitat de València

Who am I?

http://www.iiia.csic.es/~cerquide

Jesus Cerquides

Scientific Researcher, IIIA/CSIC

cerquide@iiia.csic.es

+34 934840520

Interests

- AI for Participatory Sciences & Democracy
- Probabilistic Graphical Models
- Causal Inference
- Machine Learning in Health
- Quantum Machine Learning

Academia

BarcelonaTech 1996 - 2003
Ph.D. Artificial Intelligence

UAB 2008 - 2011
Degree Mathematics
UAB prize to the best student in that promotion

BarcelonaTech 1990 - 1995
M. Sc. Computer Science
Spanish prize to the best student in CS in the country that year

News

Latest things I have been doing

What I hope you will learn

- Transformers and composites
- The main tasks you can solve with transformers
- How these tasks can be solved and main transformer constructs and layers.
- Opening transformer layers. How do transformers work internally.

Attention Timeline

1990s — **Prehistoric Era**

Rule-based methods, parsing, RNNs, LSTMs

2014 — **Simple attention mechanisms**

2017 — **Beginning of transformers**

Attention is all you need

2018 — **Explosion of transformers in NLP**

BERT, GPT-3

2018-2020 — **Explosion into other fields**

Explosion into other fields: ViTs, Alphafold-2.

2021-2022 — **Start of Generative Era**

Codex, Decision Transformers, GPT-X, DALL·E

2024 — **Present Day**

Huge models, more applications: Chat-GPT, GPT-4, Gemini, Llama and open-source LLMs, Whisper, Robotics Transformer, Stable Diffusion, Sora, and so much more…!

Future (?!)

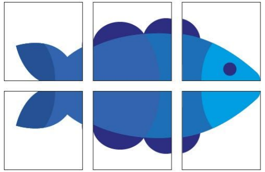
What are transformers?

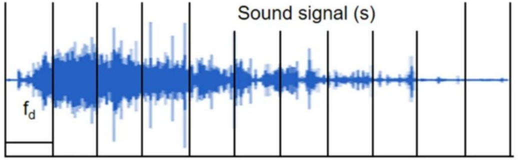
Transformers are a neural network architecture for working with **composites:**

Composite: Something that is made up of distinct parts or elements.

Examples of composites:

Text: The elements of a sentence are words

Images: 
Patches

Sound: 

Frames

Each of these elements are referred to generically as **tokens.**

The process of breaking a composite into tokens is known as **tokenization.**

What can we do with transformers?

📝 **Natural Language Processing**: text classification, named entity recognition, question answering, language modeling, summarization, translation, multiple choice, and text generation.

🖼️ **Computer Vision**: image classification, object detection, and segmentation.

🗣️ **Audio**: automatic speech recognition and audio classification.

🐙 **Multimodal**: table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.

What can we do with transformers?

Classify composites

Complete composites

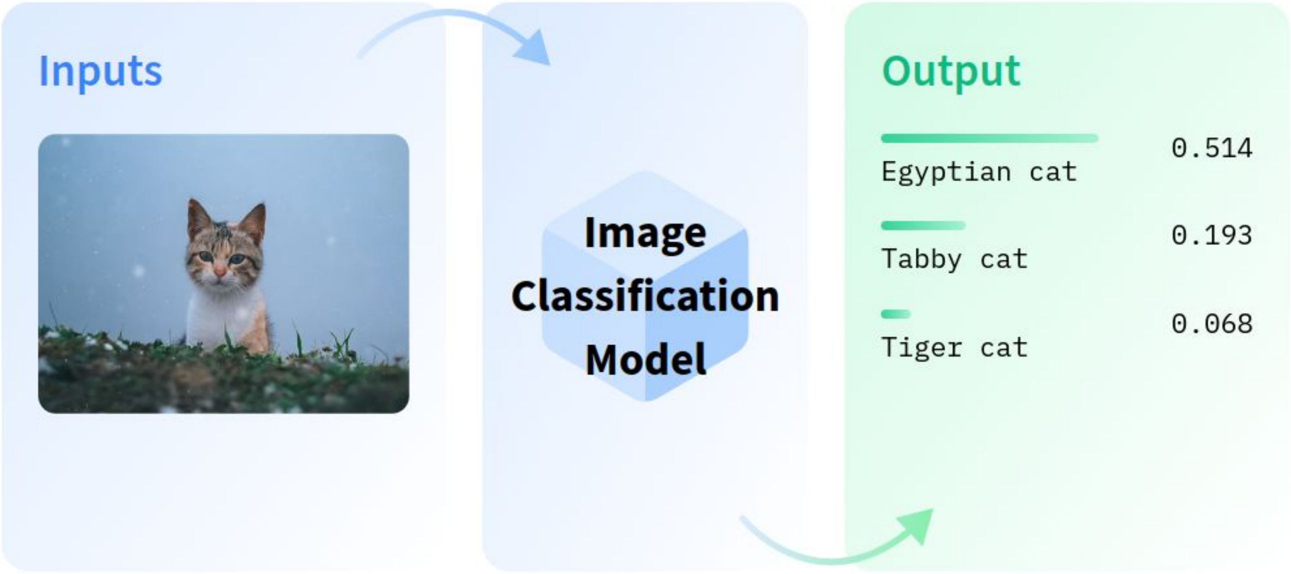
Transform composites

What can we do with transformers?
Classify composites

Task: Given a composite classify it as belonging to one specific class

Examples:

- **Image classification**

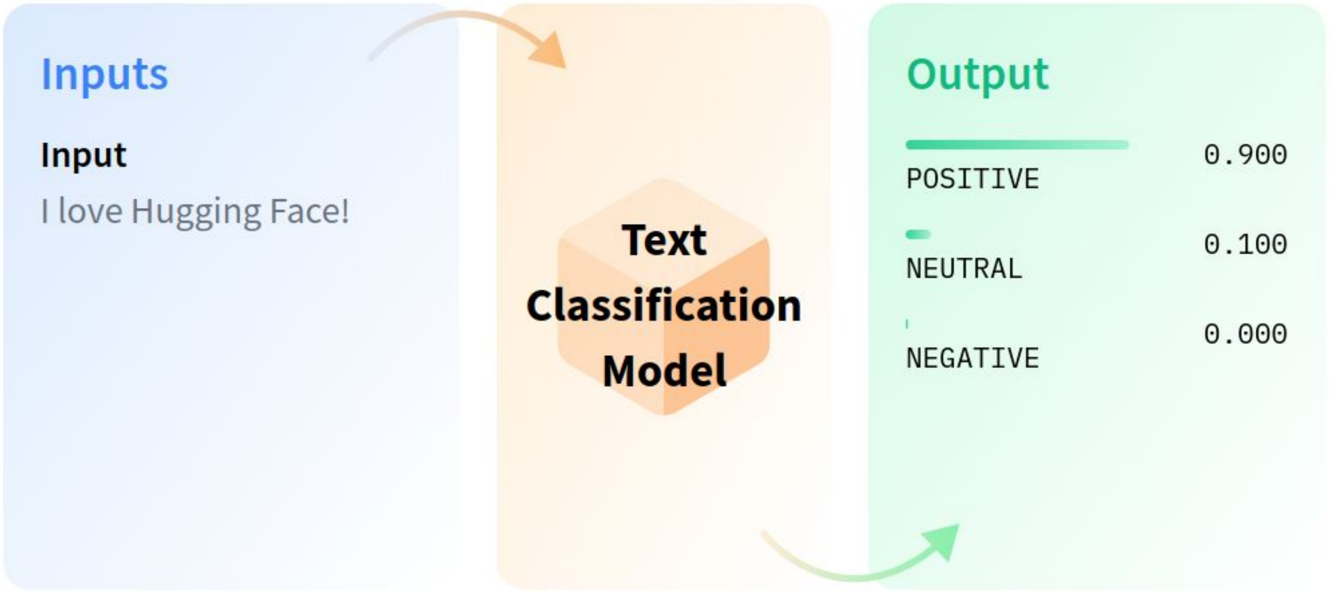


What can we do with transformers? Classify composites

Task: Given a composite classify it as belonging to one specific class

Examples:

- **Text classification**

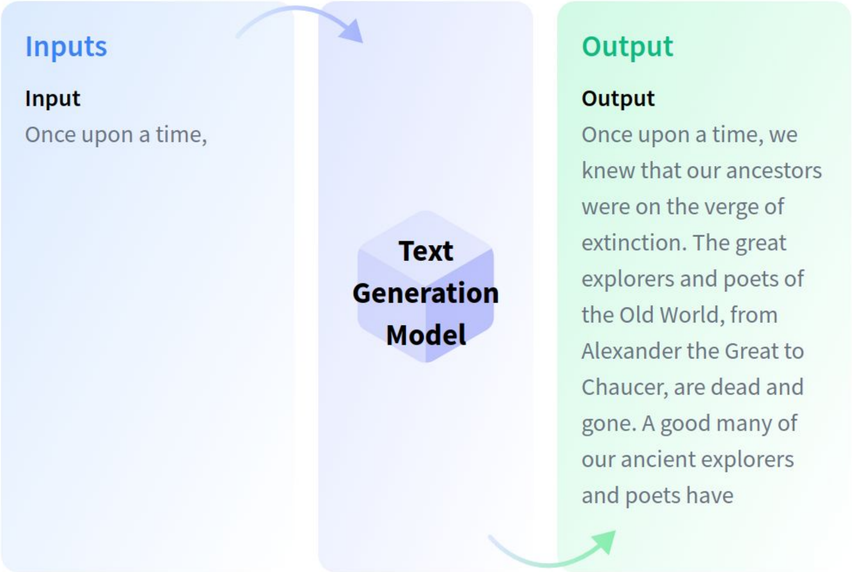


What can we do with transformers? Complete composites

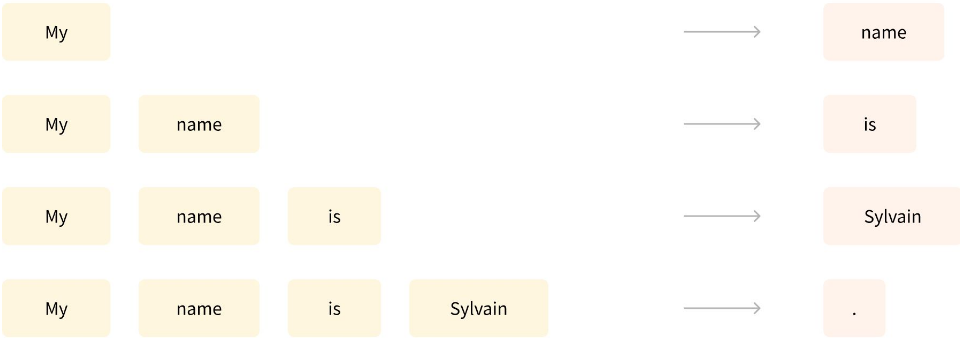
Task: Given a composite with one (or several) missing token/s, complete it in a reasonable way.

Examples:

- **Text generation (Causal Language Modeling).**



Autoregressive

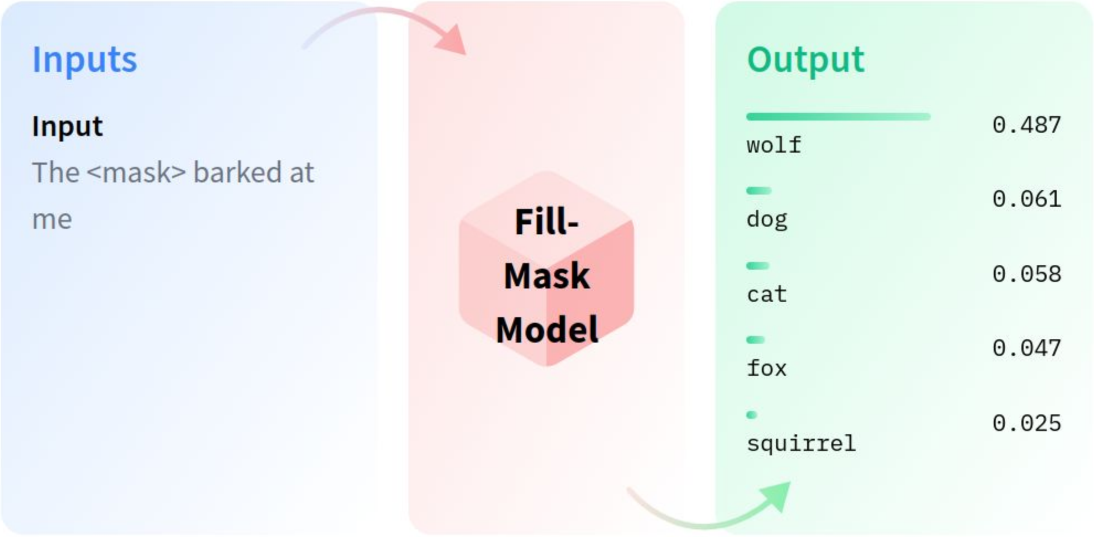


What can we do with transformers? Complete composites

Task: Given a composite with one (or several) missing token/s, complete it in a reasonable way.

Examples:

- **Masked language modeling (Fill-Mask):**

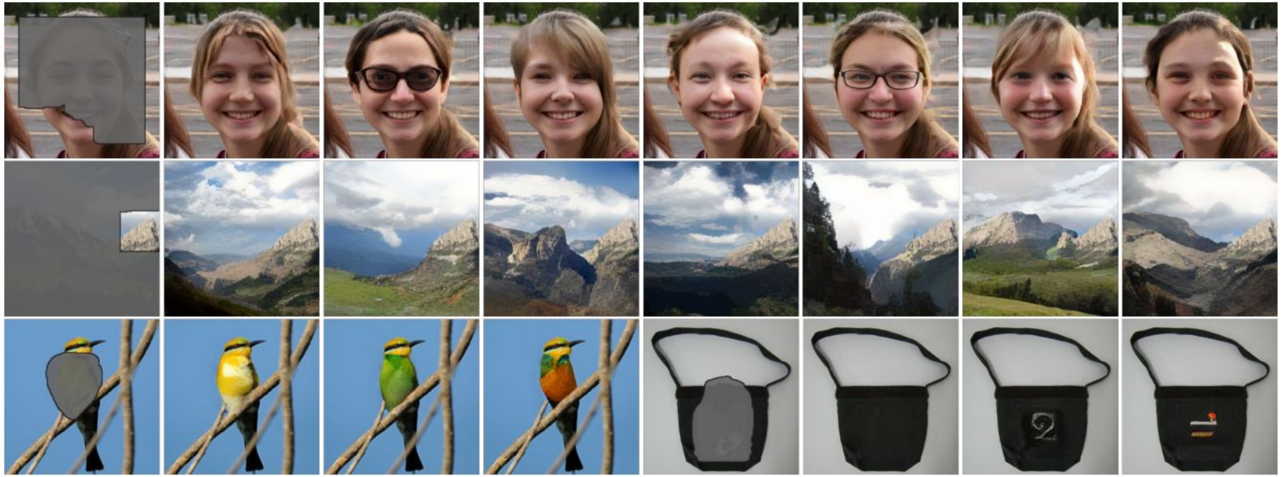


What can we do with transformers?
Complete composites

Task: Given a composite with one (or several) missing token/s, complete it in a reasonable way.

Examples:

- Image completion (Image inpainting):

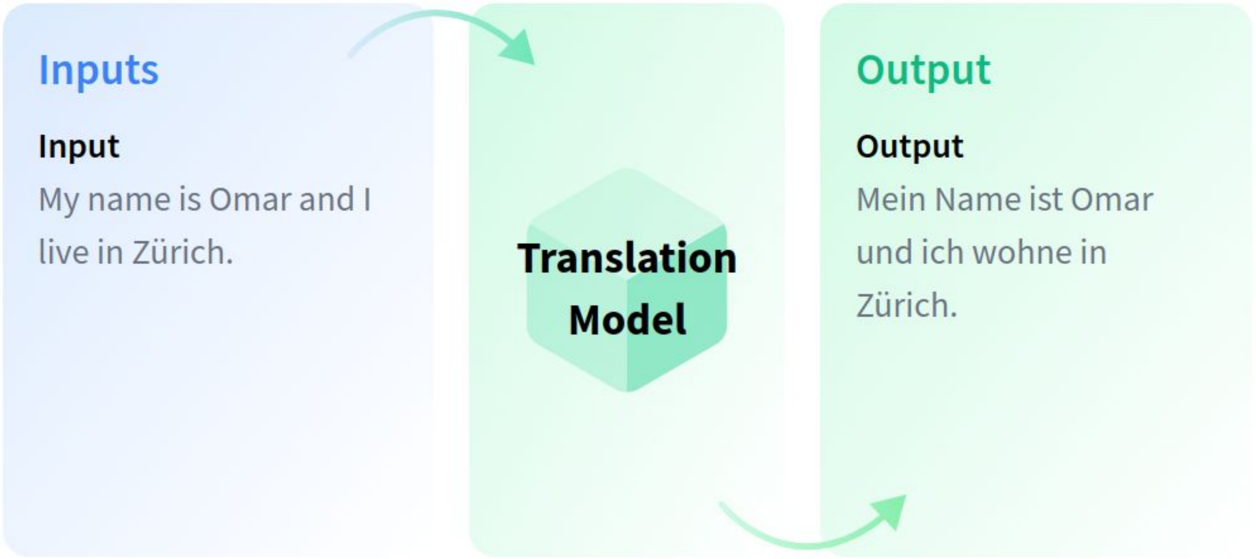


What can we do with transformers? Transform composites

Task: Given a composite transform it into a different one.

Examples:

- **Translation**



What can we do with transformers? Transform composites

Task: Given a composite transform it into a different one.

Examples:

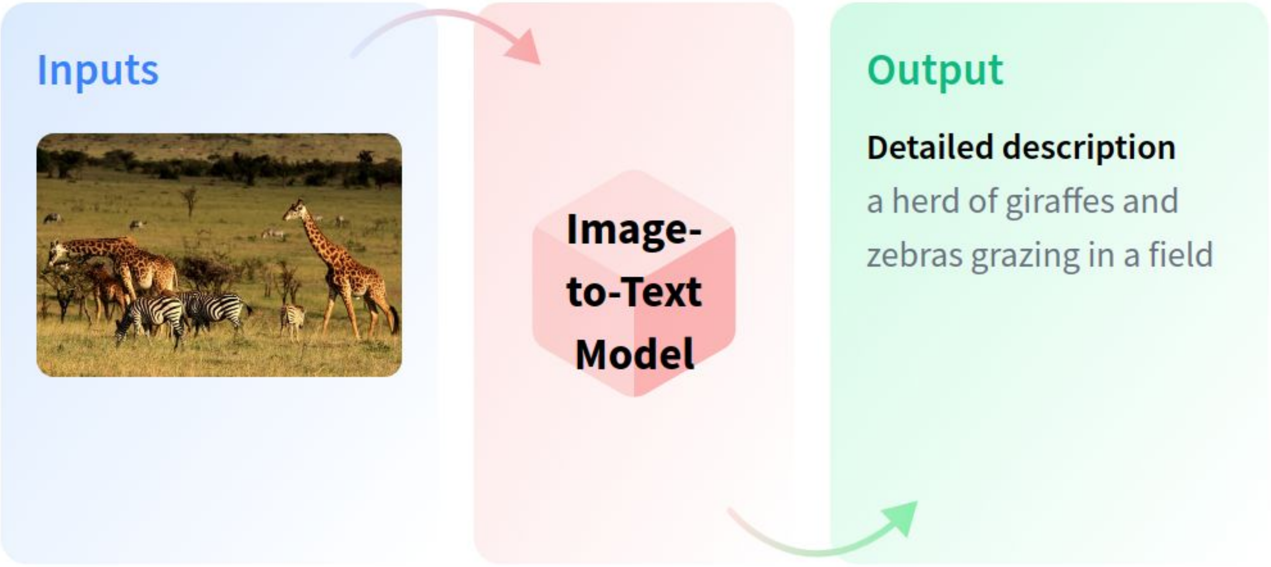
- **Summarization**

What can we do with transformers? Transform composites

Task: Given a composite transform it into a different one.

Examples:

- **Image to Text**

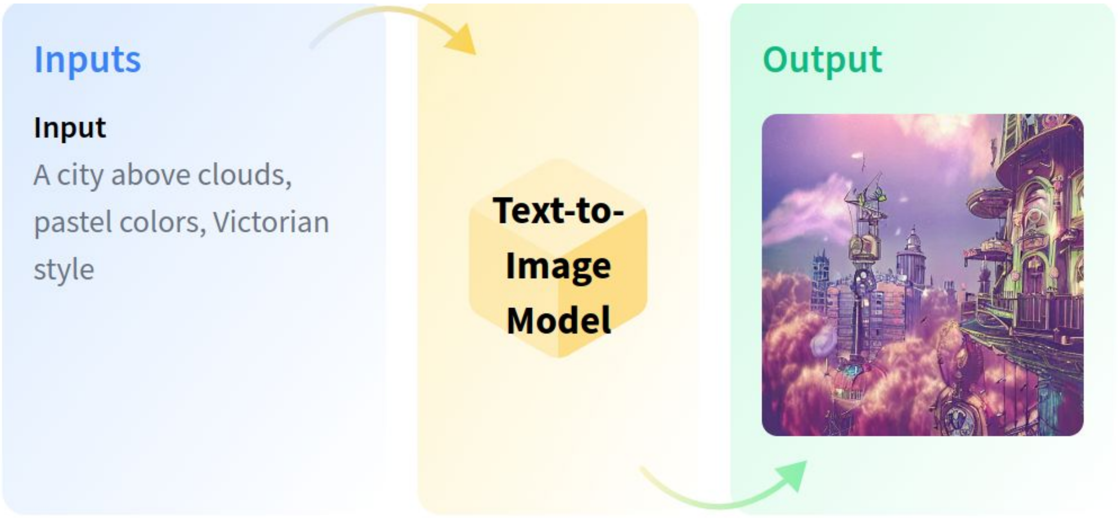


What can we do with transformers? Transform composites

Task: Given a composite transform it into a different one.

Examples:

- **Text to Image**



https://www.kdnuggets.com/2020/10/understanding-transformers-data-science-way.html

https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert/

What can we do with transformers?

Classify composites

Complete composites

Transform composites

How to classify composites with transformers?

1. Obtain a “good” semantic representation of the composite.
2. Use it to classify.

Ummmm, representations… what do we mean by that???

Representations matter

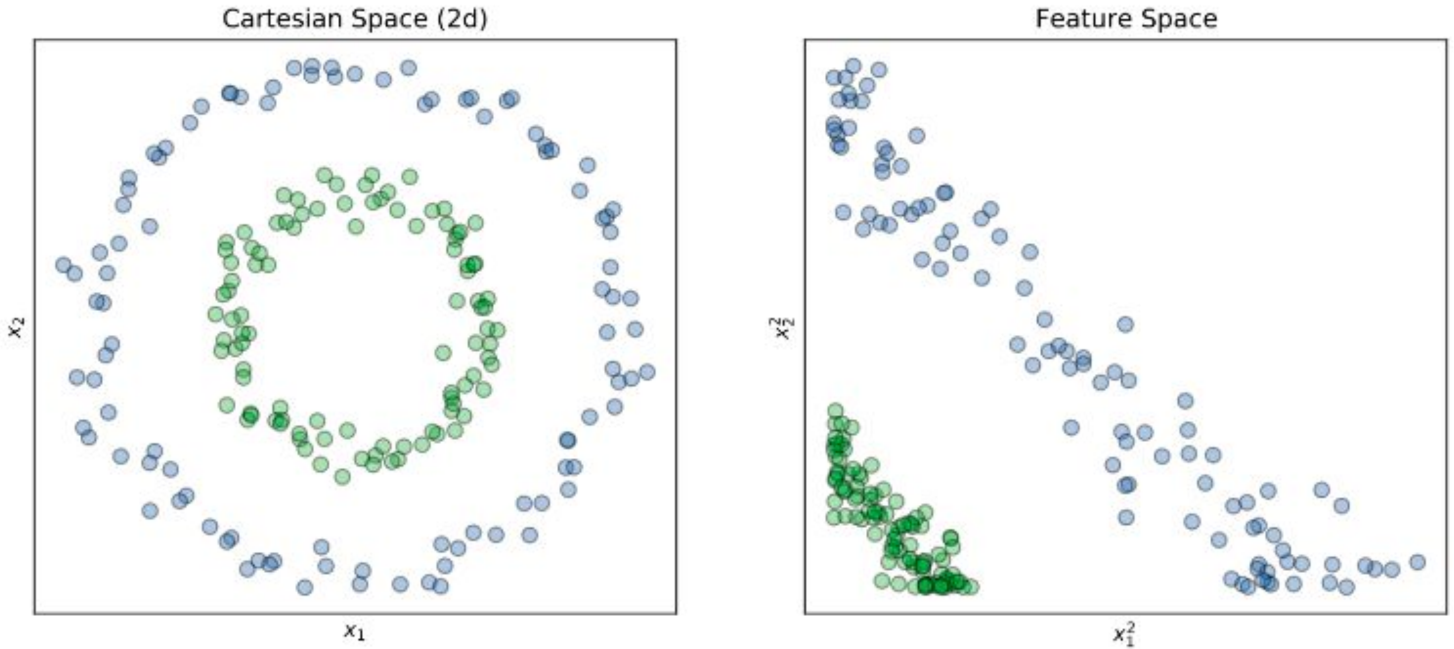
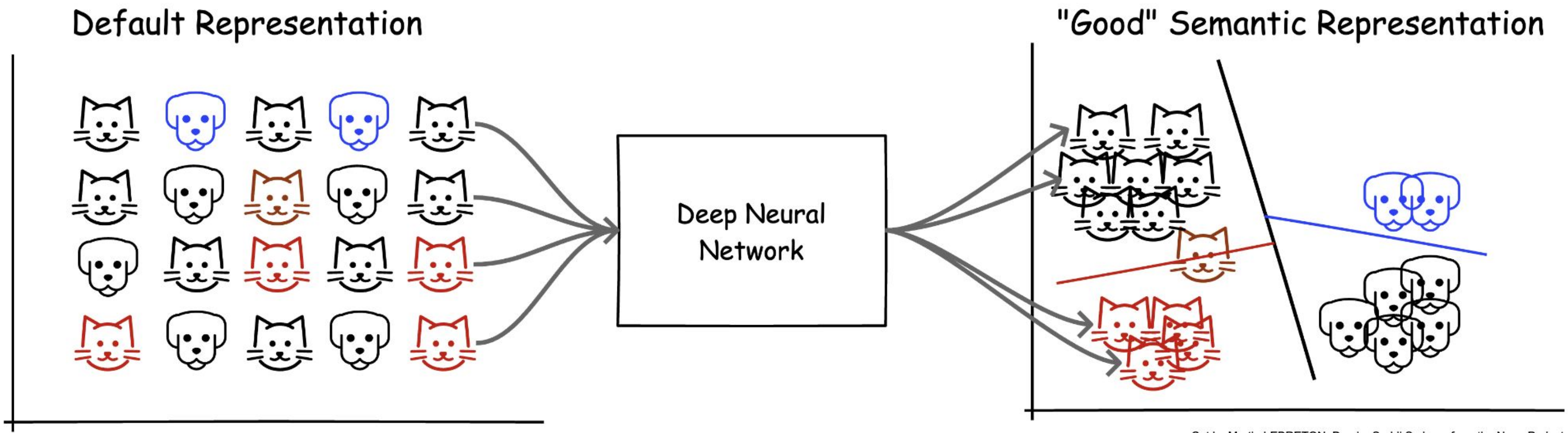


Figure 2: Representations matter for shallow machine learning models such as Logistic Regression. A simple transformation, e.g. squaring the values of the raw features, may be enough to solve the problem.

Representation learning



Cat by Martin LEBRETON, Dog by Serhii Smirnov from the Noun Project

How can we build a “good” semantic representation for composites?

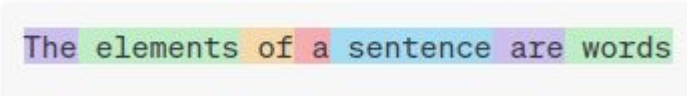
We can use an **encoder** based on transformer **layers.**

How can we build a “good” semantic representation for composites? Encoders

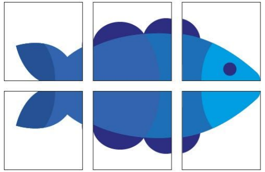
1. Obtain a representation of the composite. It does not have to be “good” or semantic. An initial representation.
2. Feed the representation through several transformer layers

How can be build a representation for composites?

Text:

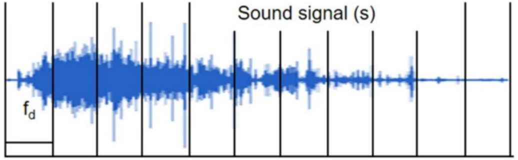


Images:



Patches

Sound:



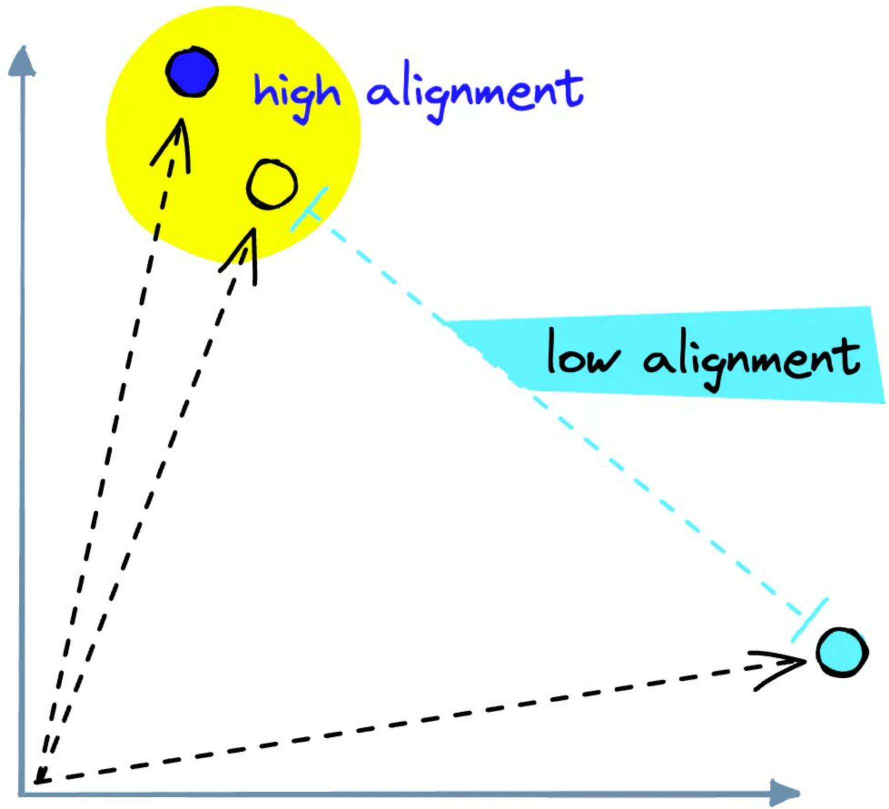
Frames

Each of these elements are referred to generically as **tokens**

We can represent each token as a vector of D real numbers.

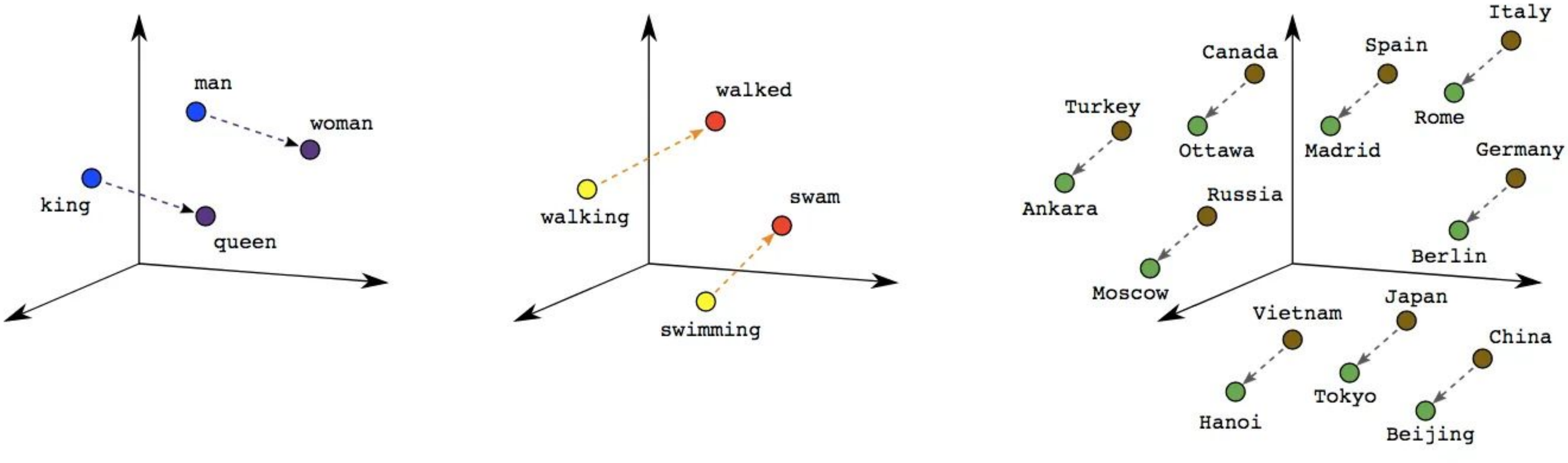
The process of going from tokens to vectors of real numbers is called **embedding**

Embeddings



Embedding is a means of representing tokens as points in a continuous vector space where the locations of those points in space are semantically meaningful to machine learning (ML) algorithms.

Word embeddings



Male-Female

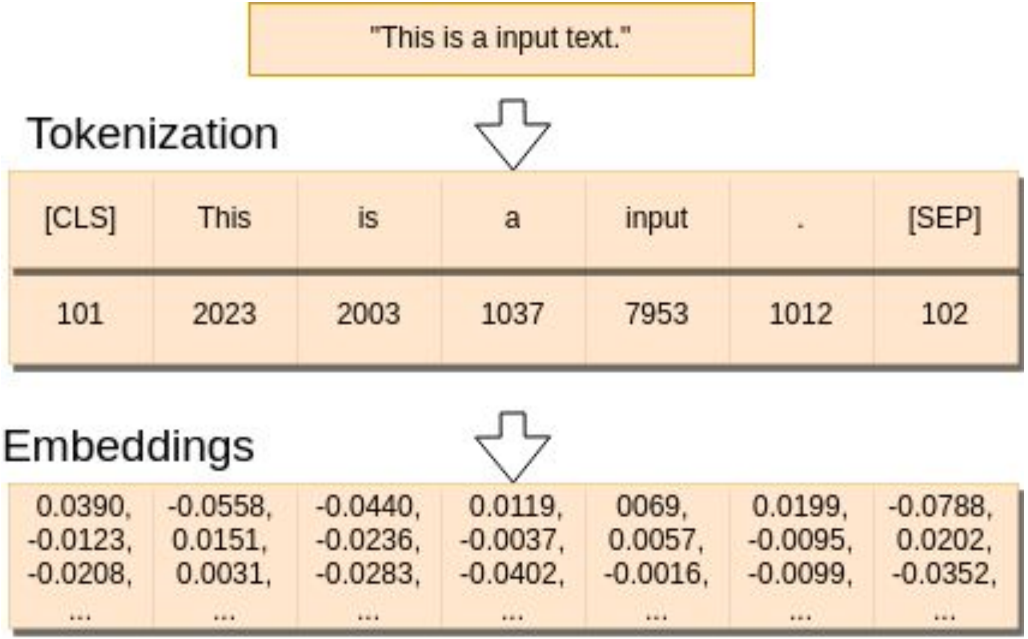
Verb Tense

Country-Capital

How can we build a representation for composites?

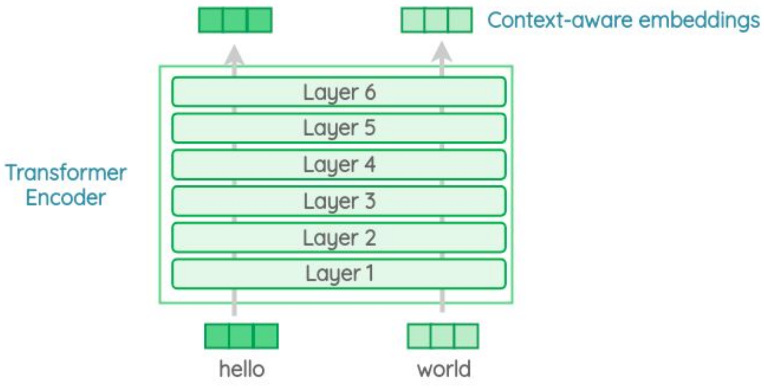
1. Tokenize the composite (divide it into tokens).

2. Embed each of the token as a vector.



How can we build a “good” semantic representation for composites? Encoders

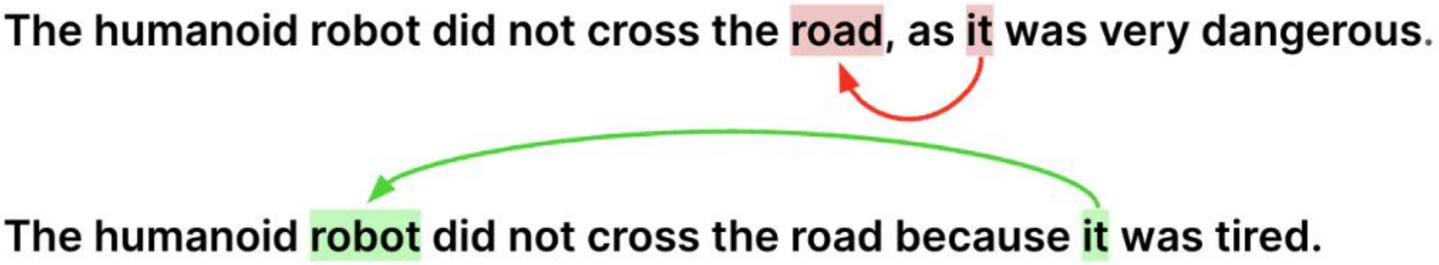
1. Obtain a representation of the composite. It does not have to be “good” or semantic. An initial representation.
2. Feed the representation through several transformer layers to obtain “context-aware” embeddings.



What are “context-aware” embeddings

The embedding of a token captures its “meaning”.

In a composite, the meaning of a token depends on other tokens

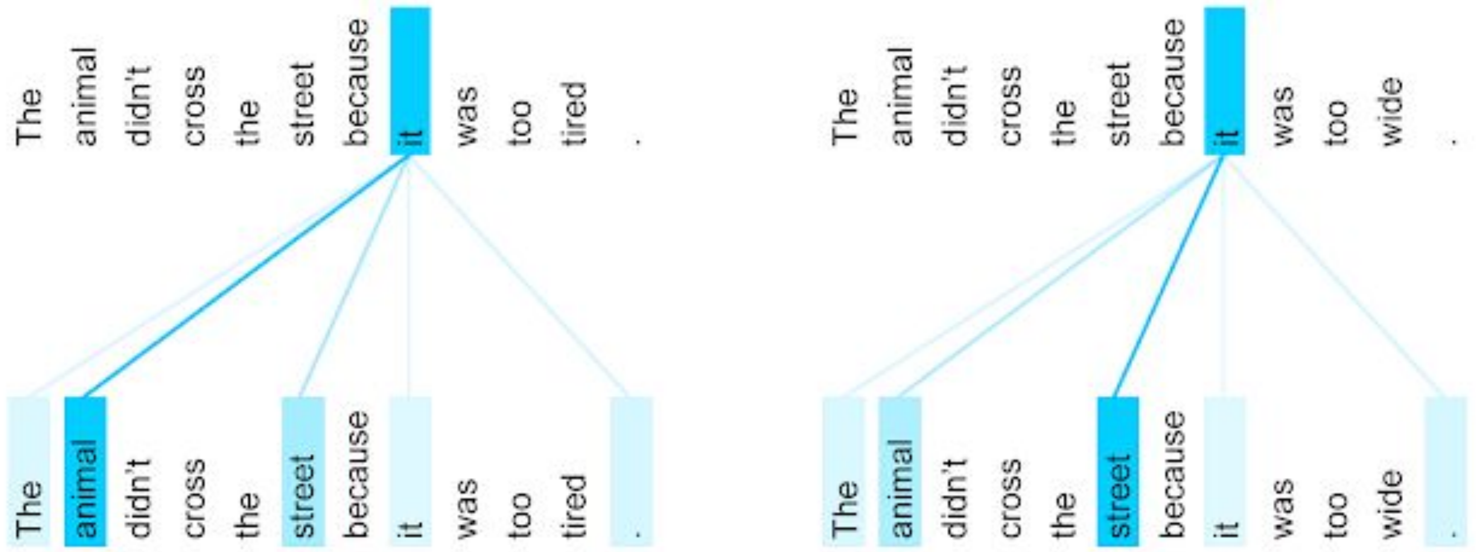


How do transformer layers deal with context?

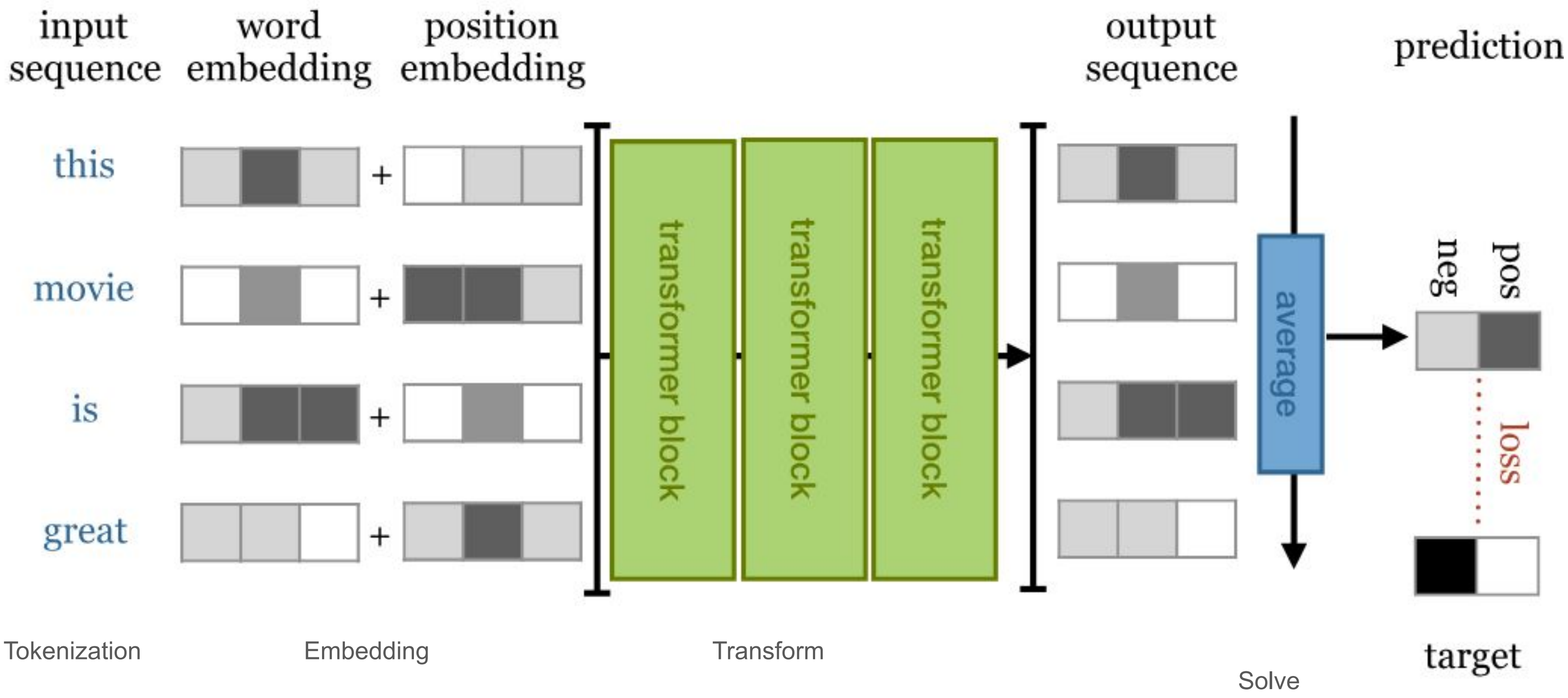
Transformer layers allows for the exchange of information between tokens.
We can understand it as:

1. Each token sends a message to every other token.
2. Each token computes how much does it want to take into account the messages of each other token. This is the idea of **attention.**
3. Each token moves away from his embedding in the direction signaled by the message of each other token weighted by the attention that he is paying to that token.

How do transformer layers deal with context? Attention

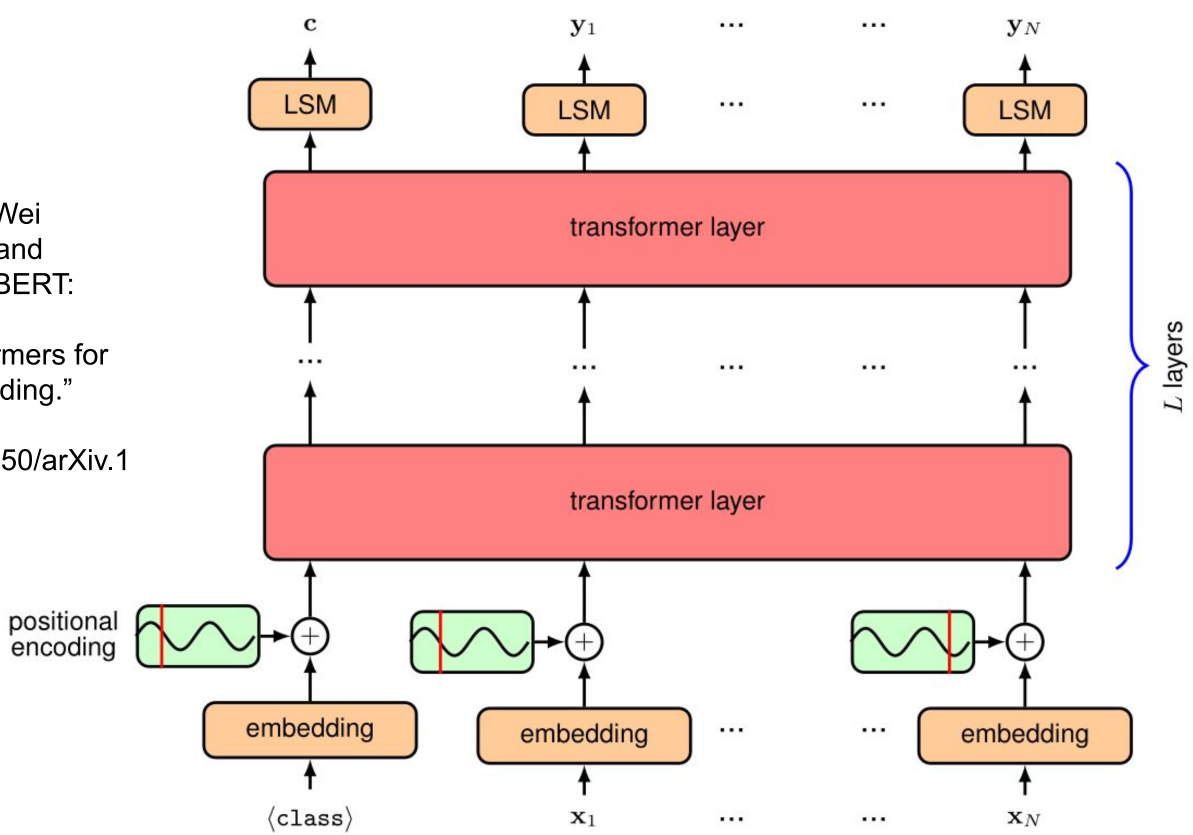


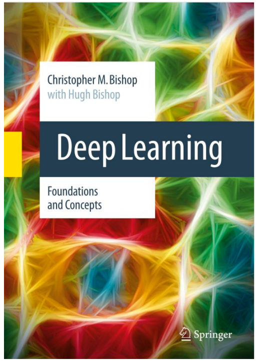
Text classification with transformers



Text classification with transformers using BERT

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding.” arXiv, May 24, 2019. https://doi.org/10.48550/arXiv.1810.04805.





Bishop, Christopher M., and Hugh Bishop. Deep Learning: Foundations and Concepts. Cham: Springer International Publishing, 2024. https://doi.org/10.1007/978-3-031-45468-4.

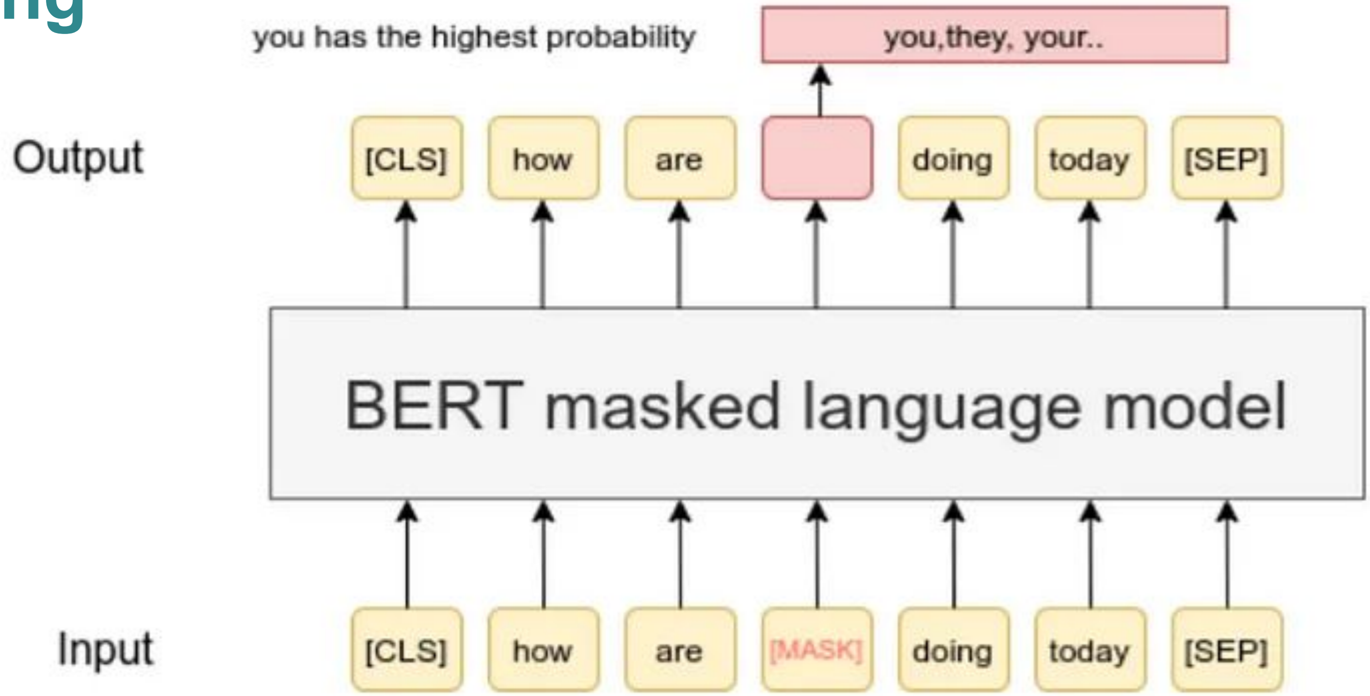
How do transformer layers deal with context?

Transformer layers allows for the exchange of information between tokens. We can understand it as:

1. Each token sends a message to every other token.
2. Each token computes how much does it want to take into account the messages of each other token. This is the idea of **attention.**
3. Each token moves away from his embedding in the direction signaled by the message of each other token weighted by the attention that he is paying to that token.

This is a general intuition. To understand better we need to understand how the encoder is trained.

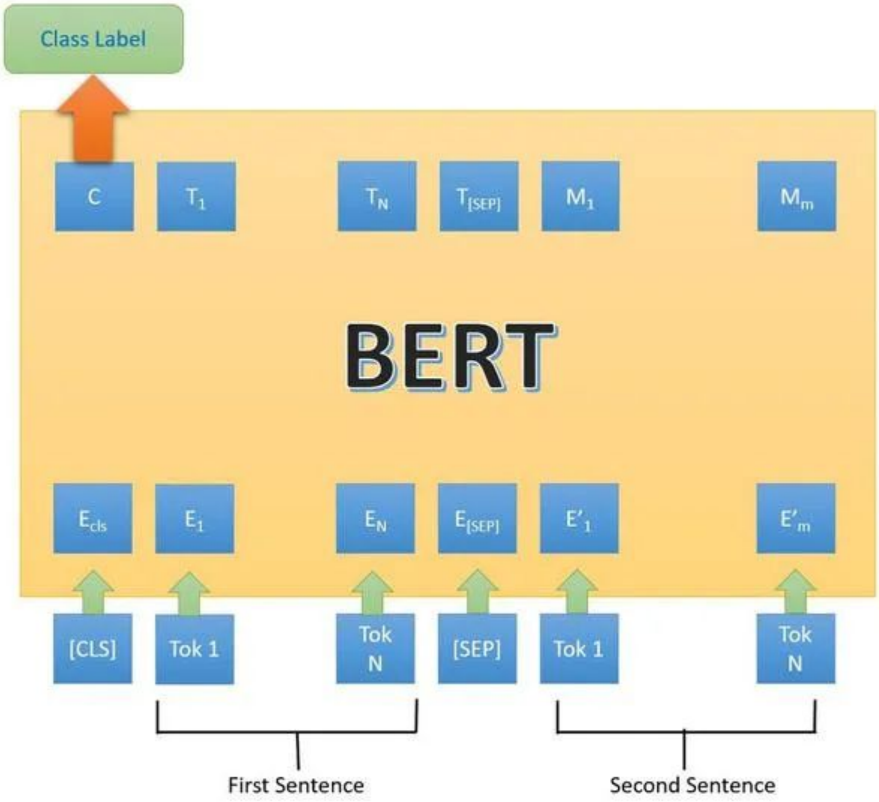
BERT training MLM



MLM teaches BERT to understand relationships between words

One of the key techniques used in pre-training BERT is the Masked Language Model (MLM). In this approach, a certain percentage of the input tokens are randomly selected (usually around 15% of the tokens) and masked (replaced with a [MASK] token). **The objective of the MLM is to predict the original masked words based on the surrounding context**. Hence, BERT is useful for completing composites (MLM).

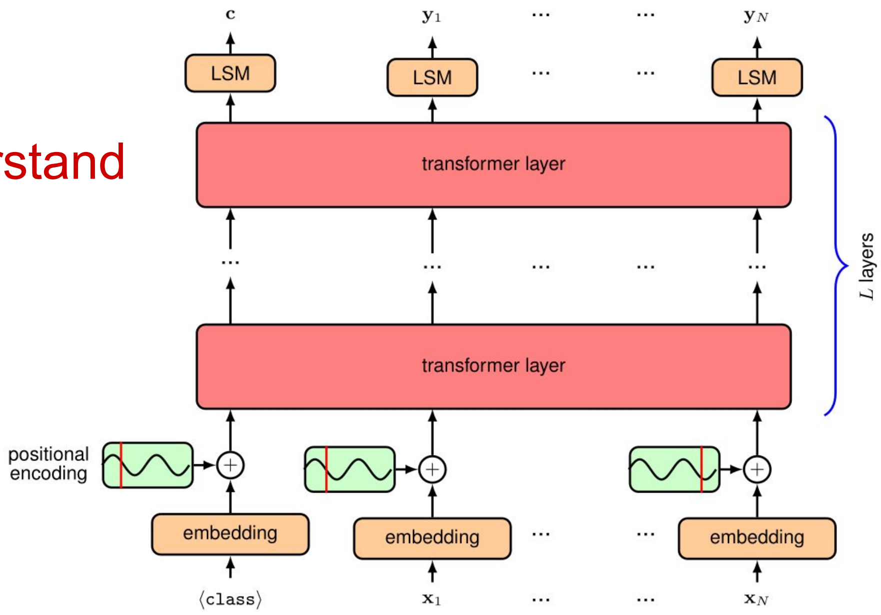
BERT training NSP

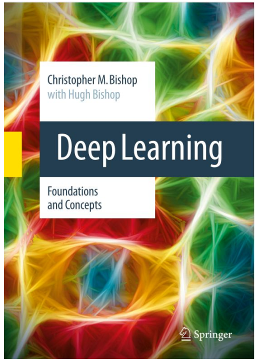


NSP teaches BERT to understand longer-term dependencies across sentences.

Text classification with transformers using BERT

Now we can understand this better!

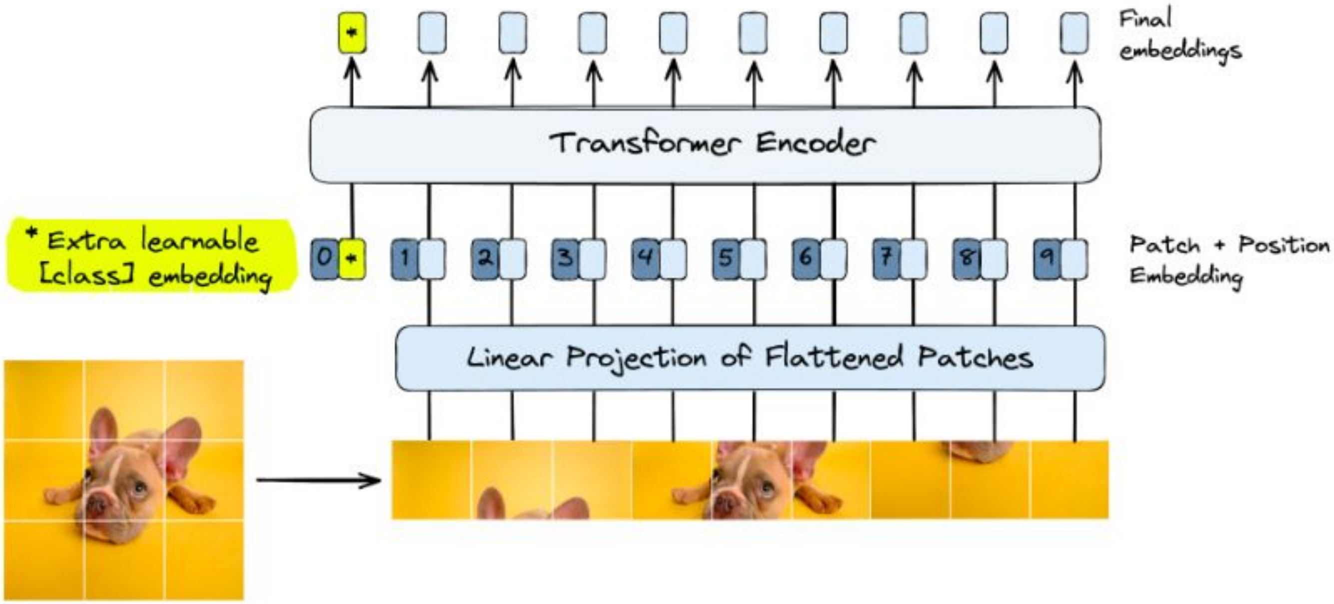




Bishop, Christopher M., and Hugh Bishop. Deep Learning: Foundations and Concepts. Cham: Springer International Publishing, 2024. https://doi.org/10.1007/978-3-031-45468-4.

Interested on fine tuning BERT for classification? Read more
Sun, Chi, Xipeng Qiu, Yige Xu, and Xuanjing Huang. "How to Fine-Tune BERT for Text Classification?" arXiv, February 5, 2020. https://doi.org/10.48550/arXiv.1905.05583.

Image classification with ViT

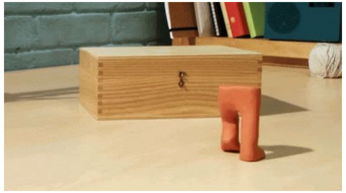


ViT process with the learnable class embedding highlight (left).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, et al. "An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale." arXiv, June 3, 2021. https://doi.org/10.48550/arXiv.2010.11929.

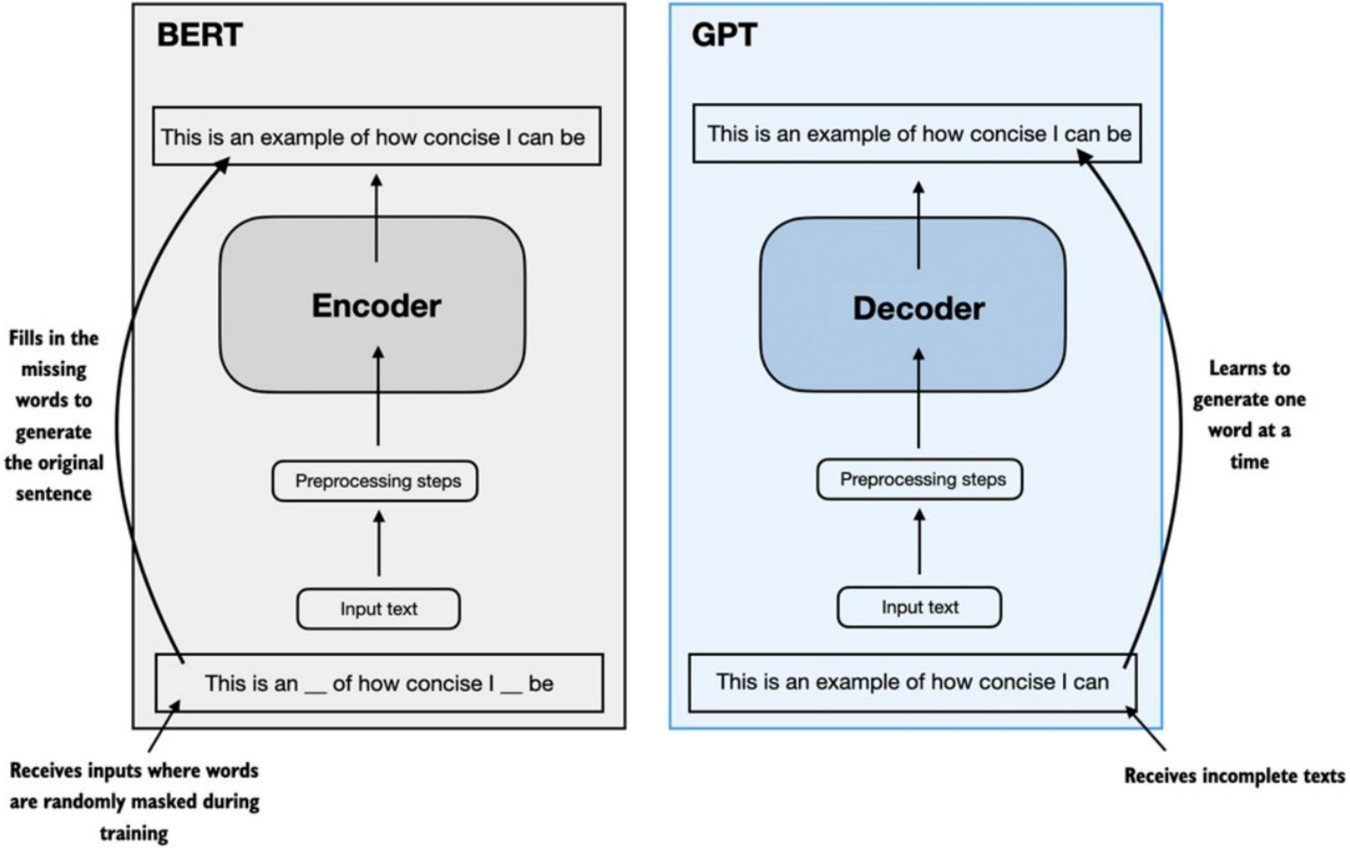
What can we do with transformers?

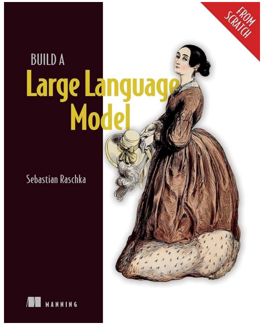
Classify composites 

Complete composites 

Transform composites

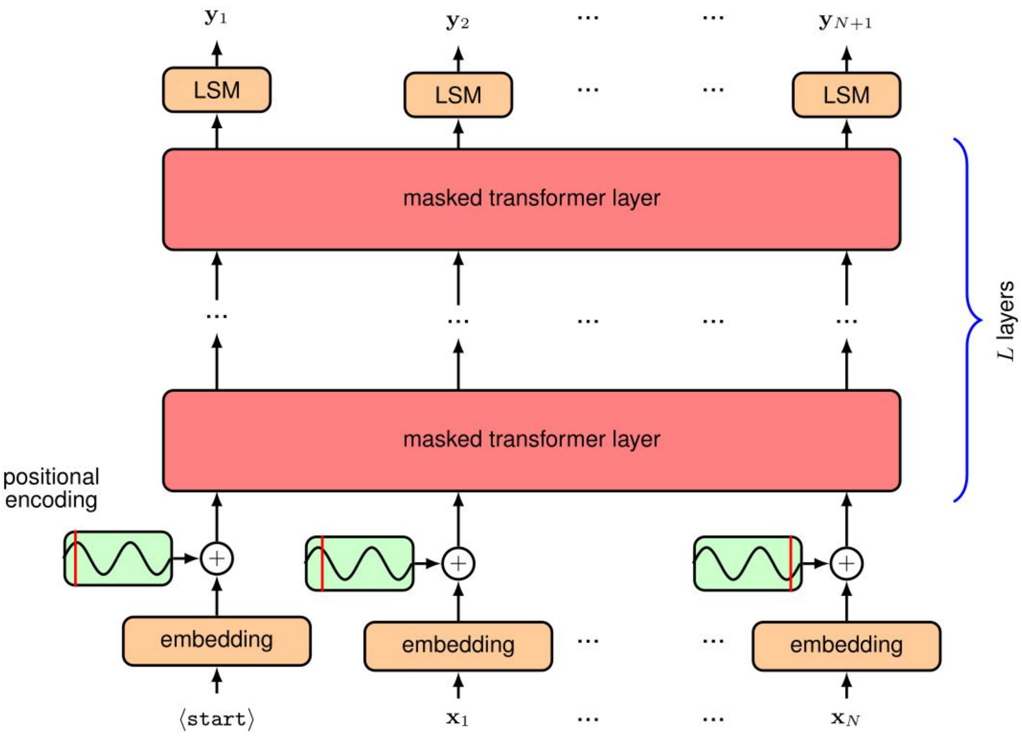
How to generate composites? Decoders

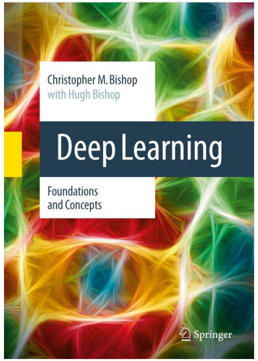




Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

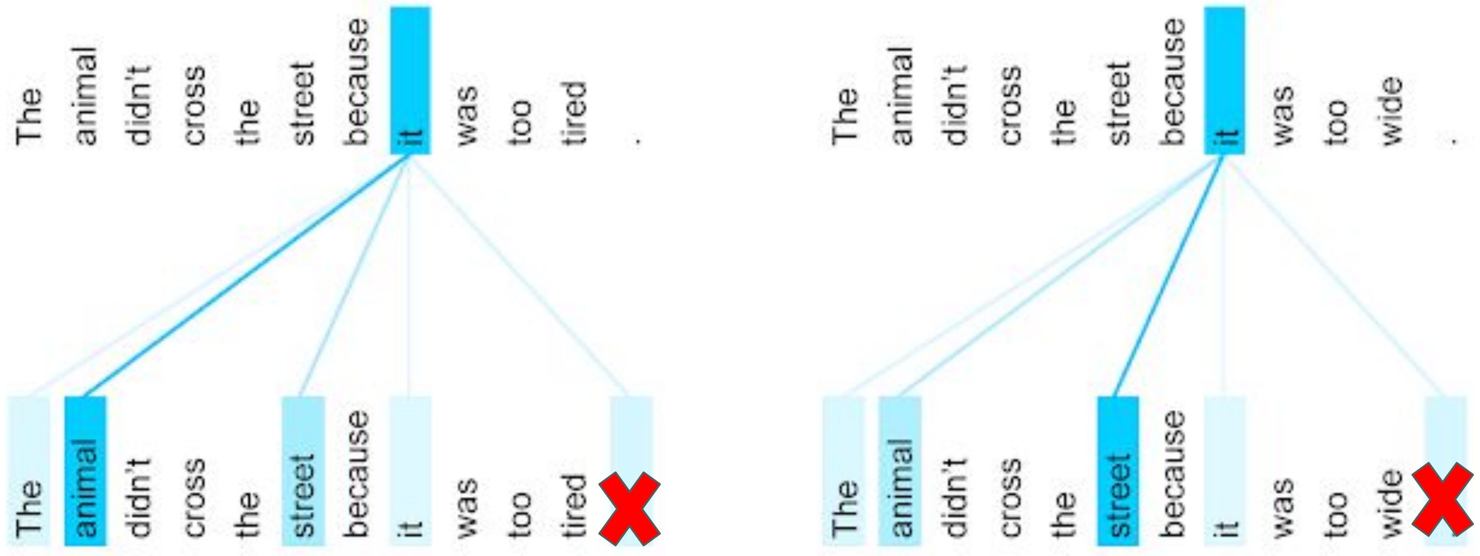
How to generate composites? Decoders



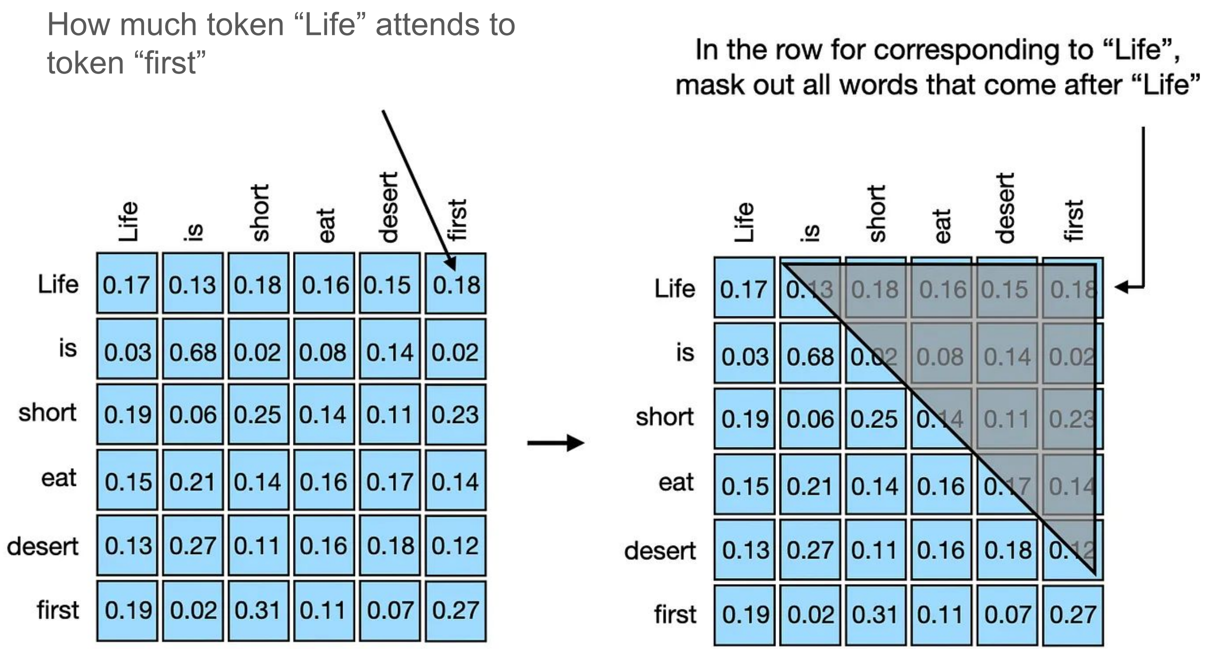


Bishop, Christopher M., and Hugh Bishop. Deep Learning: Foundations and Concepts. Cham: Springer International Publishing, 2024. https://doi.org/10.1007/978-3-031-45468-4.

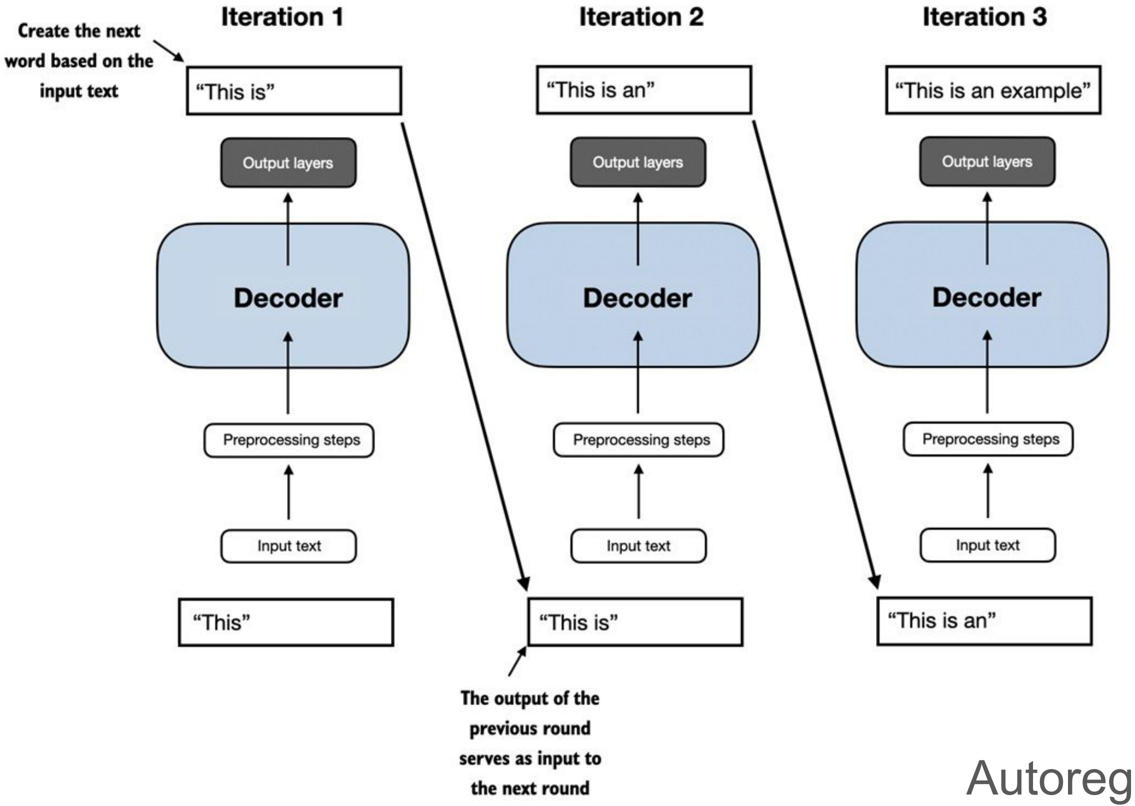
What is masked attention?

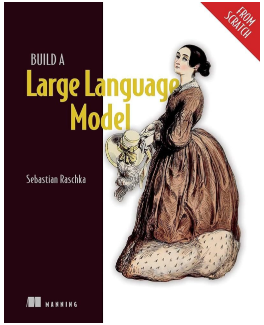


What is masked attention?



How to generate composites? Decoders

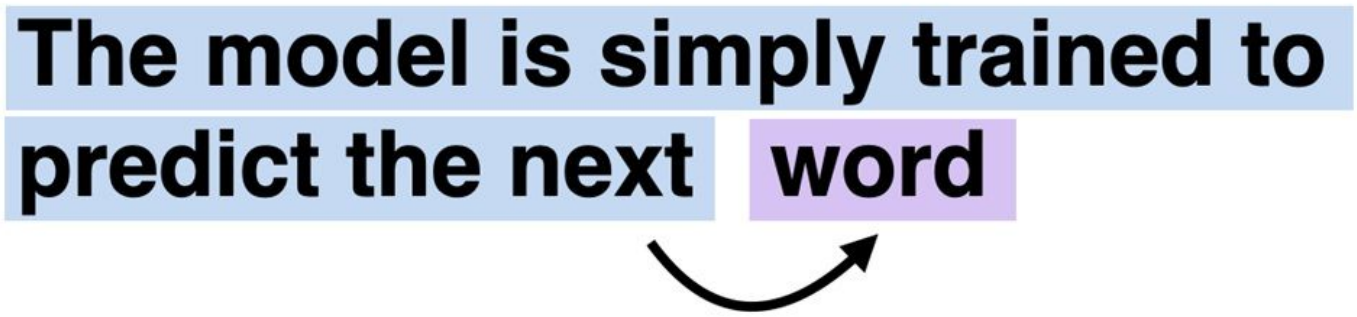


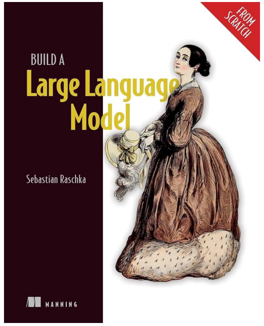


Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

Autoregresive

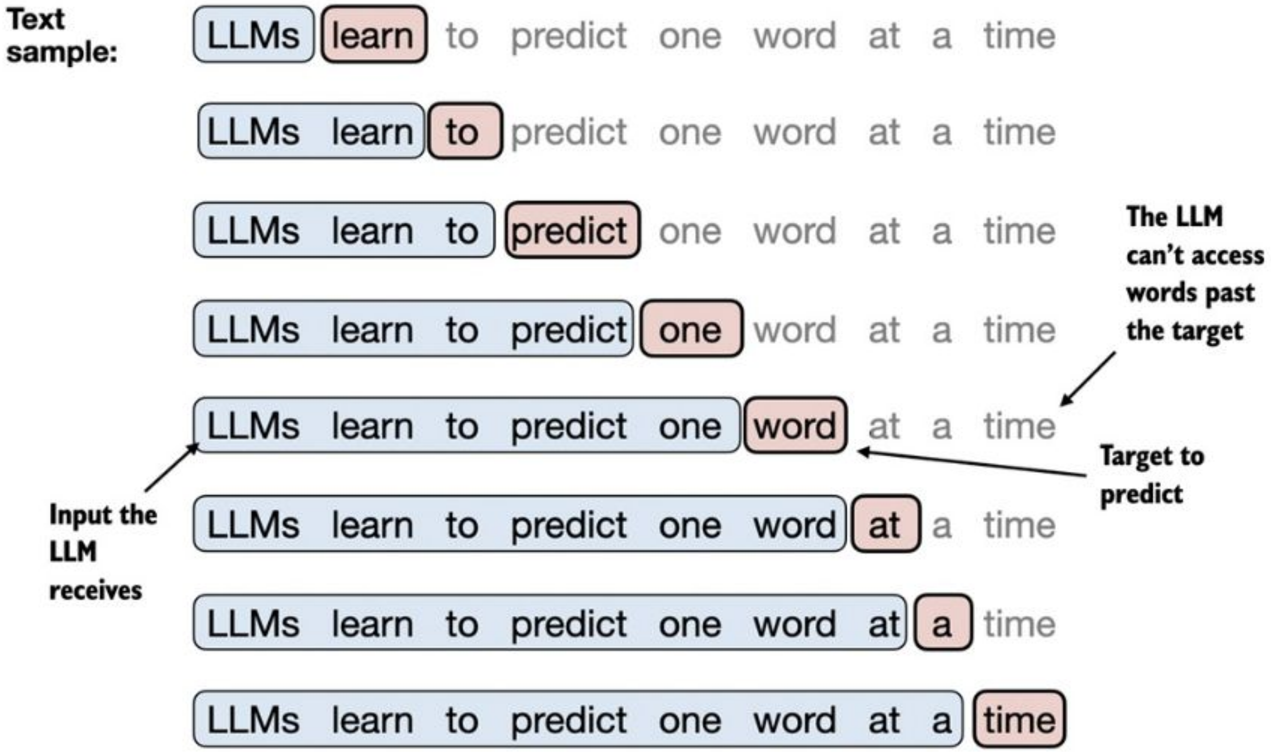
How are decoders trained?

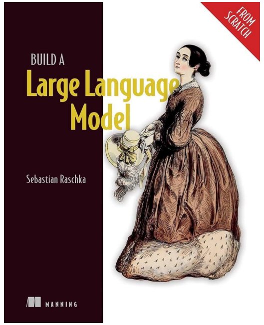




Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

How are decoders efficiently trained?





Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

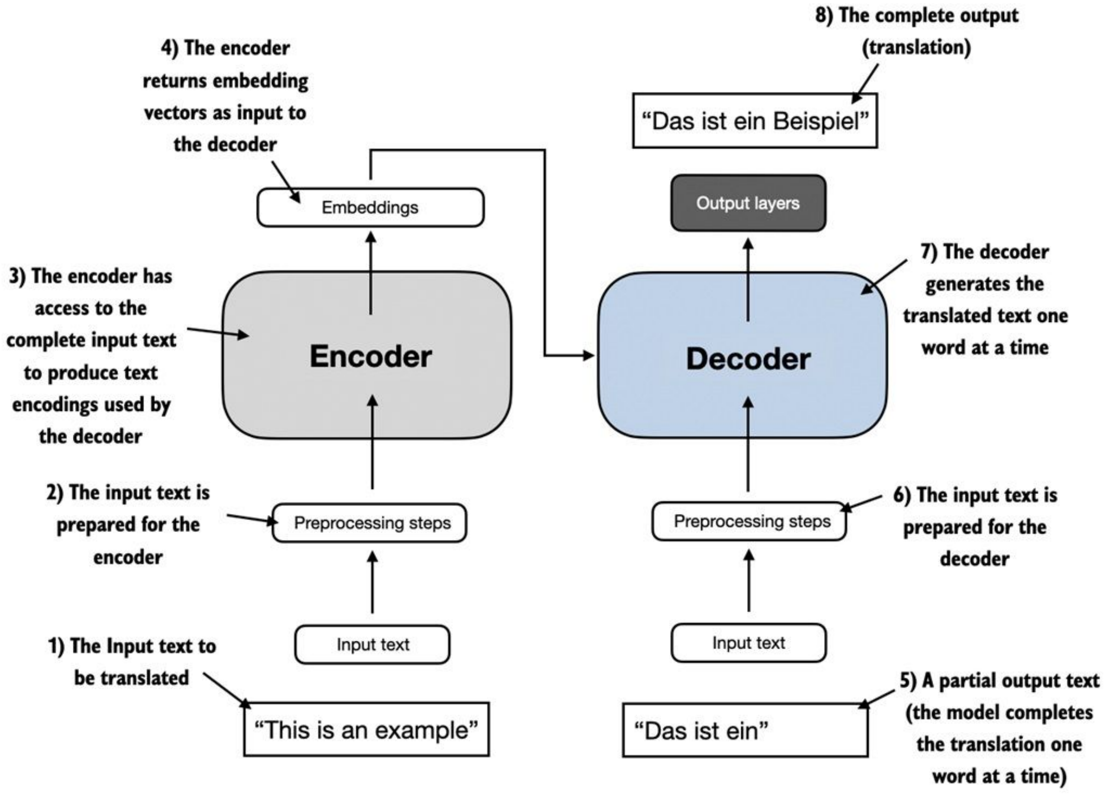
What can we do with transformers?

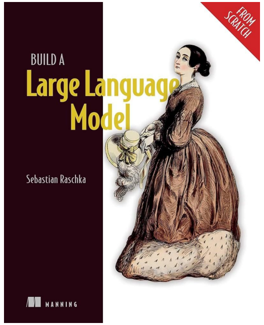
Classify composites

Complete composites

Transform composites

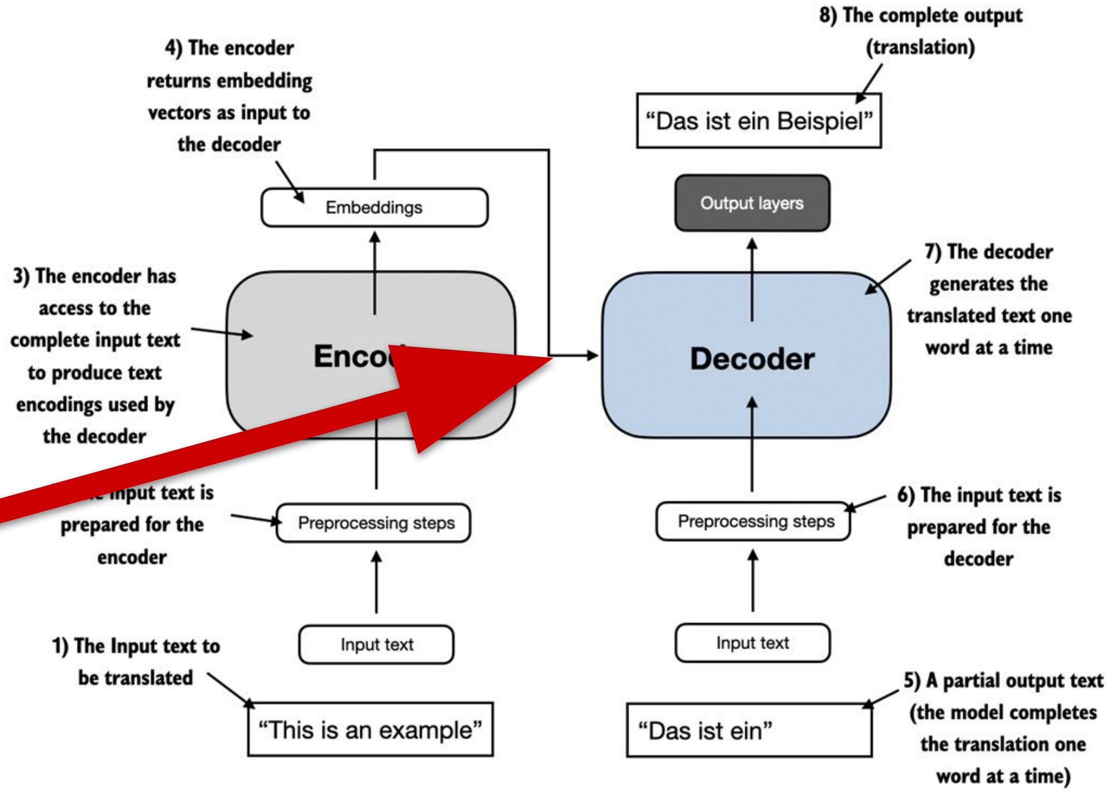
Transforming composites. Encoders and decoders together

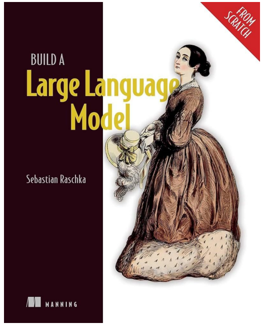




Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

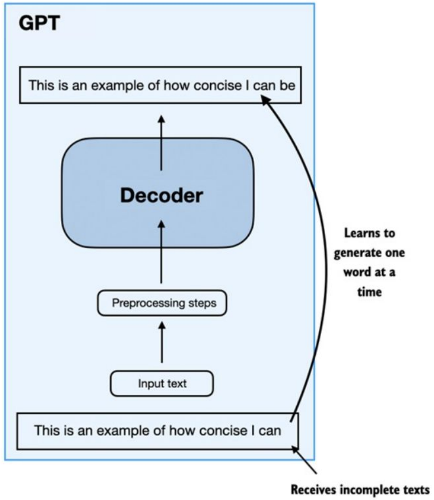
Transforming composites. Encoders and decoders together





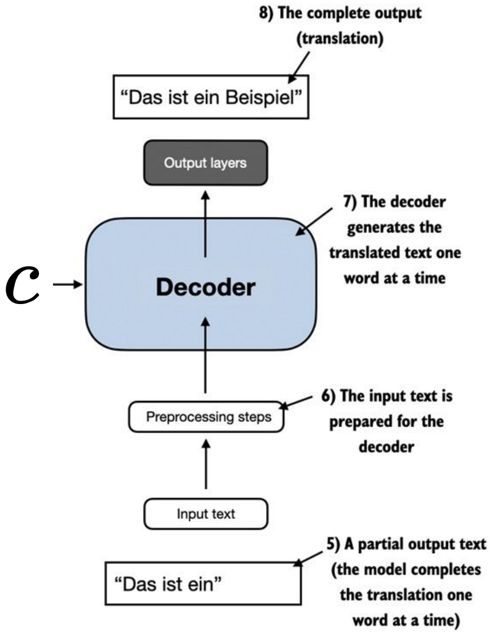
Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

Decoders vs. conditional decoders



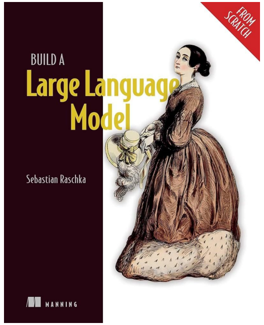
Approximate

$$p(x_n | x_1, \ldots, x_{n-1})$$



Approximate

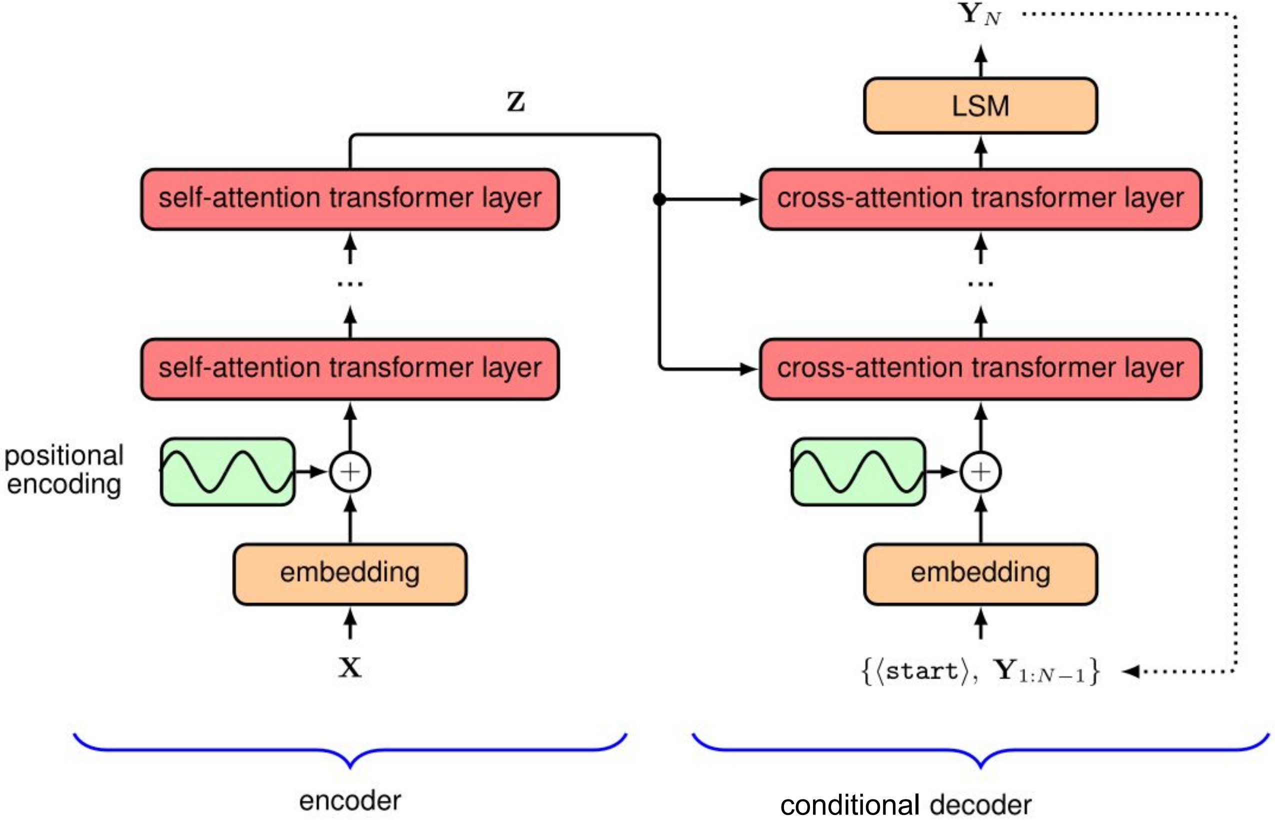
$$p(x_n | x_1, \ldots, x_{n-1}, c)$$

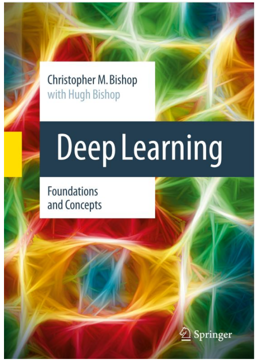


Raschka, Sebastian. Build a Large Language Model (From Scratch). Manning, 2024.

What is the most likely next word if the sentence meaning is c?

Transforming composites.





Bishop, Christopher M., and Hugh Bishop. Deep Learning: Foundations and Concepts. Cham: Springer International Publishing, 2024. https://doi.org/10.1007/978-3-031-45468-4.

What can we do with transformers?

Classify composites

Complete composites

Transform composites

Opening the box. Understanding transformer layers.

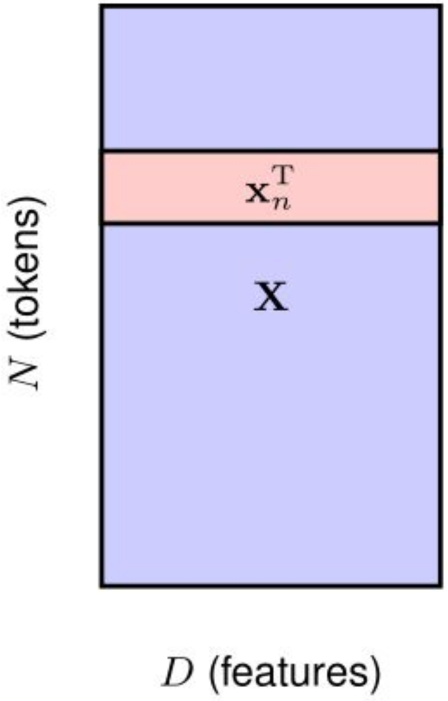
Transformer layer -> Encoders

Masked transformer layer -> Decoders

Cross-attention transformer layer -> Conditional decoders

Transformer layer

A composite as a matrix



$$\widetilde{\mathbf{X}} = \mathrm{TransformerLayer}\left[\mathbf{X}\right].$$

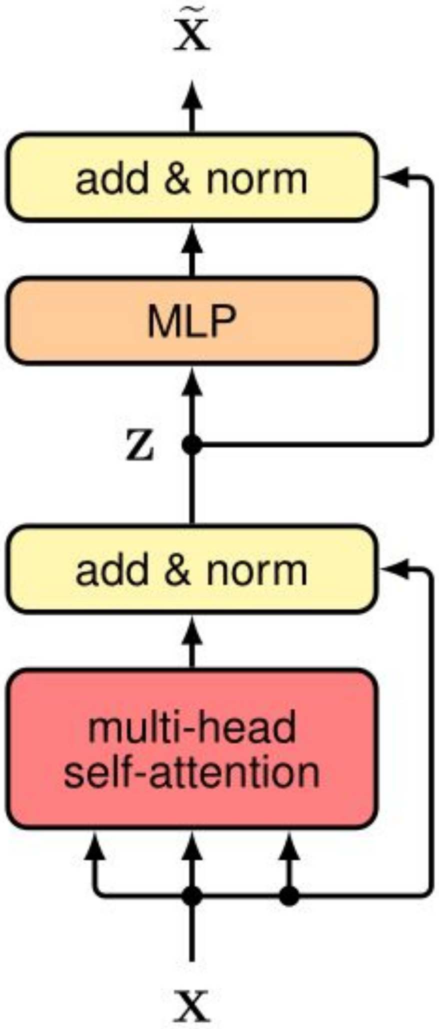
Transformer layer

Algorithm 12.3: Transformer layer

Input: Set of tokens $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
Multi-head self-attention layer parameters
Feed-forward network parameters
Output: $\widetilde{\mathbf{X}} \in \mathbb{R}^{N \times D} : \{\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_N\}$

$\mathbf{Z} = \text{LayerNorm}\left[\mathbf{Y}(\mathbf{X}) + \mathbf{X}\right]$ `// Y(X) from Algorithm 12.2`
$\widetilde{\mathbf{X}} = \text{LayerNorm}\left[\text{MLP}\left[\mathbf{Z}\right] + \mathbf{Z}\right]$ `// shared neural network`
return $\widetilde{\mathbf{X}}$

Y(X): Result of applying multi-head self-attention



Attention coefficients

$$\mathbf{y}_n = \sum_{m=1}^{N} a_{nm}\mathbf{x}_m$$

$$Y = AX$$

a_{nm}: how much token n attends to token m

$$a_{nm} \geqslant 0$$

$$\sum_{m=1}^{N} a_{nm} = 1.$$

This is a **convex combination**, play with one here:
https://www.geogebra.org/m/ekyrkytj

How to compute attention weights

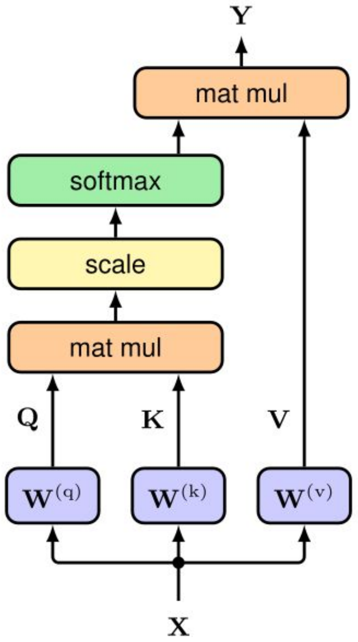
For each token x_i we compute:

- A query q_i. It encodes the kind of tokens that x_i is willing to receive messages from.
- A key k_i. It encodes the kind of tokens that x_i is willing to send messages to.
- A value v_i. It encodes the message that x_i is willing to distribute to whomever wants to pay attention to it.

Now to compute a_{nm}, the attention token n devotes to token m, we compute the dot product of $\langle q_n, k_m \rangle$. The larger the dot product, the larger the attention.

Attention via linear transformations

We can easily compute queries, keys and values using independent linear transformations of the matrix X.



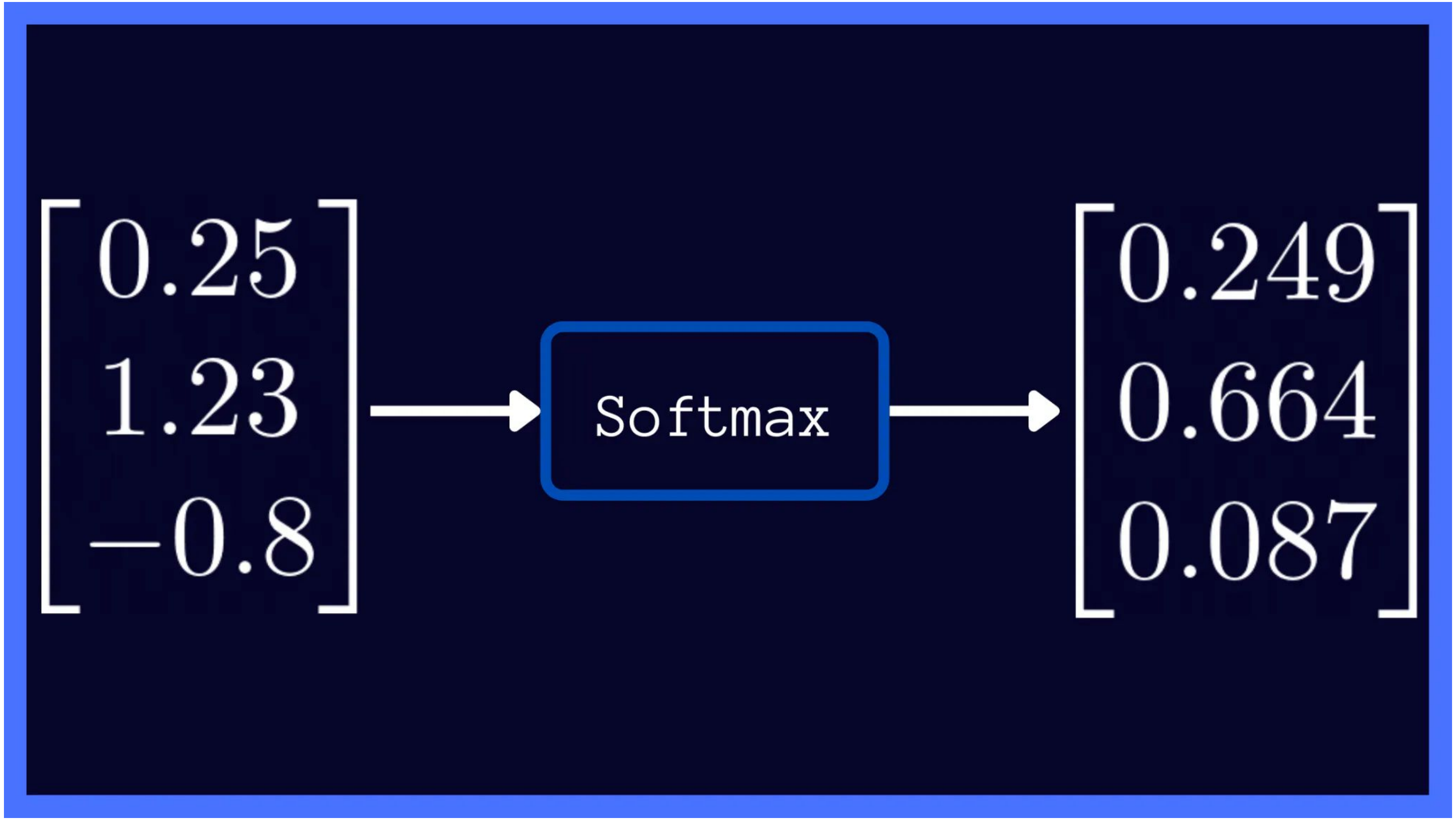
$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$$

W matrices (encoding the linear transformations) are parameters of the Transformer layer

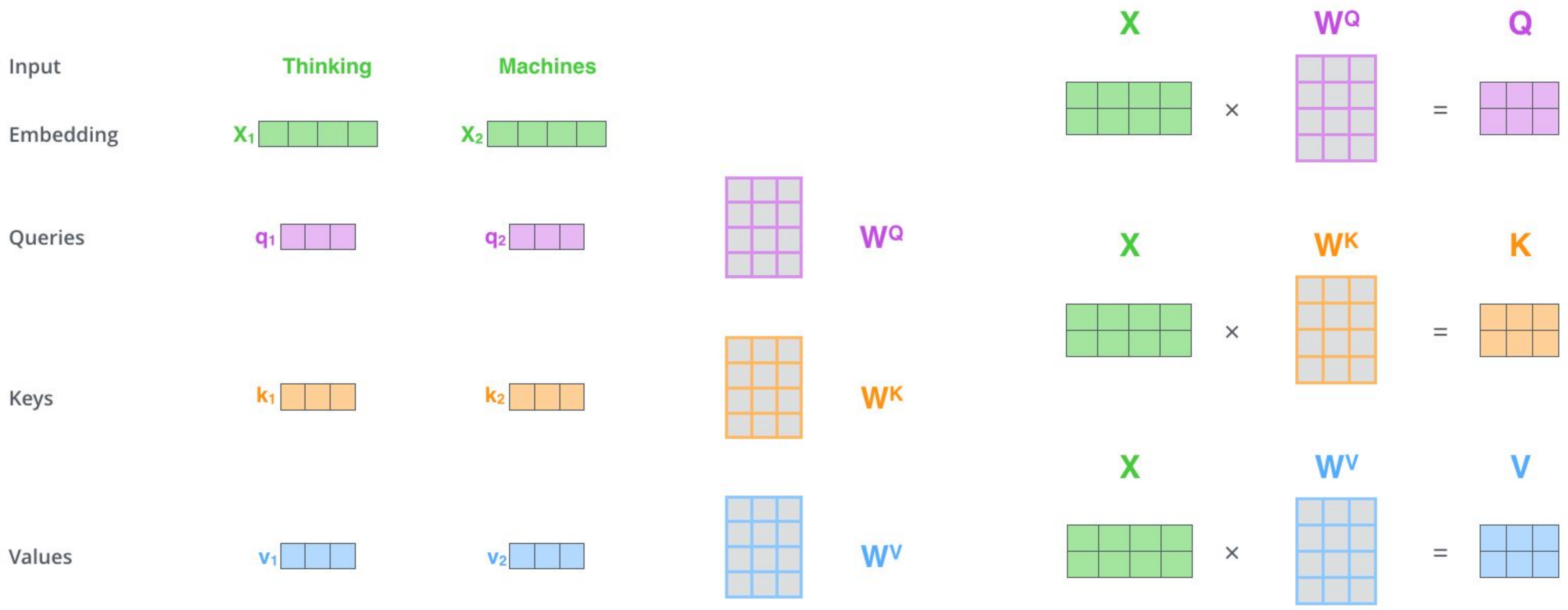
$$A = \mathrm{Softmax}\left[\frac{QK^T}{\sqrt{D_k}}\right]$$

$$Y = AV$$

Softmax

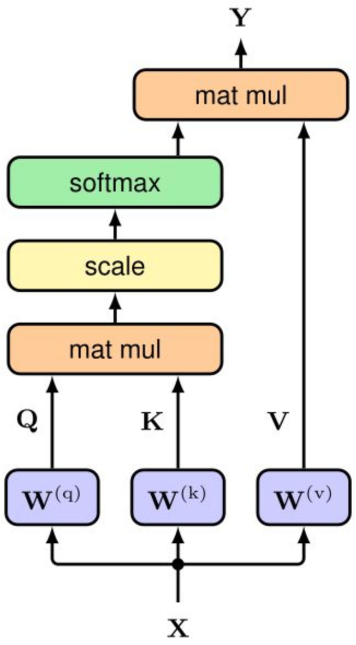


Self-Attention



https://jalammar.github.io/illustrated-transformer/

Scaled self-attention in one slide. One attention head



Algorithm 12.1: Scaled dot-product self-attention

Input: Set of tokens $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$

Weight matrices $\{\mathbf{W}^{(q)}, \mathbf{W}^{(k)}\} \in \mathbb{R}^{D \times D_k}$ and $\mathbf{W}^{(v)} \in \mathbb{R}^{D \times D_v}$

Output: $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \in \mathbb{R}^{N \times D_v} : \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$

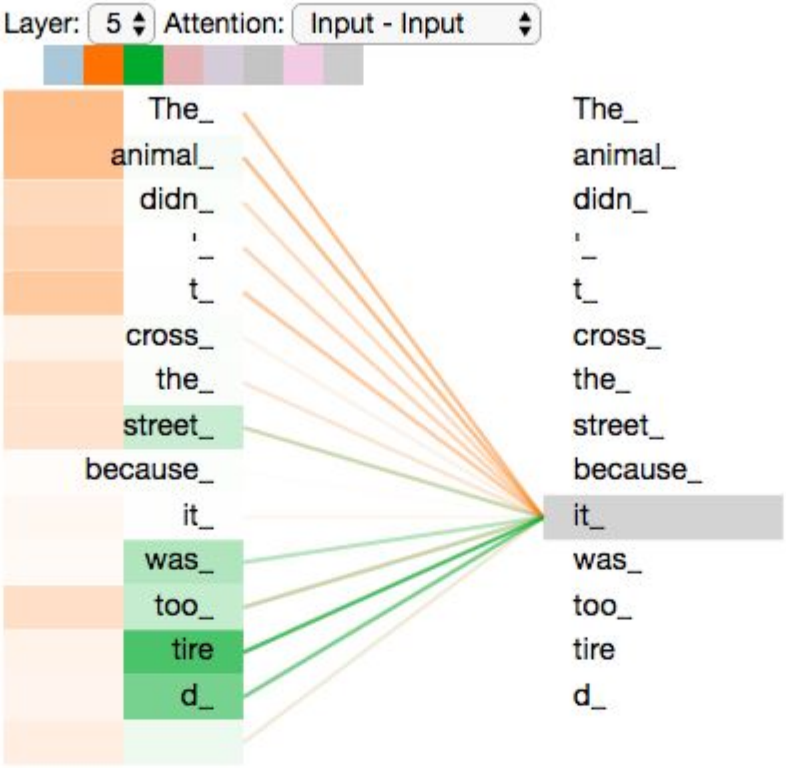
$\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$ `// compute queries` $\mathbf{Q} \in \mathbb{R}^{N \times D_k}$

$\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$ `// compute keys` $\mathbf{K} \in \mathbb{R}^{N \times D_k}$

$\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$ `// compute values` $\mathbf{V} \in \mathbb{R}^{N \times D}$

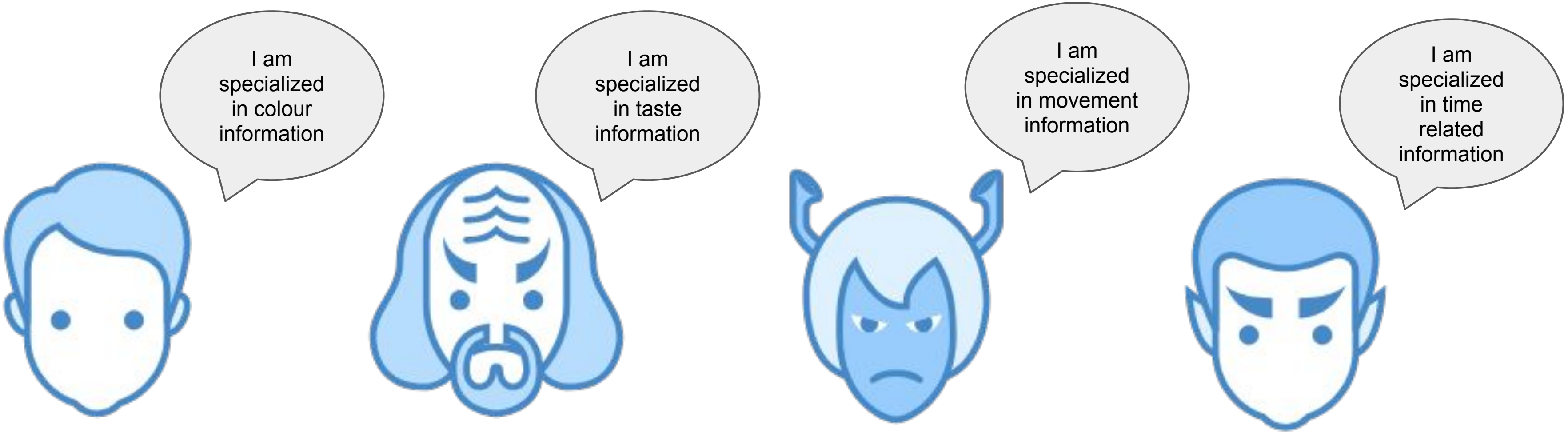
return $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left[\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{D_k}}\right]\mathbf{V}$

Multi head attention



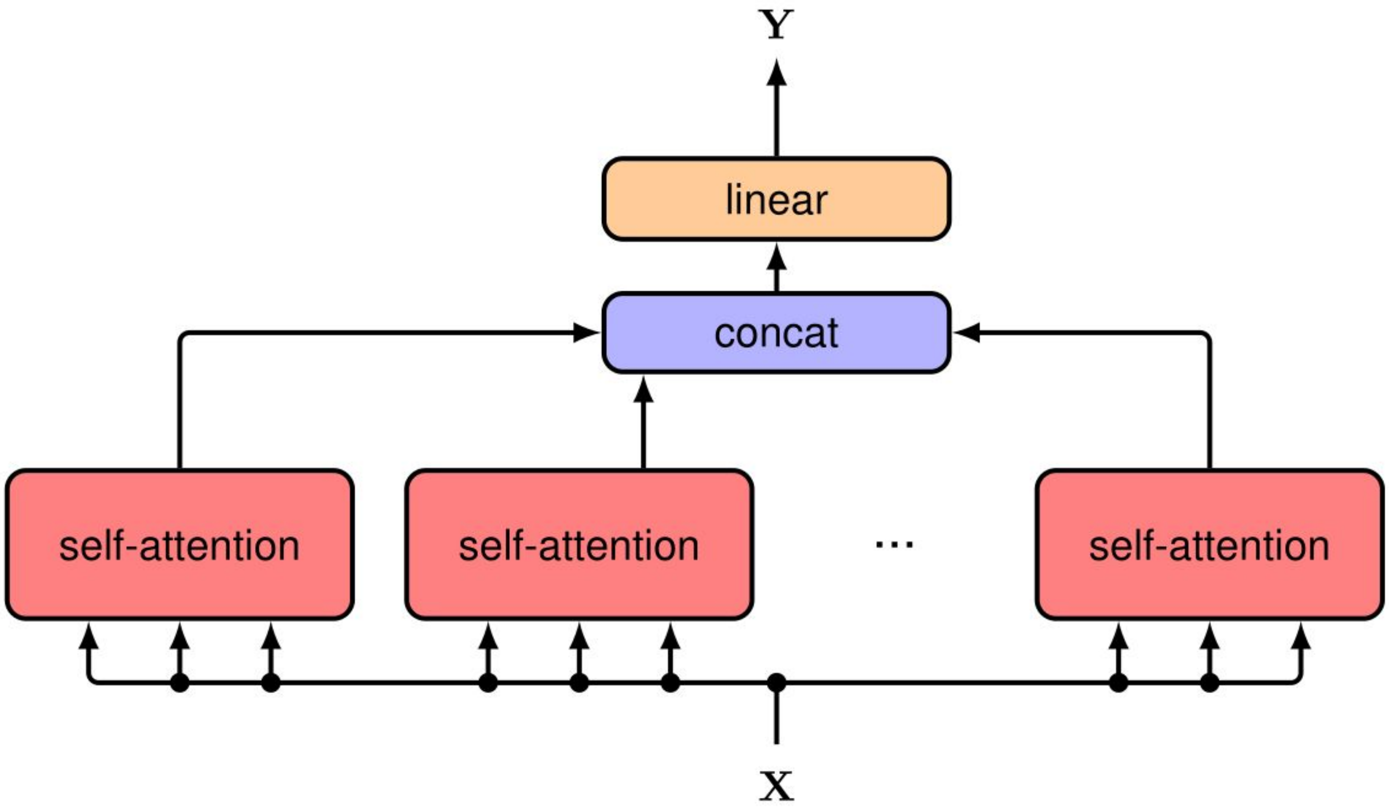
Multi head attention

Each attention head learns a specific set of parameters W

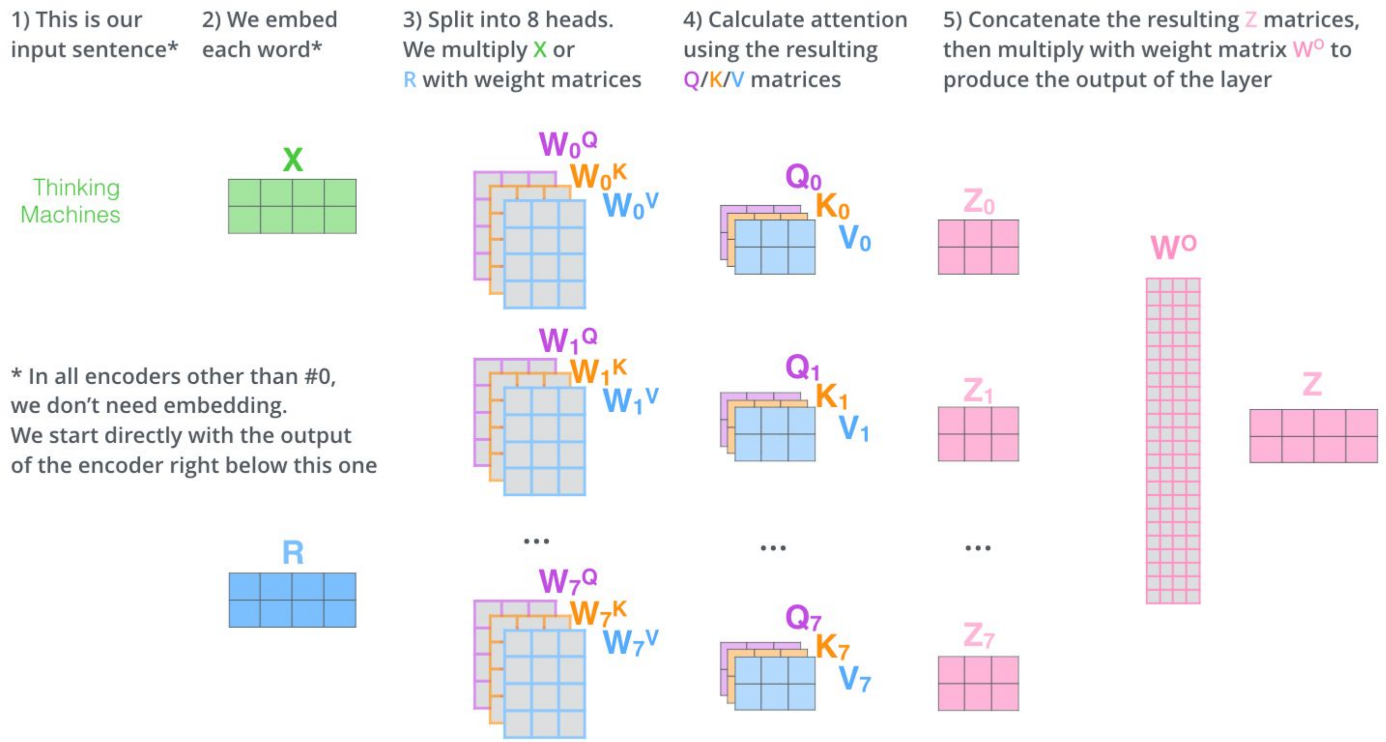


Together, we can run the galaxy!

Multi head attention



Multi-Head Attention



https://jalammar.github.io/illustrated-transformer/

Multi head attention

Algorithm 12.2: Multi-head attention

Input: Set of tokens $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
Query weight matrices $\{\mathbf{W}_1^{(\mathrm{q})}, \ldots, \mathbf{W}_H^{(\mathrm{q})}\} \in \mathbb{R}^{D \times D}$
Key weight matrices $\{\mathbf{W}_1^{(\mathrm{k})}, \ldots, \mathbf{W}_H^{(\mathrm{k})}\} \in \mathbb{R}^{D \times D}$
Value weight matrices $\{\mathbf{W}_1^{(\mathrm{v})}, \ldots, \mathbf{W}_H^{(\mathrm{v})}\} \in \mathbb{R}^{D \times D_\mathrm{v}}$
Output weight matrix $\mathbf{W}^{(\mathrm{o})} \in \mathbb{R}^{HD_\mathrm{v} \times D}$

Output: $\mathbf{Y} \in \mathbb{R}^{N \times D} : \{\mathbf{y}_1, \ldots, \mathbf{x}_N\}$

// compute self-attention for each head (Algorithm 12.1)

for $h = 1, \ldots, H$ **do**

$\quad \mathbf{Q}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{q})}, \quad \mathbf{K}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{k})}, \quad \mathbf{V}_h = \mathbf{X}\mathbf{W}_h^{(\mathrm{v})}$

$\quad \mathbf{H}_h = \mathrm{Attention}\left(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h\right)$ // $\mathbf{H}_h \in \mathbb{R}^{N \times D_\mathrm{v}}$

end for

$\mathbf{H} = \mathrm{Concat}\left[\mathbf{H_1}, \ldots, \mathbf{H_N}\right]$ // concatenate heads

return $\mathbf{Y}(\mathbf{X}) = \mathbf{H}\mathbf{W}^{(\mathrm{o})}$

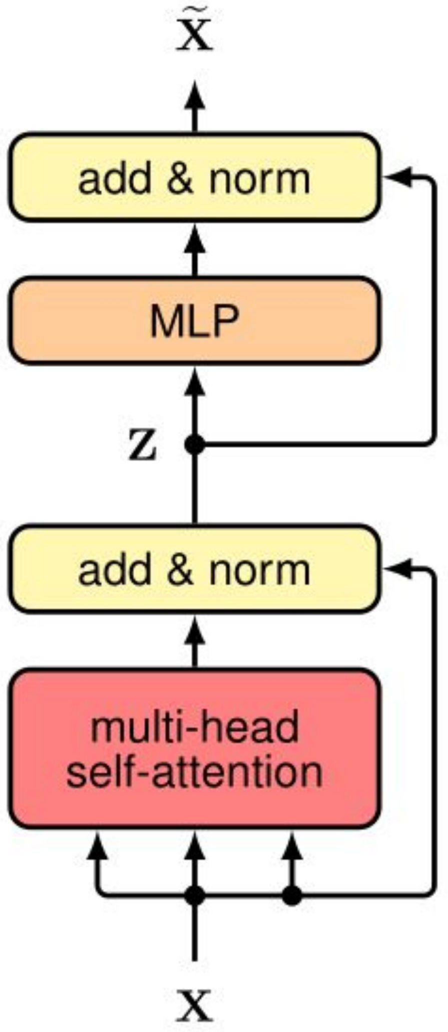
Transformer layer

Algorithm 12.3: Transformer layer

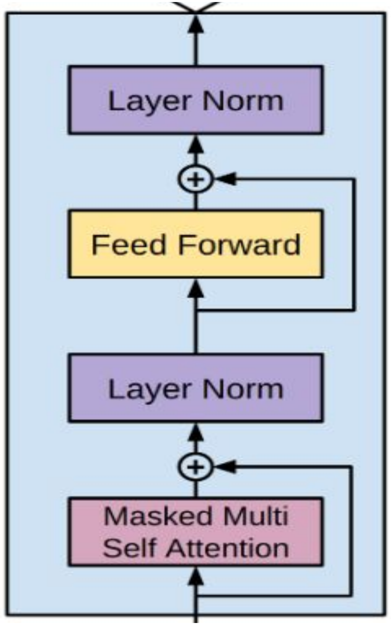
Input: Set of tokens $\mathbf{X} \in \mathbb{R}^{N \times D} : \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$
Multi-head self-attention layer parameters
Feed-forward network parameters
Output: $\widetilde{\mathbf{X}} \in \mathbb{R}^{N \times D} : \{\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_N\}$

$\mathbf{Z} = \text{LayerNorm}\left[\mathbf{Y}(\mathbf{X}) + \mathbf{X}\right]$ // $\mathbf{Y}(\mathbf{X})$ from Algorithm 12.2
$\widetilde{\mathbf{X}} = \text{LayerNorm}\left[\text{MLP}\left[\mathbf{Z}\right] + \mathbf{Z}\right]$ // shared neural network
return $\widetilde{\mathbf{X}}$

Y(X): Result of applying multi-head self-attention

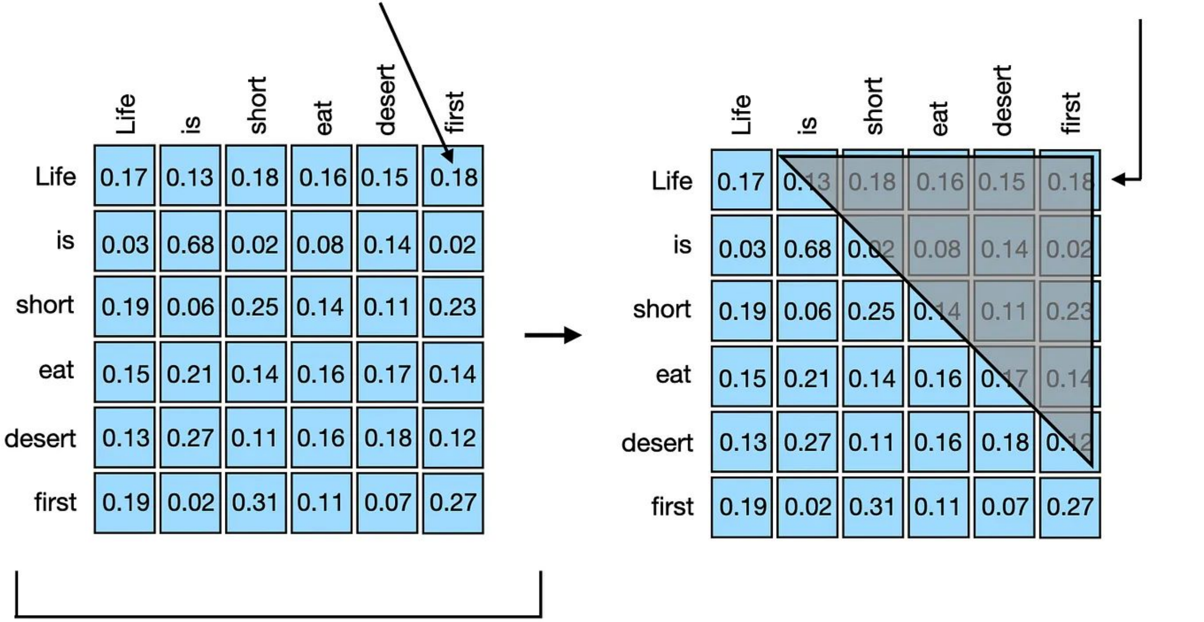


Masked transformer layer. A regular transformer layer but with masked attention.



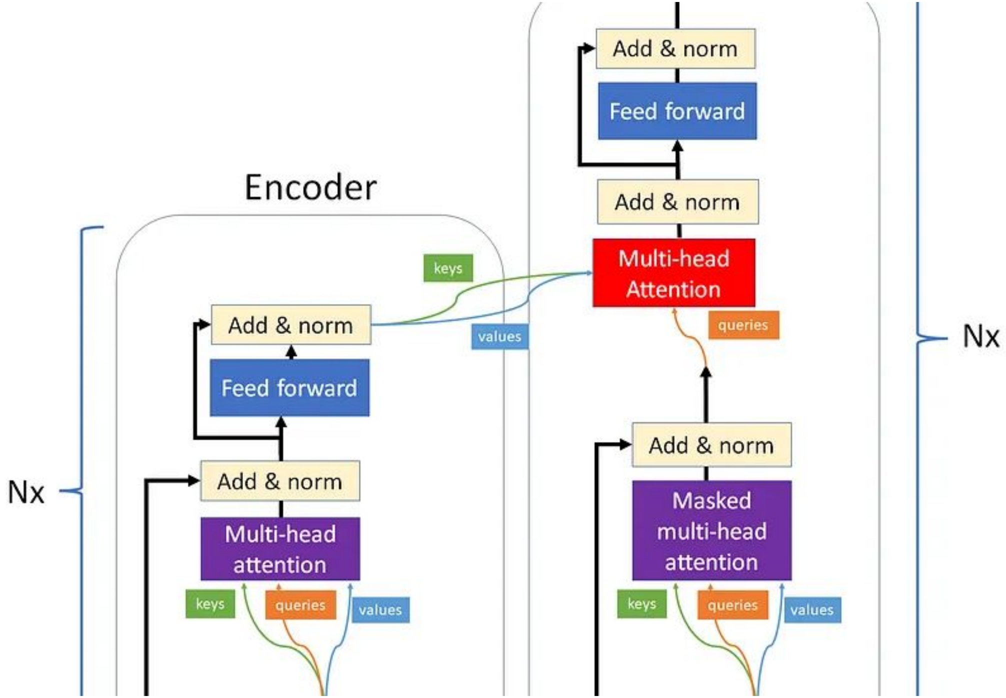
How much token “Life” attends to token “first”

In the row for corresponding to “Life”, mask out all words that come after “Life”

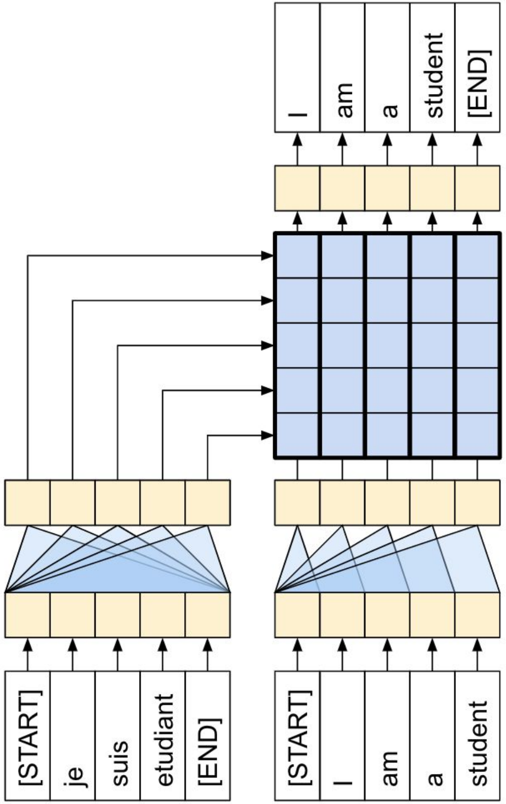


Attention weights calculated in the previous “Self-Attention” section

Cross-attention



Cross-Attention (e.g. Machine Translation)

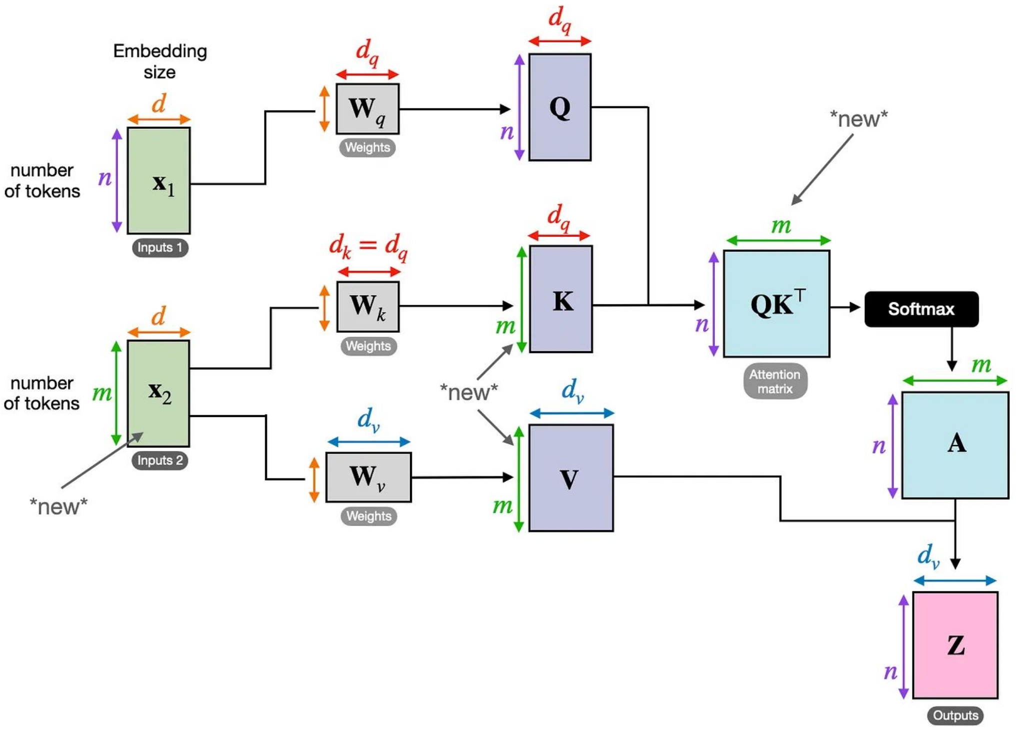


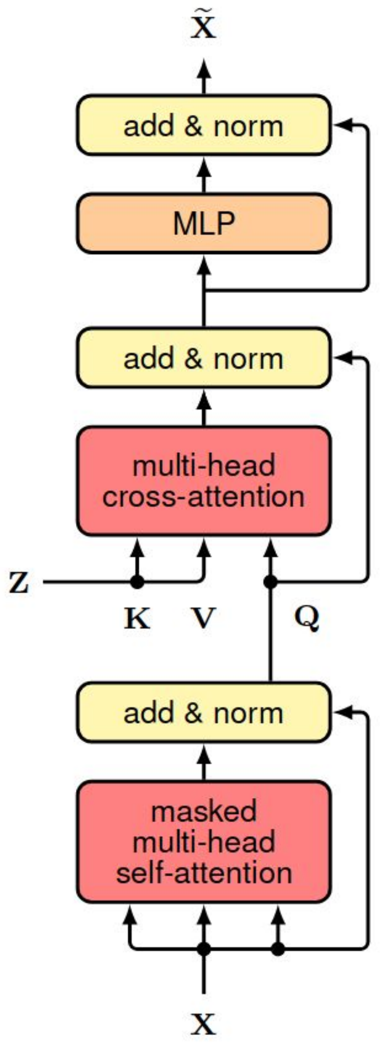
https://jalammar.github.io/illustrated-transformer/

Cross-attention transformer layer

Tokens entering through here determine the questions

Tokens entering through here determine the keys and the messages (values)





Transformer & Multi-Head Attention

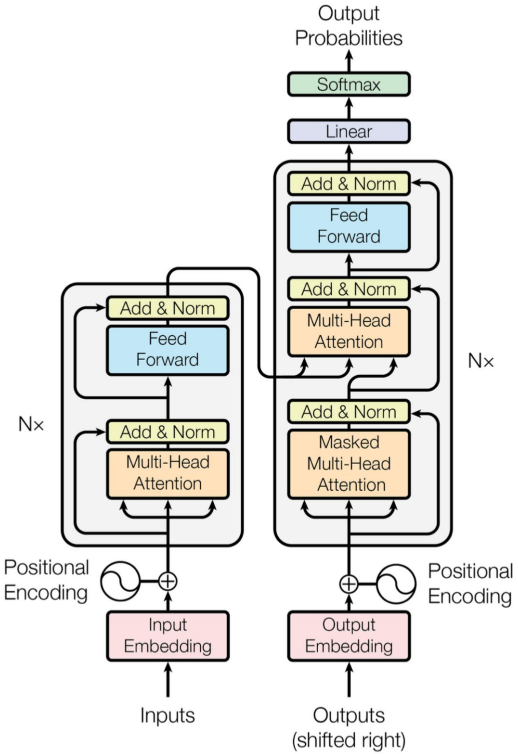


Figure 1: The Transformer - model architecture.

“Attention Is All You Need”
https://arxiv.org/abs/1706.03762

Transformers vs. RNNs

Challenges with RNNs	Transformers
● Long range dependencies ● Gradient vanishing and explosion ● Large # of training steps ● Sequential/recurrence → can't parallelize ● Complexity per layer: $O(n*d^2)$	● Can model long-range dependencies ● No gradient vanishing and explosion ● Fewer training steps ● Can parallelize computation! ● Complexity per layer: $O(n^2*d)$

Large Language Models

- Scaled up versions of Transformer architecture, e.g. millions/billions of parameters
- Typically trained on massive amounts of “general” textual data (e.g. web corpus)
- Training objective is typically “next token prediction”: $P(W_{t+1}|W_t,W_{t-1},...,W_1)$
- Emergent abilities as they scale up (e.g. chain-of-thought reasoning)
- Heavy computational cost (time, money, GPUs)
- Larger general ones: “plug-and-play” with few or zero-shot learning
  - Train once, then adapt to other tasks without needing to retrain
  - E.g. in-context learning and prompting

Emergent Abilities of Large Language Models

- Why do LLMs work so well? What happens as you scale up?
- Potential explanation: emergent abilities!
- An ability is emergent if it is present in larger but not smaller models
- Not have been directly predicted by extrapolating from smaller models
- Performance is near-random until a certain critical threshold, then improves heavily
  - Known as a “phase transition” and would not have been extrapolated

Wei et al., 2022. https://arxiv.org/abs/2206.07682

Few-Shot Prompting

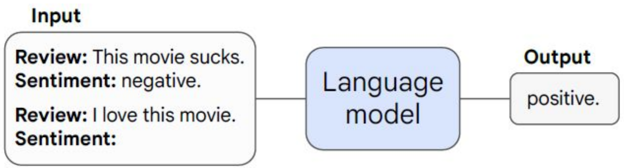


Figure 1: Example of an input and output for few-shot prompting.

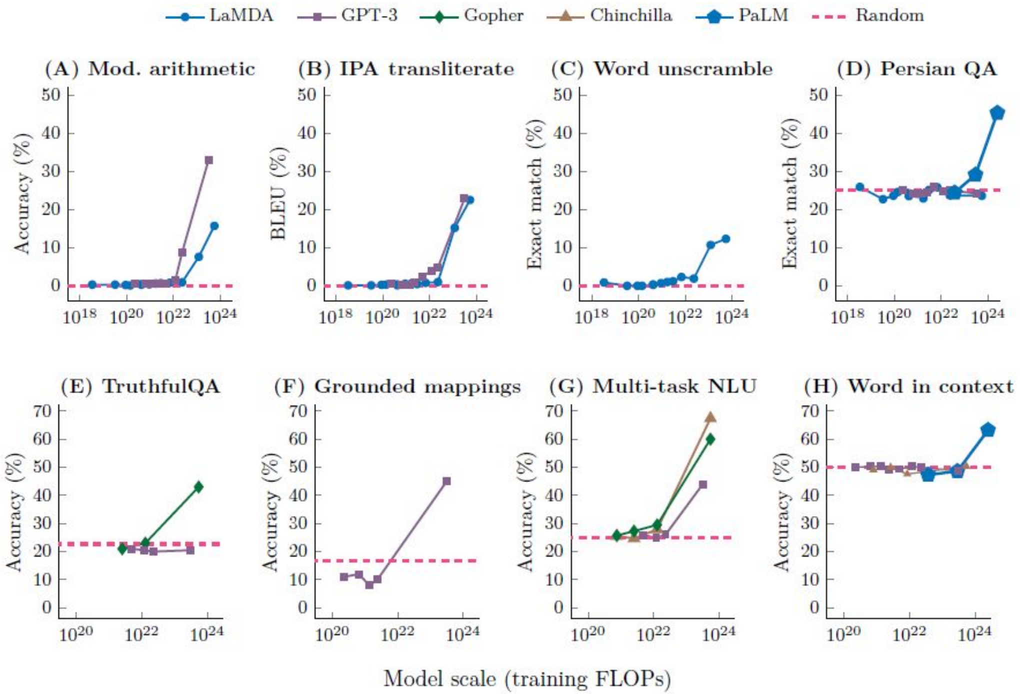


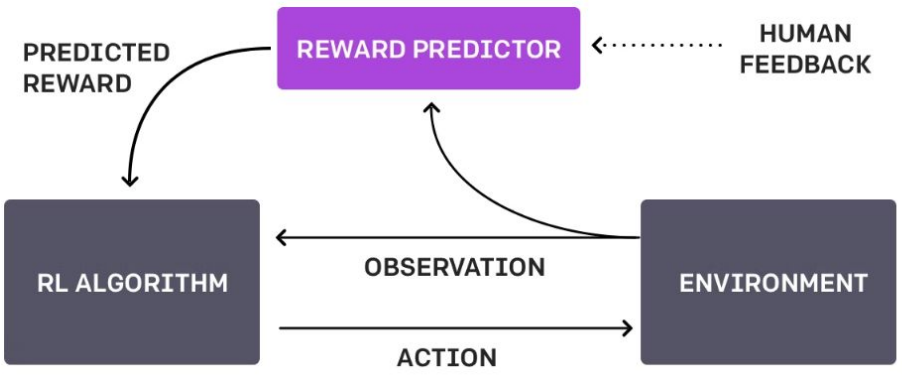
Figure 2: Eight examples of emergence in the few-shot prompting setting. Each point is a separate model. The ability to perform a task via few-shot prompting is emergent when a language model achieves random performance until a certain scale, after which performance significantly increases to well-above random. Note that models that used more training compute also typically have more parameters—hence, we show an analogous figure with number of model parameters instead of training FLOPs as the x-axis in Figure 11. A–D: BIG-Bench (2022), 2-shot. E: Lin et al. (2021) and Rae et al. (2021). F: Patel & Pavlick (2022). G: Hendrycks et al. (2021a), Rae et al. (2021), and Hoffmann et al. (2022). H: Brown et al. (2020), Hoffmann et al. (2022), and Chowdhery et al. (2022) on the WiC benchmark (Pilehvar & Camacho-Collados, 2019).

Potential Explanations of Emergence

- Currently few explanations for why these abilities emerge
- Evaluation metrics used to measure these abilities may not fully explain why they emerge
- Disclaimer: maybe emergent abilities of LLMs are a mirage!!!
  - https://arxiv.org/abs/2304.15004
  - *“Emergent abilities appear due to the researcher's choice of metric rather than due to fundamental changes in model behavior with scale”*

Reinforcement Learning with Human Feedback (RLHF)

- RLHF: technique that trains a "reward model" directly from human feedback
- Uses the model as a reward function to optimize an agent's policy using reinforcement learning (RL) through an optimization algorithm
- Ask humans to rank instances of the agent's behavior, e.g. which produced response is better



Direct Preference Optimization (DPO)

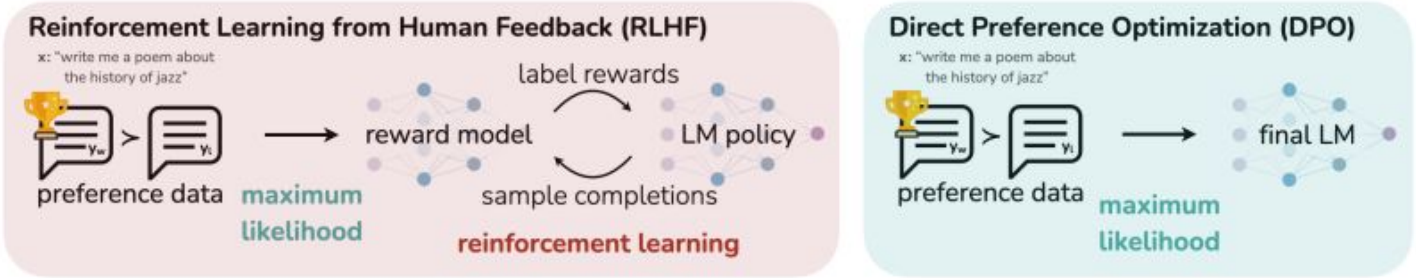


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL.

https://arxiv.org/pdf/2305.18290.pdf

GPT-4

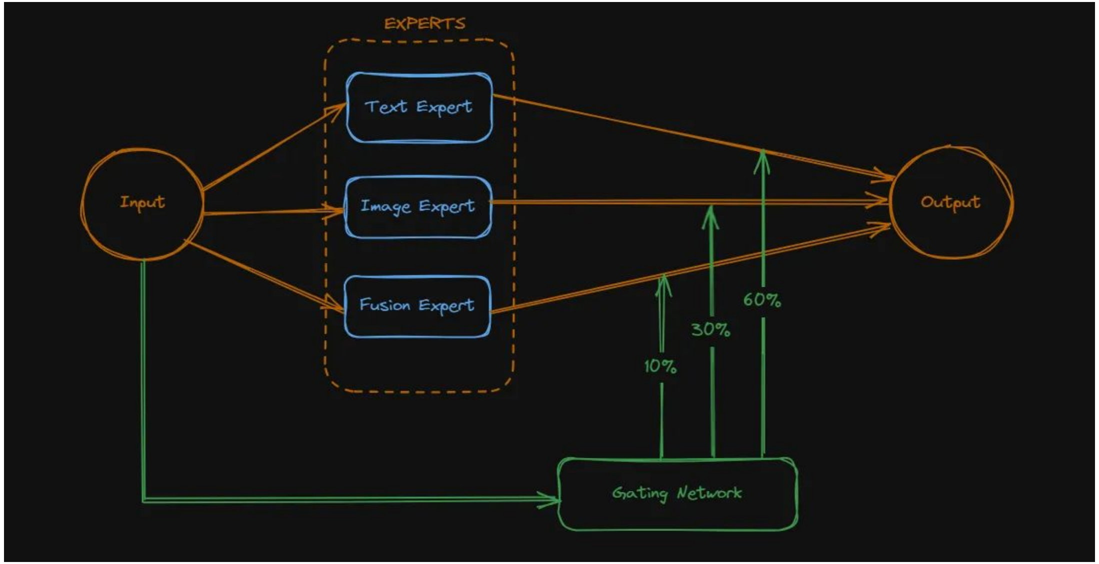
- Supervised learning on large dataset, then RLHF and RLAIF
- GPT-4 trained on both images and text, vision is also out!
  - Discuss humor in images, summarize screenshot text, etc.
- GPT-4 is "more reliable, creative, and able to handle much more nuanced instructions than GPT-3.5”
- Much longer context windows of 8,192 and 32,768 tokens
- Does exceptionally well on standardized tests
- Did not release technical details of GPT-4

Gemini

- Latest: **Gemini 1.5 Pro**
- Gemini Ultra performs better than ChatGPT on 30 of the 32 academic benchmarks in reasoning and understanding it tested on.
- Effectively processes and integrates data from diff modalities:
  - Text, audio, image, video
- Based on a Mixture-of-Experts (MoE) model
  - Significantly improves efficiency in training and application

Gemini

- Based on a Mixture-of-Experts (MoE) model
  - Combination of multiple small Neural networks known as 'Experts' which are trained and capable of handling particular data and performing specialised tasks.
  - 'Gating network' which predicts which response is best suited to address the request.



Source link

Where we are (2024)

Recently Taken Off:

- LLM boom: ChatGPT, GPT-4, Gemini, open-source models
- Human alignment and interaction
  - Reinforcement learning & human feedback
- Controlling toxicity, bias, and ethics
- More use in unique applications: audio, art/music, neuro/bio, coding, games, physical tasks, etc.
- Other: diffusion models (e.g. text-to-image/video gen)
  - Also, Diffusion Transformer (DiT)

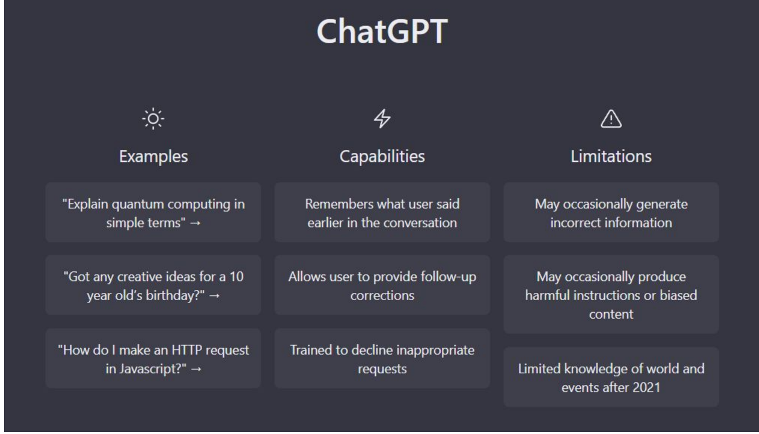
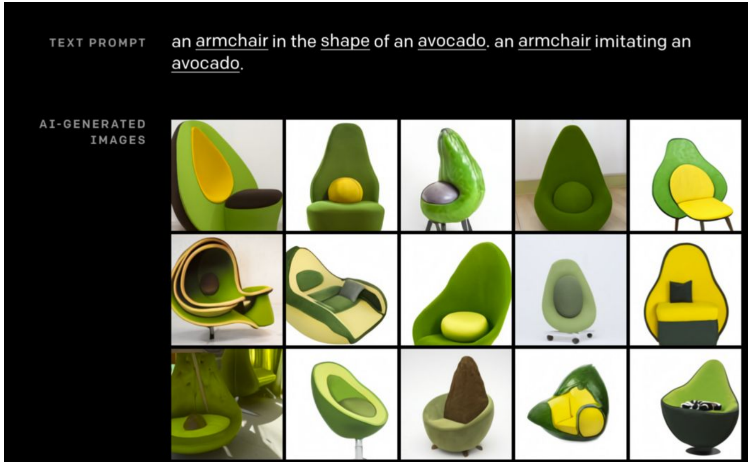


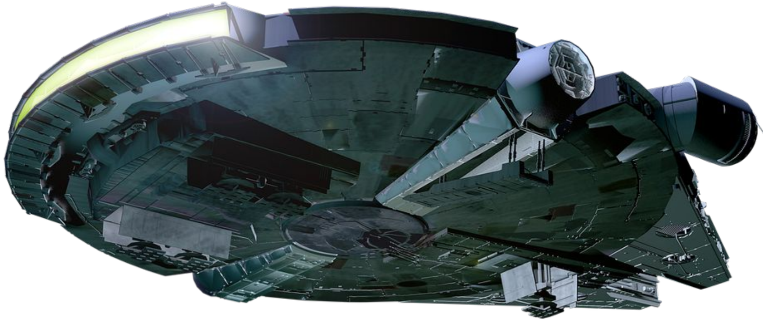
Image source: https://openai.com/blog/chatgpt/



The Future (What's Next?)

- Can enable a lot more applications:
  - Generalist Agents
  - Longer video understanding and generation, finance + business
  - Incredibly long sequence modeling (GPT authors a novel)
  - Domain-specific "Foundation models" - DoctorGPT, LawyerGPT, …
  - Potential real-world impacts:
    - Personalized education and tutoring systems
    - Advanced healthcare diagnostics, environmental monitoring & protection, etc.
    - Real-time multilingual communication
    - Interactive entertainment & gaming (e.g. NPCs)

The Future (What’s Missing?)



- Missing Ingredients (to AGI/ASI?):
    - Reducing computation complexity
    - Enhanced human controllability
    - Alignment with language models of human brain
    - Adaptive learning and generalization across domains
    - Multi-sensory multimodal embodiment (e.g. intuitive physics and commonsense)
    - Infinite/external memory: like Neural Turing Machines
    - Infinite/constant self-improvement and self-reflection capabilities
    - Complete autonomy and long-horizon decision-making
    - Emotional intelligence and social understanding
    - Ethical reasoning and value alignment

Questions?

Hugging face

https://medium.com/@anthony.demeusy/introduction-to-hugging-face-a-starters-guide-to-using-online-models-d7d7923a9aa5

Transformers and LLMs: An Introduction

Challenges and Weaknesses

Challenges of NLP:

- Discrete nature of text
- More difficult data augmentation
- Text is “precise” - one wrong word changes entire meaning of a sentence
- Potential for long context lengths and memories (e.g. conversations)
- Many more…

Weaknesses of earlier models/apporoaches:

- Short context length
- “Linear” reasoning - no attention mechanism to focus on other parts
- Earlier approaches (e.g. word2vec) do not adapt based on context

NLP Throughout the Years

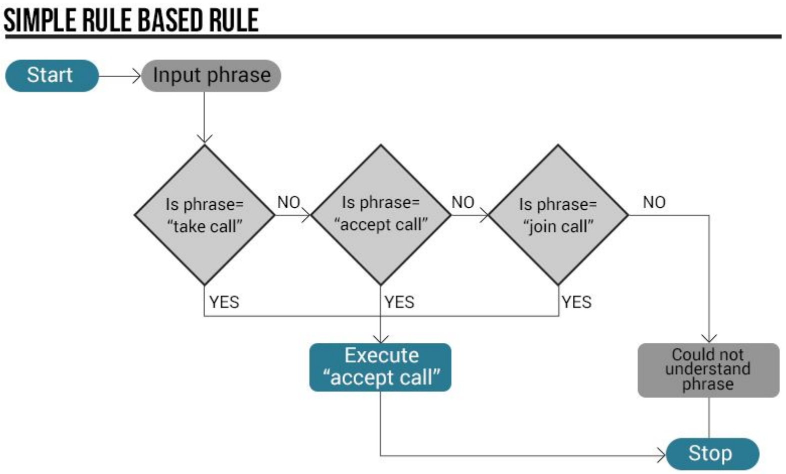
Rule Based NLP Systems

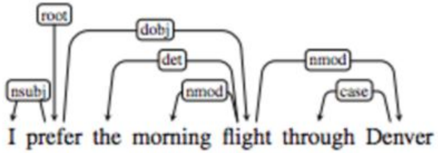
```
  Welcome to
                    EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

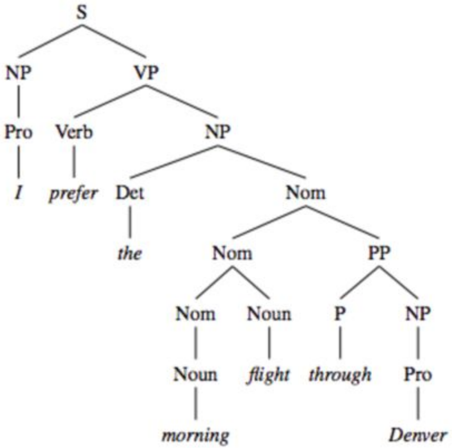
Eliza Chatbot (1966), MIT by Joseph Weizenbaum

Linguistic Foundations

- ▶ Rule-based approaches
- ▶ Semantic parsing
- ▶ Analyzing linguistic structure and grammars of text

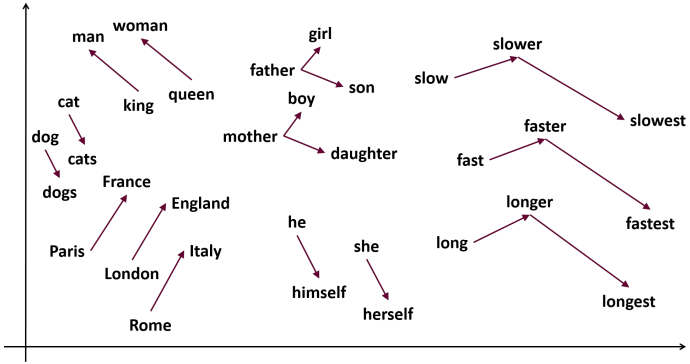




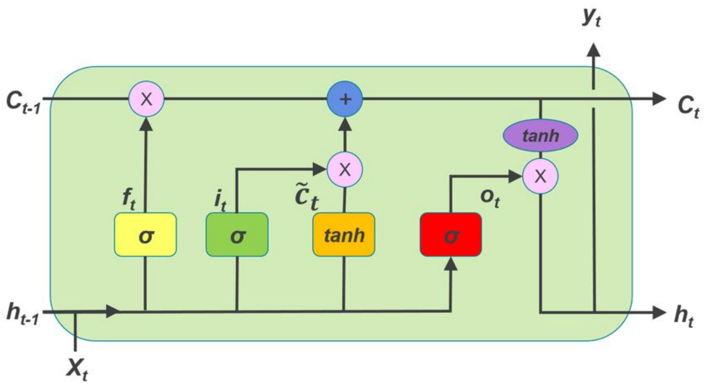


Word Embeddings

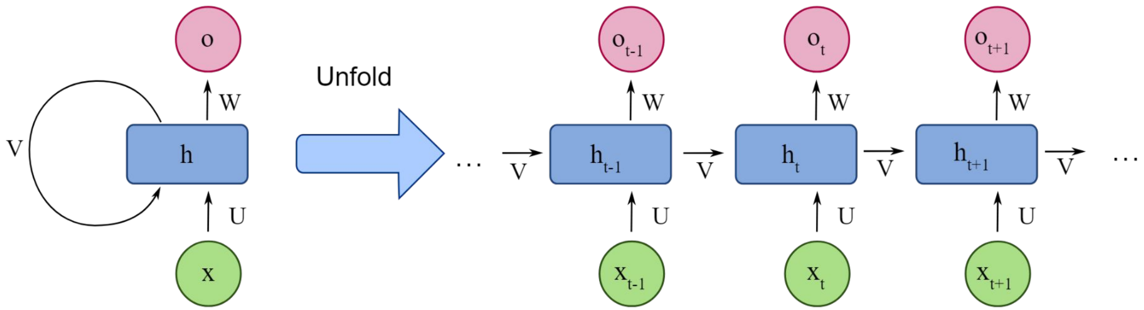
- Represent each word as a “vector” of numbers
- Converts a “discrete” representation to “continuous”, allowing for:
  - More “fine-grained” representations of words
  - Useful computations such as cosine/eucl distance
  - Visualization and mapping of words onto a semantic space
- Examples:
  - Word2Vec (2013), GloVe, BERT, ELMo



Seq2seq Models



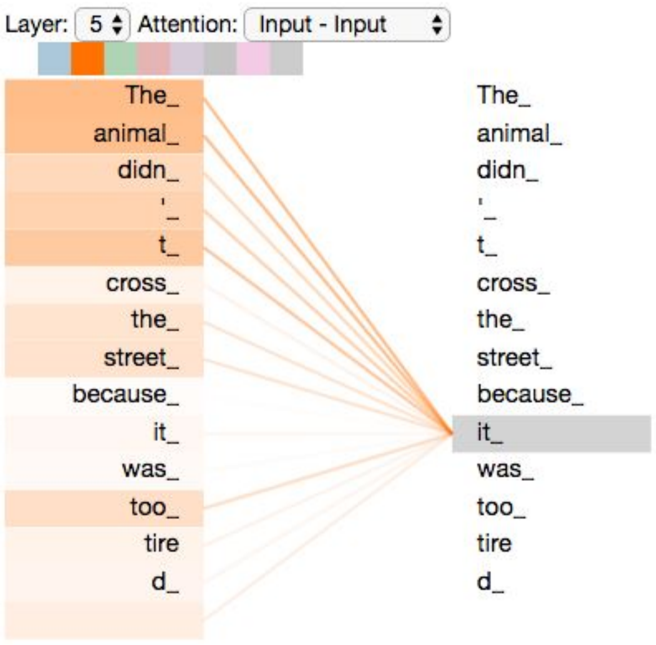
- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory Networks (LSTMs)
- “Dependency” and info between tokens
- Gates to “control memory” and flow of information



Attention and Transformers

- Allows to “focus attention” on particular aspects of the input text
- Done by using a set of parameters, called "weights," that determine how much attention should be paid to each input at each time step
- These weights are computed using a combination of the input and the current hidden state of the model
- Attention weights are computed (dot product of the query, key and value matrix), then a softmax function is applied to the dot product

$$\mathrm{attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$



https://arxiv.org/abs/1706.03762
https://jalammar.github.io/illustrated-transformer/

Analogy for Q, K, V

- Library system
- Imagine you're looking for information on a specific topic (query)
- Each book in the library has a summary (key) that helps identify if it contains the information you're looking for
- Once you find a match between your query and a summary, you access the book to get the detailed information (value) you need
- Here, in Attention, we do a “soft match” across multiple values, e.g. get info from multiple books (“book 1 is most relevant, then book 2, then book 3, etc.”)

$$\text{attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Attention and Transformers

- Attention weights used to compute the context vector, which is a weighted sum of the input at different positions
- Context vector is used to update the hidden state of the model, which is used to generate the final output
- "Pay attention" to different parts of the input, depending on the task at hand → more accurate and natural-sounding output, esp. when working with longer inputs (e.g. paragraphs)

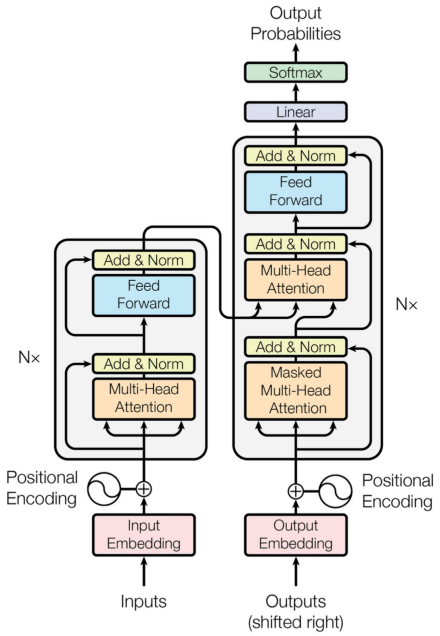
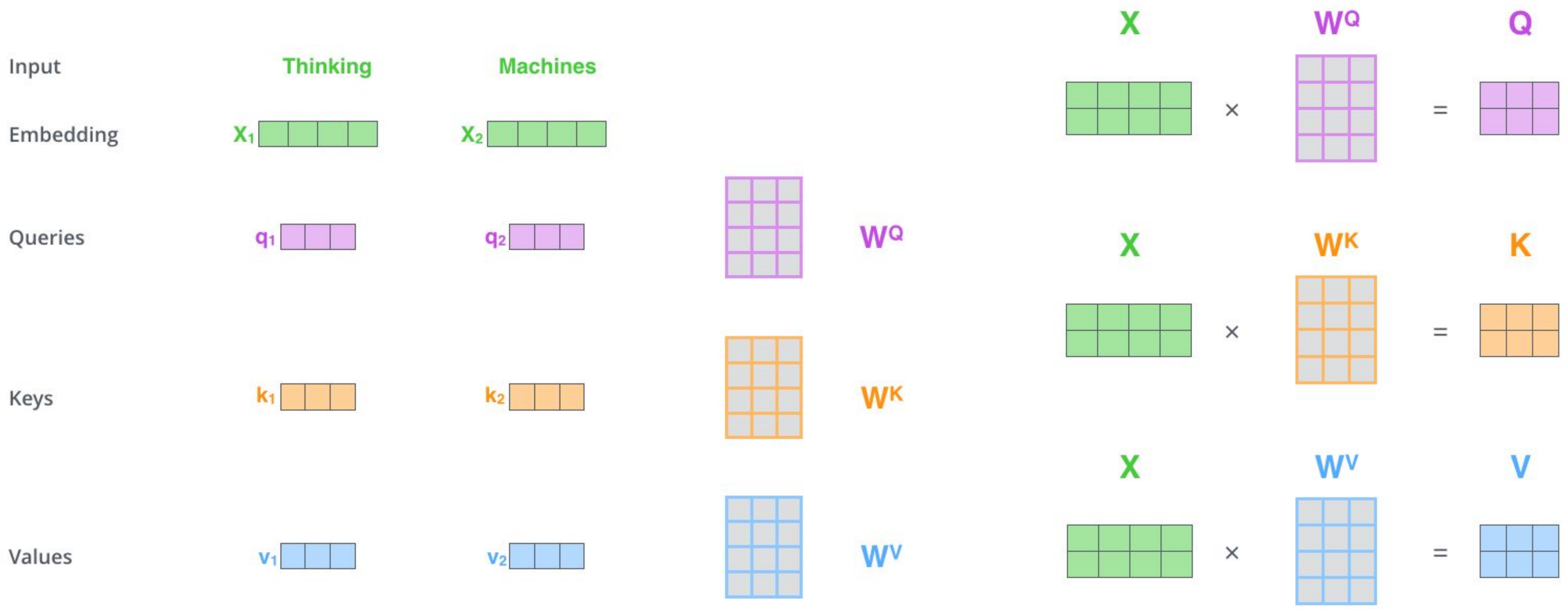


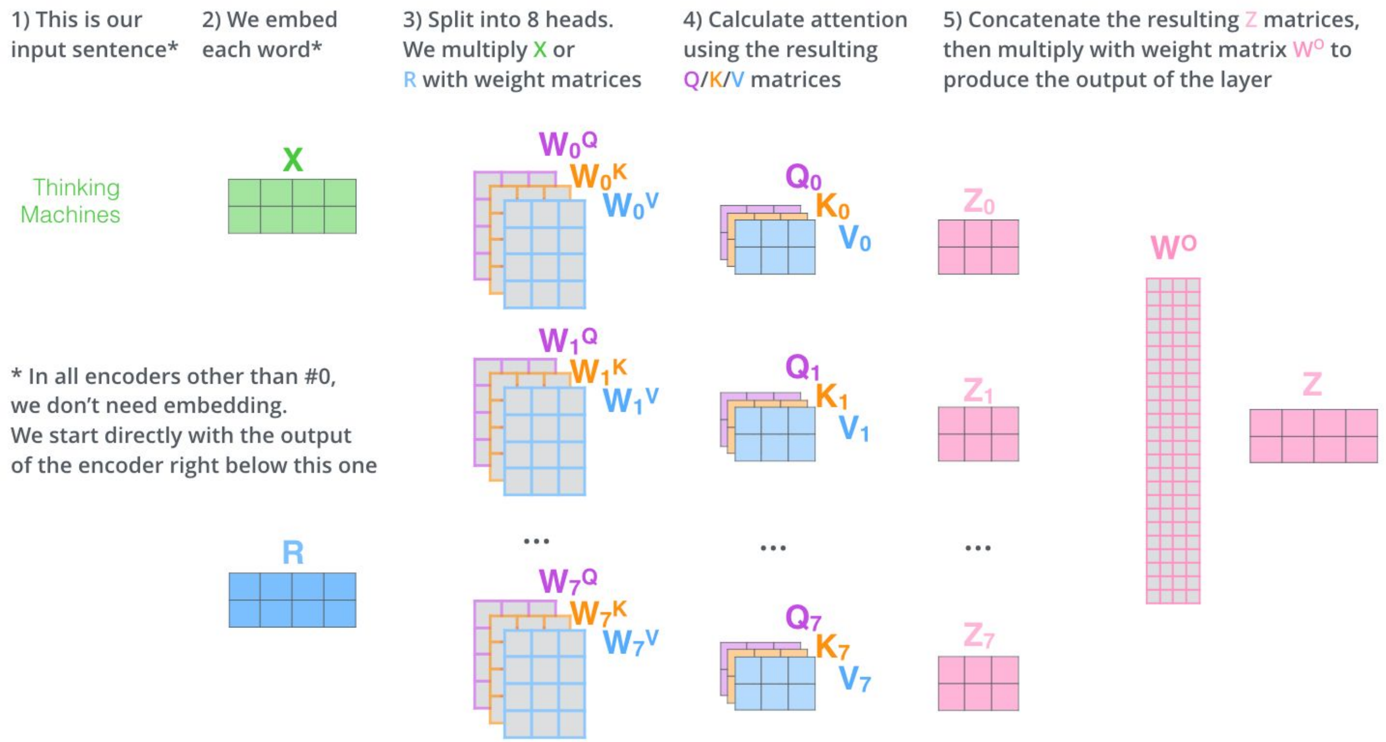
Figure 1: The Transformer - model architecture.

Self-Attention



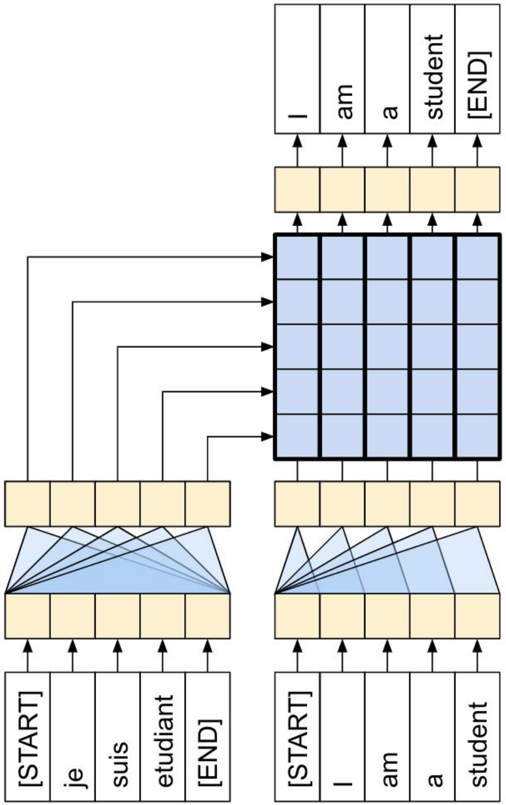
https://jalammar.github.io/illustrated-transformer/

Multi-Head Attention



https://jalammar.github.io/illustrated-transformer/

Cross-Attention (e.g. Machine Translation)



https://jalammar.github.io/illustrated-transformer/

Transformers vs. RNNs

Challenges with RNNs	Transformers
● Long range dependencies ● Gradient vanishing and explosion ● Large # of training steps ● Sequential/recurrence → can't parallelize ● Complexity per layer: $O(n*d^2)$	● Can model long-range dependencies ● No gradient vanishing and explosion ● Fewer training steps ● Can parallelize computation! ● Complexity per layer: $O(n^2*d)$

Large Language Models

- Scaled up versions of Transformer architecture, e.g. millions/billions of parameters
- Typically trained on massive amounts of “general” textual data (e.g. web corpus)
- Training objective is typically “next token prediction”: $P(W_{t+1}|W_t,W_{t-1},...,W_1)$
- Emergent abilities as they scale up (e.g. chain-of-thought reasoning)
- Heavy computational cost (time, money, GPUs)
- Larger general ones: “plug-and-play” with few or zero-shot learning
  - Train once, then adapt to other tasks without needing to retrain
  - E.g. in-context learning and prompting

Emergent Abilities of Large Language Models

- Why do LLMs work so well? What happens as you scale up?
- Potential explanation: emergent abilities!
- An ability is emergent if it is present in larger but not smaller models
- Not have been directly predicted by extrapolating from smaller models
- Performance is near-random until a certain critical threshold, then improves heavily
  - Known as a “phase transition” and would not have been extrapolated

Wei et al., 2022. https://arxiv.org/abs/2206.07682

Few-Shot Prompting

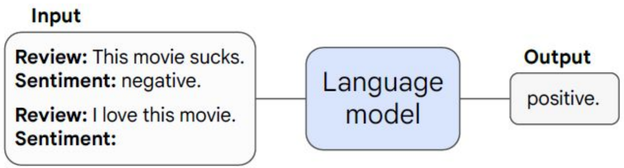


Figure 1: Example of an input and output for few-shot prompting.

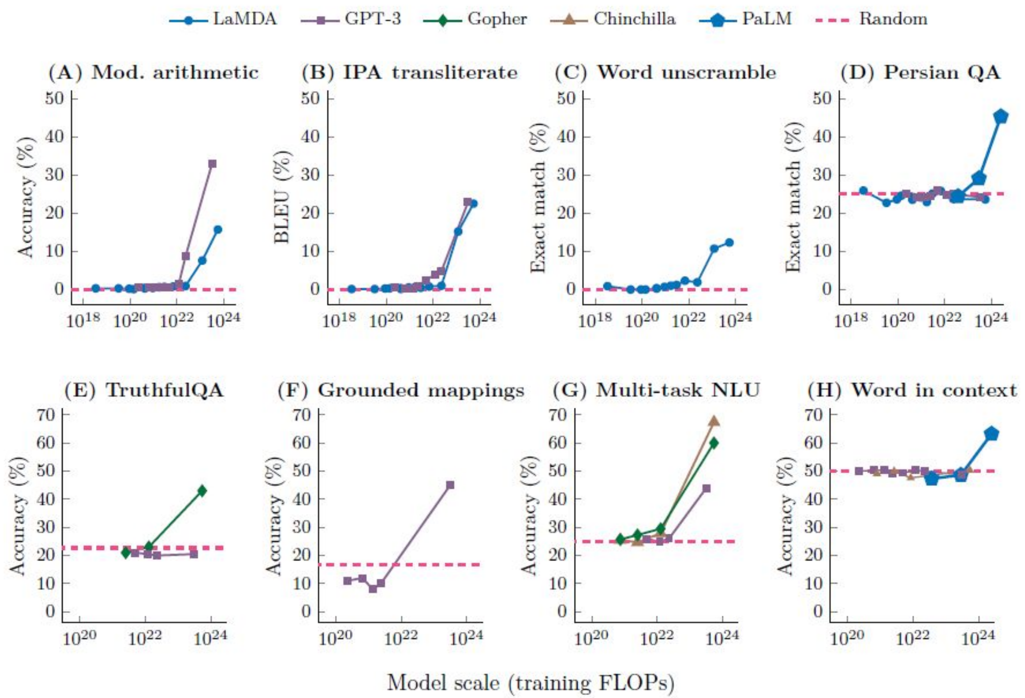


Figure 2: Eight examples of emergence in the few-shot prompting setting. Each point is a separate model. The ability to perform a task via few-shot prompting is emergent when a language model achieves random performance until a certain scale, after which performance significantly increases to well-above random. Note that models that used more training compute also typically have more parameters—hence, we show an analogous figure with number of model parameters instead of training FLOPs as the x-axis in Figure 11. A–D: BIG-Bench (2022), 2-shot. E: Lin et al. (2021) and Rae et al. (2021). F: Patel & Pavlick (2022). G: Hendrycks et al. (2021a), Rae et al. (2021), and Hoffmann et al. (2022). H: Brown et al. (2020), Hoffmann et al. (2022), and Chowdhery et al. (2022) on the WiC benchmark (Pilehvar & Camacho-Collados, 2019).

Potential Explanations of Emergence

- Currently few explanations for why these abilities emerge
- Evaluation metrics used to measure these abilities may not fully explain why they emerge
- Disclaimer: maybe emergent abilities of LLMs are a mirage!!!
  - https://arxiv.org/abs/2304.15004
  - *“Emergent abilities appear due to the researcher's choice of metric rather than due to fundamental changes in model behavior with scale”*

Beyond Scaling

- Further scaling could endow even-larger LMs with new emergent abilities
- While scaling is a factor in emergent abilities, it is not the only factor
- E.g. new architectures, higher-quality data, and improved training procedures, could enable emergent abilities on smaller models
- Further research may make the abilities available for smaller models
- Other directions: improving few-shot prompting abilities of LMs, theoretical and interpretability research, and computational linguistics work

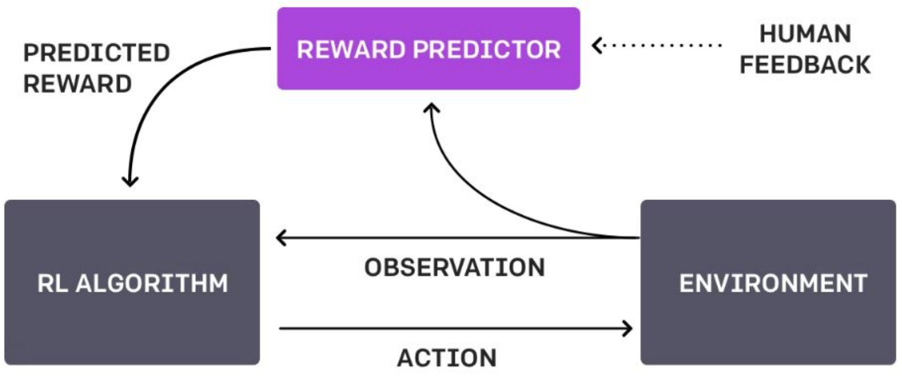
Questions for the Group

- Do you believe emergent abilities will continue to arise with more scale? Will there be a limit? Possibly even diminishing returns?
- What are your thoughts on the current trend of larger models and more data? Do you believe this is a good direction for the research community, or rather “inhibiting our creativity”?
- Thoughts on retrieval-based or retrieval-augmented systems compared to simply “learning everything” within the parameters of the model?

RLHF, ChatGPT, GPT-4, Gemini

Reinforcement Learning with Human Feedback (RLHF)

- RLHF: technique that trains a "reward model" directly from human feedback
- Uses the model as a reward function to optimize an agent's policy using reinforcement learning (RL) through an optimization algorithm
- Ask humans to rank instances of the agent's behavior, e.g. which produced response is better



Direct Preference Optimization (DPO)

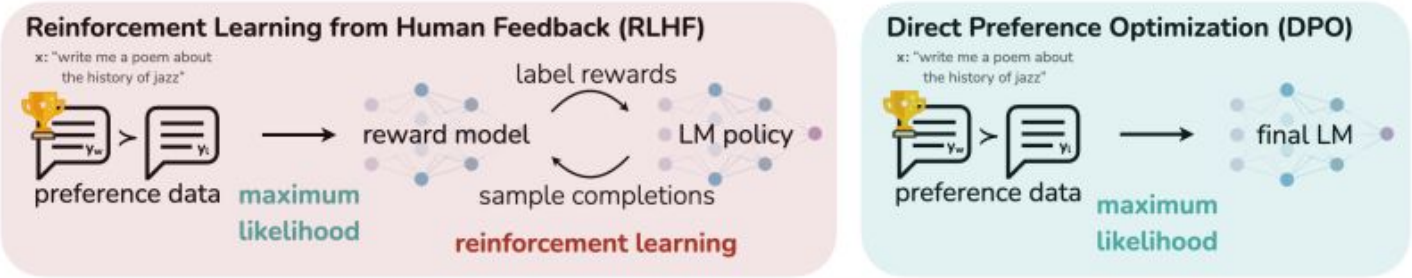
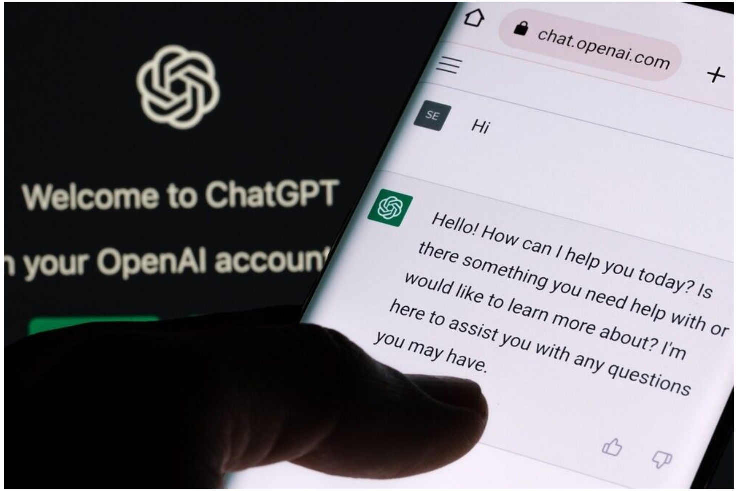


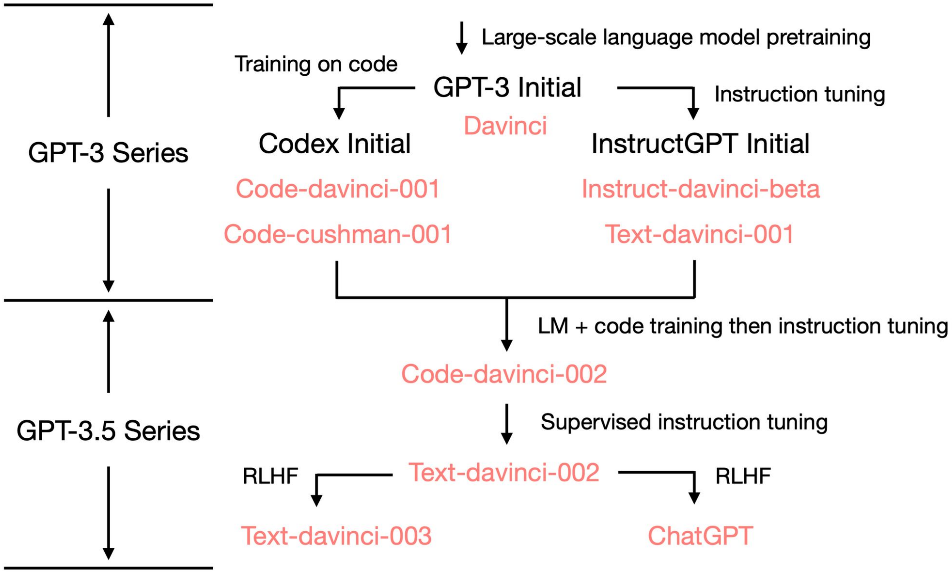
Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, without an explicit reward function or RL.

https://arxiv.org/pdf/2305.18290.pdf

ChatGPT

- Finetuned on GPT-3.5, which is a series of models trained on a mix of text and code using instruction tuning and RLHF
- Taken the world by storm!





GPT-4

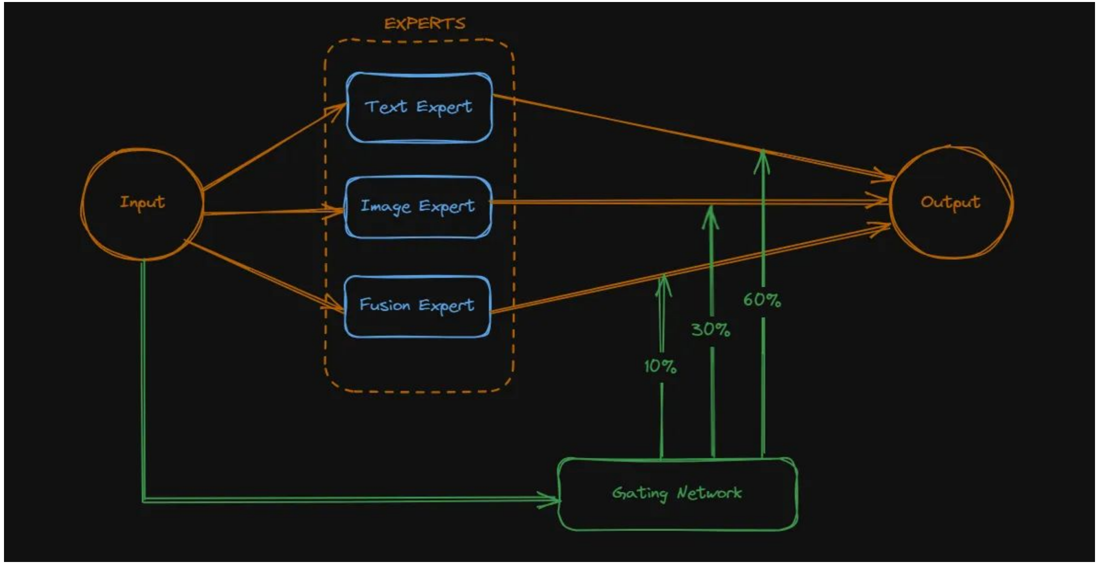
- Supervised learning on large dataset, then RLHF and RLAIF
- GPT-4 trained on both images and text, vision is also out!
  - Discuss humor in images, summarize screenshot text, etc.
- GPT-4 is "more reliable, creative, and able to handle much more nuanced instructions than GPT-3.5"
- Much longer context windows of 8,192 and 32,768 tokens
- Does exceptionally well on standardized tests
- Did not release technical details of GPT-4

Gemini

- Latest: **Gemini 1.5 Pro**
- Gemini Ultra performs better than ChatGPT on 30 of the 32 academic benchmarks in reasoning and understanding it tested on.
- Effectively processes and integrates data from diff modalities:
  - Text, audio, image, video
- Based on a Mixture-of-Experts (MoE) model
  - Significantly improves efficiency in training and application

Gemini

- Based on a Mixture-of-Experts (MoE) model
  - Combination of multiple small Neural networks known as 'Experts' which are trained and capable of handling particular data and performing specialised tasks.
  - 'Gating network' which predicts which response is best suited to address the request.



[Source link]

Gemini 1.5

Where we are (2024)

Recently Taken Off:

- LLM boom: ChatGPT, GPT-4, Gemini, open-source models
- Human alignment and interaction
  - Reinforcement learning & human feedback
- Controlling toxicity, bias, and ethics
- More use in unique applications: audio, art/music, neuro/bio, coding, games, physical tasks, etc.
  - Speakers will touch (or have touched) on these!
- Other: diffusion models (e.g. text-to-image/video gen)
  - Also, Diffusion Transformer (DiT)

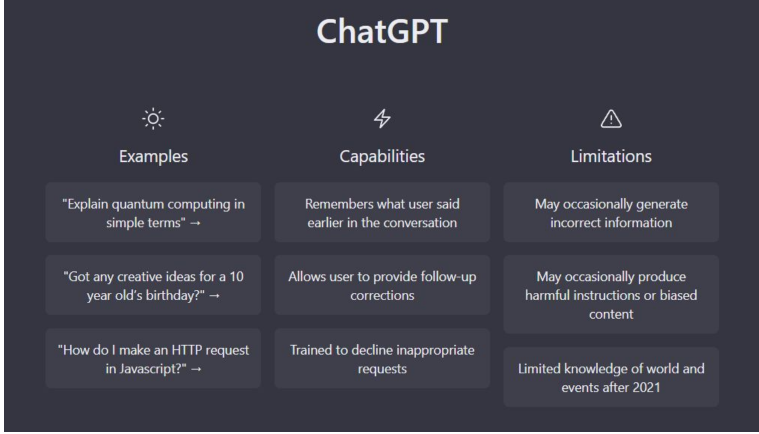
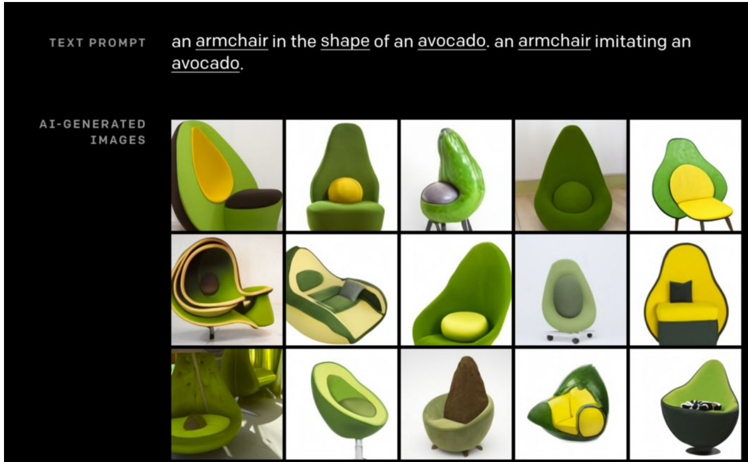


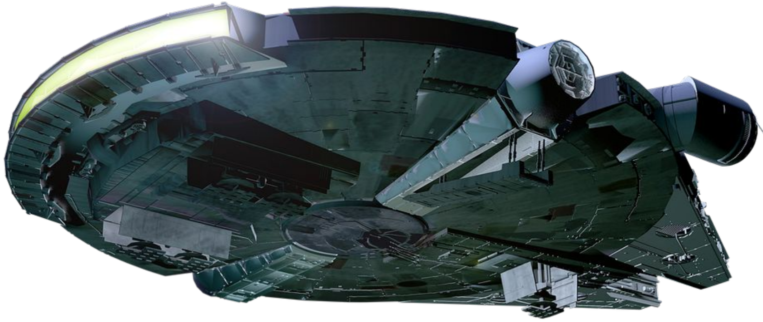
Image source: https://openai.com/blog/chatgpt/



The Future (What's Next?)

- Can enable a lot more applications:
  - Generalist Agents
  - Longer video understanding and generation, finance + business
  - Incredibly long sequence modeling (GPT authors a novel)
  - Domain-specific "Foundation models" - DoctorGPT, LawyerGPT, …
  - Potential real-world impacts:
    - Personalized education and tutoring systems
    - Advanced healthcare diagnostics, environmental monitoring & protection, etc.
    - Real-time multilingual communication
    - Interactive entertainment & gaming (e.g. NPCs)

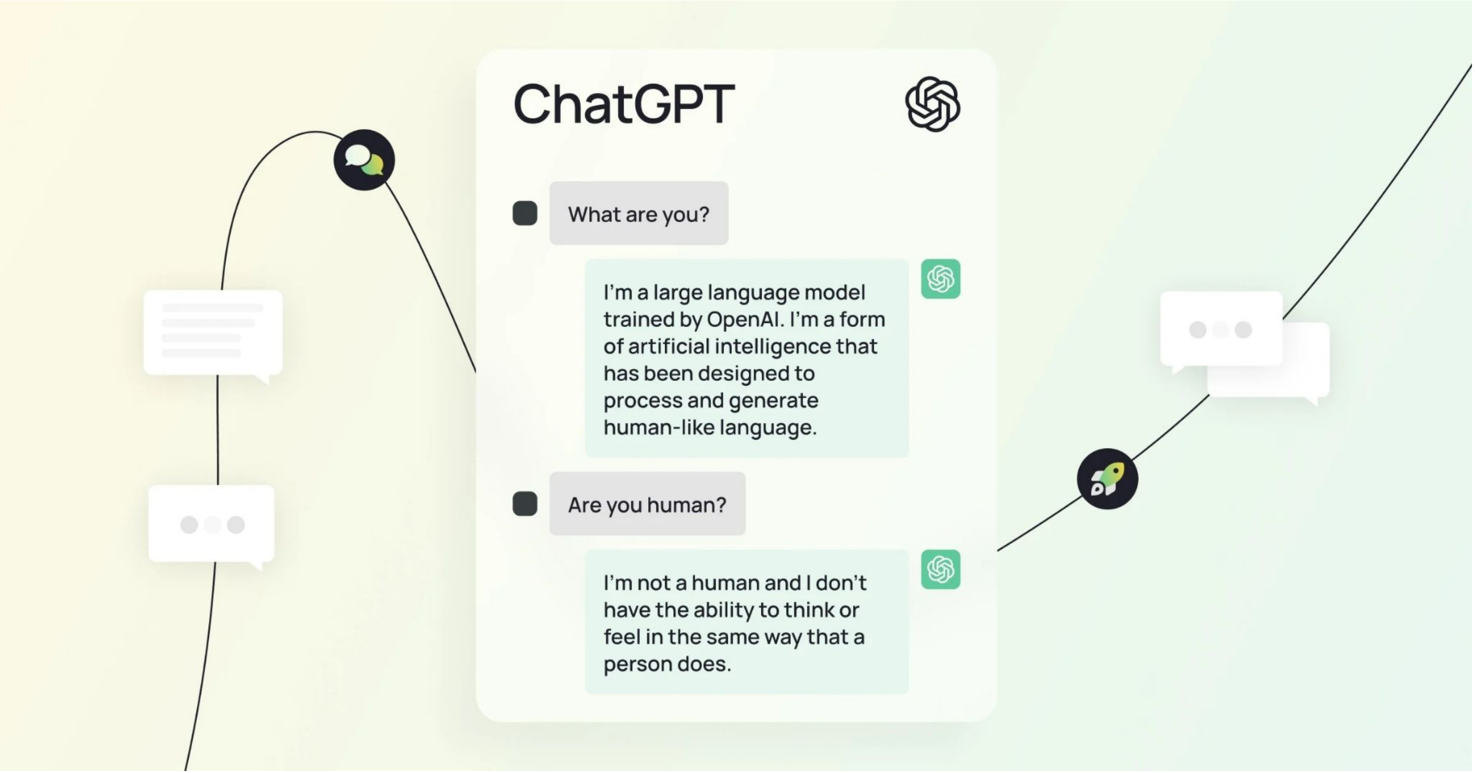
The Future (What's Missing?)



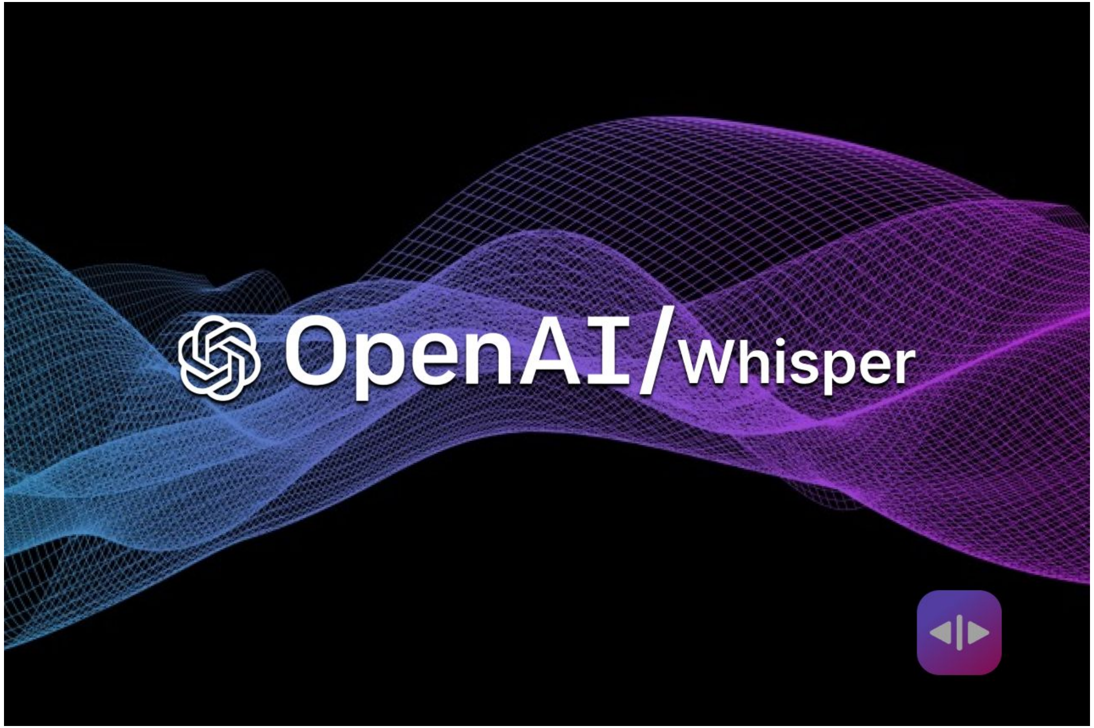
- Missing Ingredients (to AGI/ASI?):
    - Reducing computation complexity
    - Enhanced human controllability
    - Alignment with language models of human brain
    - Adaptive learning and generalization across domains
    - Multi-sensory multimodal embodiment (e.g. intuitive physics and commonsense)
    - Infinite/external memory: like Neural Turing Machines
    - Infinite/constant self-improvement and self-reflection capabilities
    - Complete autonomy and long-horizon decision-making
    - Emotional intelligence and social understanding
    - Ethical reasoning and value alignment

Major Applications of Transformers

Text and Language



Audio: Speech + Music



Vision: Analyzing Images & Videos

Vision Transformer (ViT)

Vision: Generating Images & Video

Images: Stable Diffusion, Dall-E, Midjourney, etc.

Videos: Sora, Pika, etc.

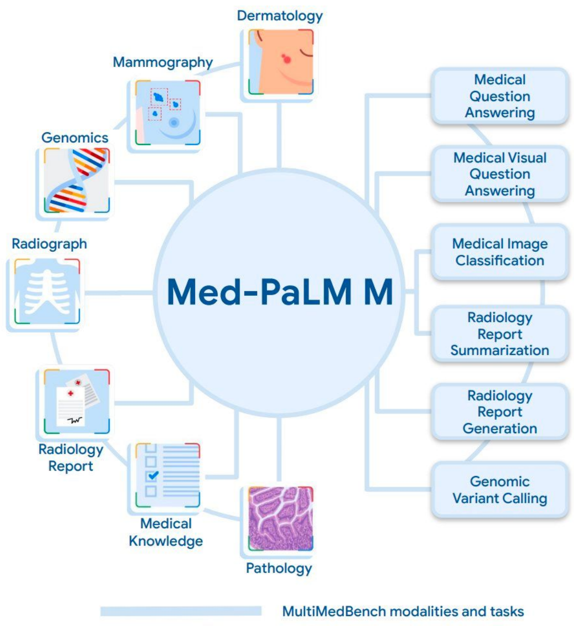
Robotics, Simulations, Physical Tasks

E.g. Voyager, Mobile ALOHA

Playing Games

E.g. AlphaGo, AlphaStar,
AI for MOBAs (e.g. Dota 2 / LoL)

Biology + Healthcare



E.g. Med-PaLM, AlphaFold

Recent Trends and Remaining Weaknesses of LLMs

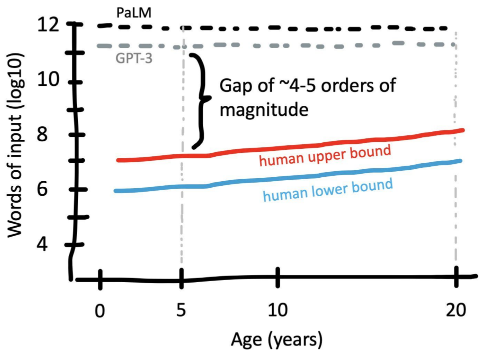
Requiring Large Amounts of Data, Compute, and Cost

- Current LLMs take immense amounts of data, compute, and $ to train
- Requires training over weeks/months over thousands of GPUs
- BabyLM challenge: can we train LLMs using similar amounts of data a baby is exposed to while growing up?

BabyLM: Children vs. LLMs

- Children are different due to several reasons:
  - LMs do statistical learning, which requires more data to learn statistical relations between words and get abstraction/generalization/reasoning
  - Children may learn in smarter, e.g. more explicit compositional/hierarchical manners, learning abstraction/generalization/reasoning more easily

BabyLM: Children vs. LLMs



- ▶ Thoughts/ideas from Michael C. Frank's tweet
- ▶ 4-5 orders of input magnitude diff b/w human and LLM emergence
- ▶ Factor 1: Innate knowledge - relates to priors
- ▶ Factor 2: multimodal grounding
- ▶ Factor 3: active, social learning
- ▶ Factor 4: evaluation differences

Minified LLMs and On-Device LLMs

- Big trend of using LLMs for applications and everyday purposes
- A requirement is ability to run quickly and easily on-devices
- AutoGPT and ChatGPT “plug-ins”
- Right now, work on smaller open-source models (e.g. LLaMA, Mistral)
- In the future: ability to finetune and run models locally, even on your phone!
  - Getting more possible due to more open-source, but still very large and $

Memory Augmentation & Personalization

- Weakness of LLMs is that they are frozen in knowledge at a particular point in time, and don’t augment knowledge “on the fly”
- Hope to be able to remember the information while chatting with a particular user, both within the same conversation and across conversations
  - Would help with context window limits and adapting to the particular user
- Widescale: somehow update the model “on the fly” with info from several users
- Further, they usually do not adapt their talking style and persona to the particular user, which could have applications such as mental health therapy

Memory Augmentation & Personalization

- Potential approaches:
  - Memory bank - not feasible/efficient with larger amounts of data
  - Prefix-tuning approaches (finetune a small part of the model) - too expensive
  - Some prompt-based approach - do not see how this would be possible to change the model itself, but can at least help it “personalize” to the user
  - RAG: retrieval-augmented generation (data store, augment context each time)
    - Relies on high-quality external data store
    - Typically not end-to-end
    - Not within the “brain” of the model but outside:
      - Suitable for knowledge/facts, but not fundamental capabilities and skills

Pretraining Data Synthesis & Selection

- Lots of work these days on synthetic data generation (e.g. using GPT-4) to train other models, e.g. smaller models or peer models
- More work on understanding how to best synthesize and select the pretraining data
- Related to model distillation: knowledge from a large complex model (Teacher) is transferred to a smaller, more efficient model (student)
  - Goal: achieve similar performance with less computational cost
- Example: Microsoft Phi models ("Textbooks Are All You Need!")
  - https://arxiv.org/abs/2306.11644
- Nathan Lambert's Summary on Synthetic Data in his Interconnect Newsletter

Microsoft Phi-2 Model

- Phi-2, a 2.7 billion-parameter model, excels in reasoning and language understanding, challenging models up to 25x larger
- Emphasizes "textbook-quality" training data and synthetic datasets for teaching common sense and general knowledge
  - Training data mix: synthetic datasets to teach the model commonsense reasoning and general knowledge, including science, daily activities, ToM, etc.
  - Further carefully selected web data filtered by educational value + content quality
- Phi-2 designed as a resource for research on interpretability, safety improvements, and fine-tuning across tasks

New Knowledge or “Memorizing”?

- When LLM is prompted and says something, is what it says truly “novel/new”?
- **Innovation vs. Regurgitation:** ongoing debates about whether LLMs can truly invent new ideas or are primarily recombining existing knowledge (since learn patterns from lots of text)
- **Test-time Contamination**: models might regurgitate rather than synthesize information due to overlap between training and evaluation data, leading to misleading benchmark results
- **Cognitive Simulation:** some argue LLMs mimic human thought processes, suggesting a form of "understanding," while others see this as simply “sophisticated pattern matching”
- **Ethical and Practical Implications**: this impacts trustworthiness, copyright issues, and the educational use of LLM outputs
  - E.g. copyright lawsuit by New York Times (NYT) on OpenAI!

Continual Learning

- AKA, infinite and permanent fundamental self-improvement
- Similar to humans: we constantly learn everyday from every interaction
  - Don’t need to “finetune ourselves” once in a while
- Very challenging, could be the key to AGI!
- Currently work on: finetune a small model based on traces from better model or same model after filtering those traces
  - More like re-training and distillation than true “continual learning”
- Work showing that reasonably sampled data with interjected augmented reasoning and further filtering can be used to further finetune or optimize (e.g. using DPO)
  - E.g. UltraChat-200k and Zephyr
  - E.g. LLMs Can Self-Improve paper

Interpretability of LLMs

- Enormous number of parameters trained on tons of data → “huge black-box” that is hard to interpret and understand
- More work on interpretability is required
- Would allow us to better understand models, leading to better ideas of what/how to improve, easier control, and better alignment/safety
- Mechanistic interpretability: understand how individual components + operations in an ML model contribute to its overall decision-making process
  - Goal: unpack the "black box" of models for clearer insight into how they work

Model Editing & Mechanistic Interpretability

- ► Also work on mechanistic interpretability and model editing (e.g. edit specific nodes)
- ► Relevant paper: https://arxiv.org/abs/2202.05262
- ► Development of a causal intervention method to trace decisive neuron activations for model factual predictions

- ► Rank-One Model Editing (ROME) to modify model weights for updating factual associations
- ► Mid-layer feedforward modules play a significant role in storing factual associations
  - ► Manipulation of these can be a feasible approach for model editing

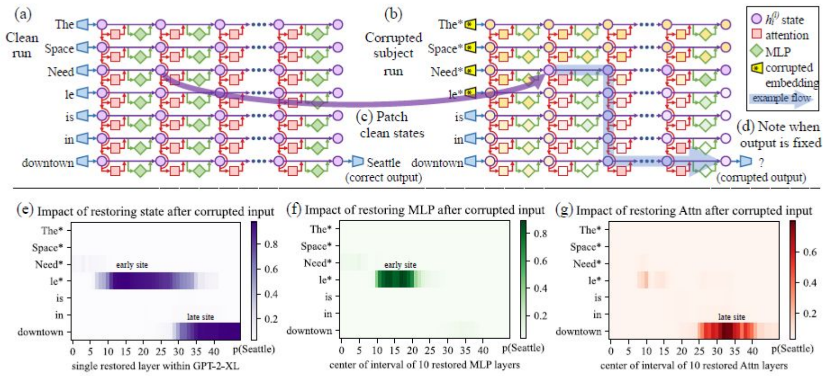
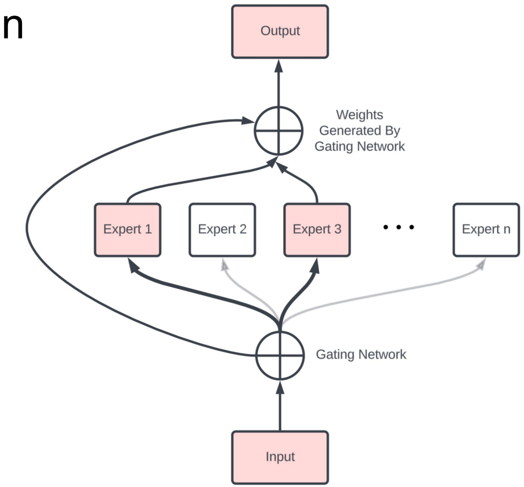


Figure 1: **Causal Traces** compute the causal effect of neuron activations by running the network twice: (a) once normally, and (b) once where we corrupt the subject token and then (c) restore selected internal activations to their clean value. (d) Some sets of activations cause the output to return to the original prediction; the light blue path shows an example of information flow. The causal impact on output probability is mapped for the effect of (e) each hidden state on the prediction, (f) only MLP activations, and (g) only attention activations.

Model Modularity + Mixture of Experts (MoE)

- Mixture of Experts (MoE) very prevalent these days in LLMs:
  - E.g. GPT-4, Gemini, etc.
- Goal: have several models/“experts” work together to solve a problem
  - Each expert may be specialized for a task/purpose
  - Try to use the diff skill-sets together to arrive at a generation
- Research on how to better define and connect these “experts”



Model Modularity + Mixture of Experts (MoE)

- Single model variation (?)
  - Potential to segment/compartmentalize a single NN model into different compartments with their own focus, similar to the human brain?
    - E.g. part of the network for fact-based info, another for spatial reasoning, another for mathematical + logical reasoning, etc.
  - Maybe add more layers on top of the foundation model
    - Particular layers correspond to something (e.g. new domain), and try to tune these new layers specifically

Self-Improvement / Self-Reflection

- Found models can reflect on their own output to iteratively improve/refine them
- Examples of works: ReAct, Reflexion, Self-refine
- Training LMs with Language Feedback: https://arxiv.org/abs/2204.14146

Self-Improvement / Self-Reflection

- Tried multiple layers/levels of self-reflection… showing continual improvement
  - Hypothesize that results will improve to a certain point and then degrade, and depends both on the model scale and the task at hand
  - Some folks believe that AGI is a “constant state of self-reflection”
- Can investigate further improvements to chain-of-thought reasoning and self-reflection

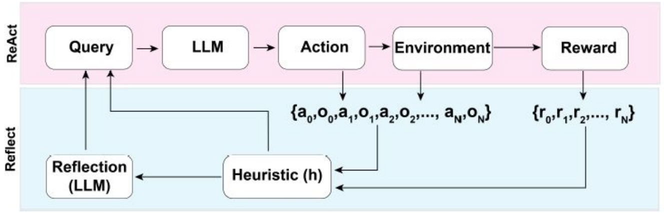
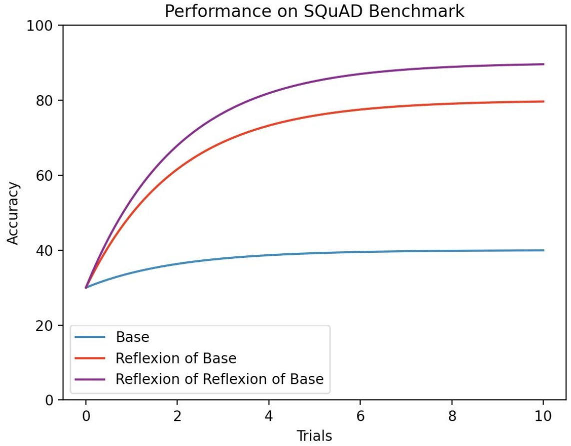


Figure 1: Reflexion can be added to any decision-making approach. We enable ReAct agents to use self-reflection to improve their own performance.



Hallucination Problem

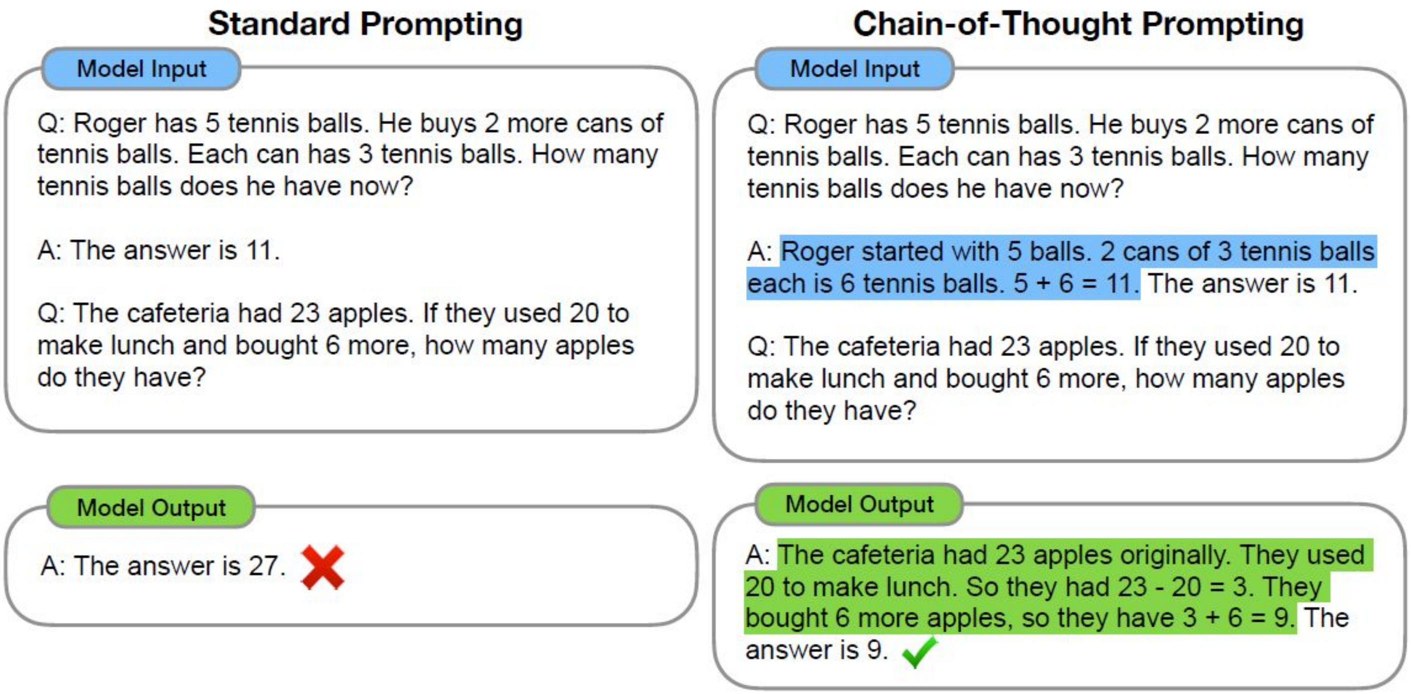
- Model “does not know when it does not know”
- Due to sampling procedure, generates text and sounds confident about it, even when littered with factual errors
- Can enhance through retrieval (e.g. through Google), and also:
  - Internal-based “fact verification”
  - Output verification and regeneration (self-refinement/improvement)
  - Modifying the token sampling to shy away from hallucination
  - Some way to “predict hallucination” before generation and prevent it
- “Confidence” rating would also help check the reliability of the output

Reasoning: Sufficient? Intermediate Guidance Helps…

- Chain-of-thought (CoT) - series of intermediate reasoning steps
- Shown to improve LLM performance on complex reasoning tasks
- Inspired by human thought process: decompose multi-step problems
- Also provides an interpretable window into behavior of the model (how it arrived at an answer, where it goes wrong in its reasoning path)
- CoT exploits the fact that deep down in the model's weights, it knows more about the problem than just prompting it to get a response

Ref: https://arxiv.org/abs/2201.11903

Chain-of-Thought Reasoning



Ref: https://arxiv.org/abs/2201.11903

Improving Chain-of-Thought Reasoning

- CoT results in performance gains for larger LMs, but still remain a non-negligible fraction of errors
- CoT error breakdown:
  - 8% from just a calculator error
  - 16% from symbol mapping error
  - 22% from “one missing step” error
  - Remaining errors due to semantic understanding issues and incoherent CoT
- We can investigate methods to address above errors and improve CoT in general

Ref: https://arxiv.org/abs/2201.11903

Chain-of-Thought Reasoning for Smaller Models

- Currently, CoT works effectively for models of approx. 100B params or more
- Initial paper found “one-step missing” and “semantic understanding” CoT errors to be the most common among smaller models
- 3 potential reasons:
  - Fail at even relatively easy symbol mapping tasks
  - Seem to have inherently weaker arithmetic abilities
  - Often had logical loopholes and did not arrive at a final answer
- Improve CoT for smaller models → significant value to the research community

Ref: https://arxiv.org/abs/2201.11903

Generalizing Chain-of-Thought Reasoning

- Find CoT to have a more rigid definition and format
- Further, its advantages are for particular domains and types of questions
  - Task is challenging and requires multi-step reasoning
  - Scaling curve of the problem/task is relatively flat
- Humans think through different types of problems in multiple ways
- Our “scratchpad” is more flexible and open to different reasoning structures
- Can maybe generalize CoT to be more flexible somehow

Tree of Thoughts

- ToT: “consider multiple different reasoning paths and self-evaluating choices to decide the next course of action, as well as looking ahead or backtracking when necessary to make global choices”

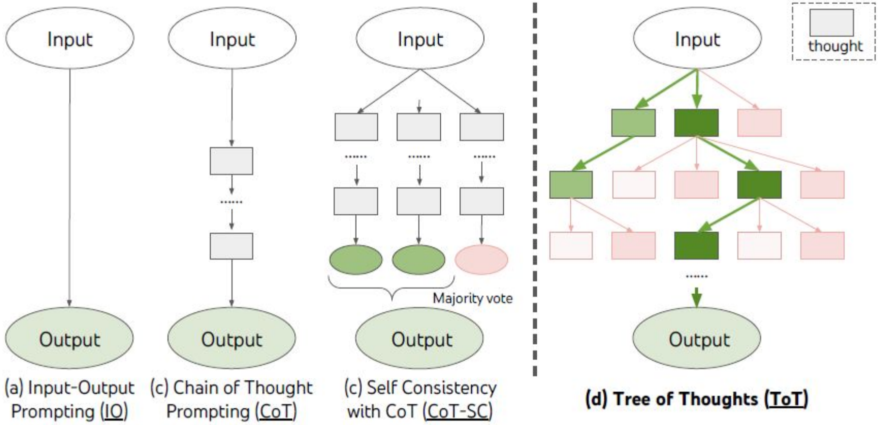


Figure 1: Schematic illustrating various approaches to problem solving with LLMs. Each rectangle box represents a *thought*, which is a coherent language sequence that serves as an intermediate step toward problem solving. See concrete examples of how thoughts are generated, evaluated, and searched in Figures 2,4,6.

Ref: https://arxiv.org/abs/2305.10601

Socratic Questioning

- “Divide-and-conquer fashion algorithm that simulates the self-questioning and recursive thinking process.”
- “Self-questioning module using a large-scale LM to propose subproblems related to the original problem as intermediate steps and recursively backtracks and answers the sub-problems to reach the original problem.”

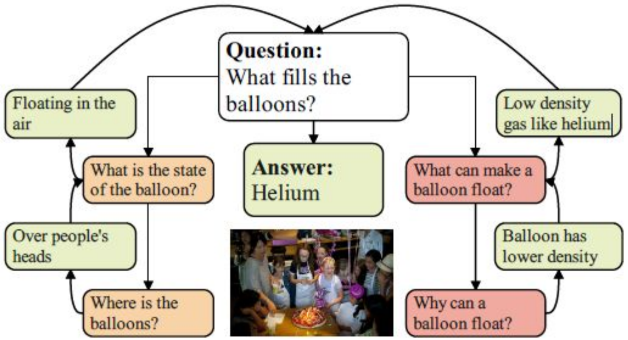


Figure 1: Example of a complex visual question solved in the human thinking process, involving raising visual and commonsense questions.

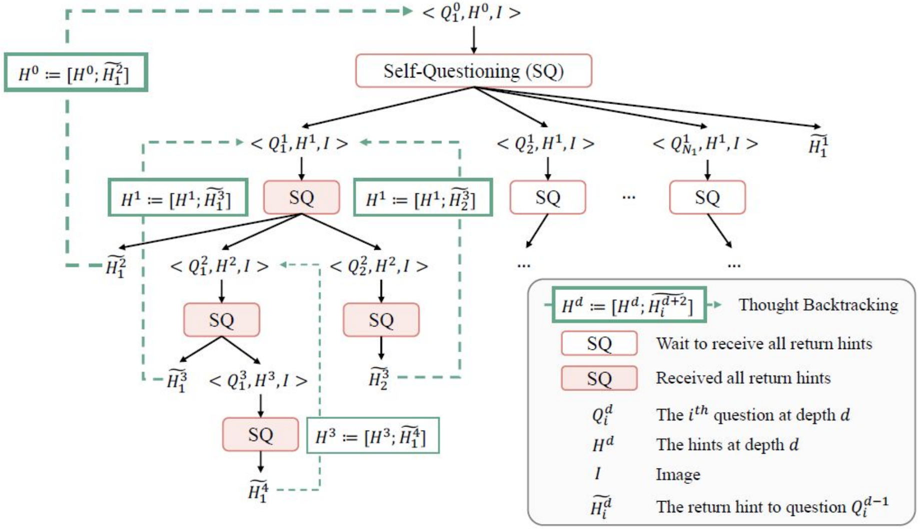


Figure 2: Overview of our SOCRATIC QUESTIONING framework.

Ref: https://arxiv.org/abs/2305.14999

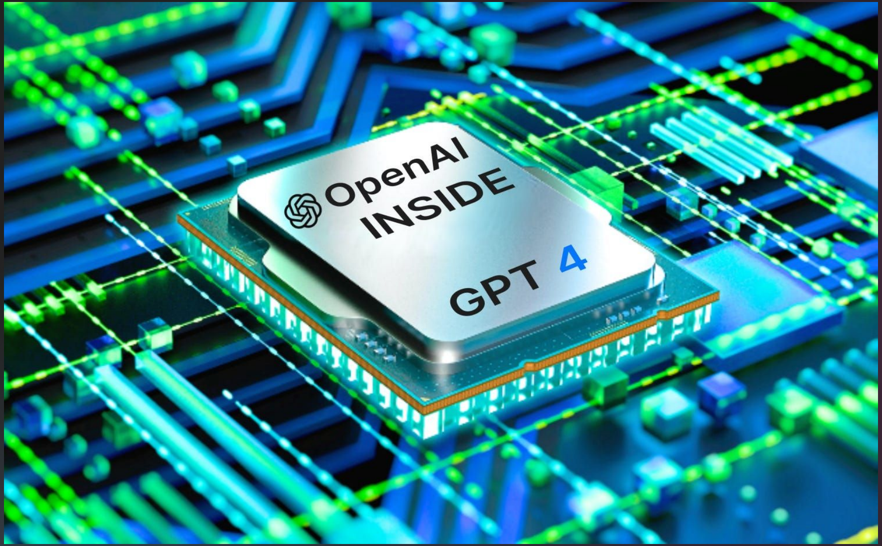
From Language Models to AI Agents

Introduction

- Actions and Emergent Agent Architectures
- Building Human-like AI Agents
- Computer Interactions using AI
- Long-Term Memory and Personalization
- Agent to Agent communication
- Future Directions for Autonomous AI Agents

Building AI Agents

1. Why?
2. How?
3. Ingredients?
4. What can they do?



Key thesis: Humans will communicate with AI using natural language and AI will operate machines allowing for more intuitive and efficient operations

Software 3.0

AI Agents

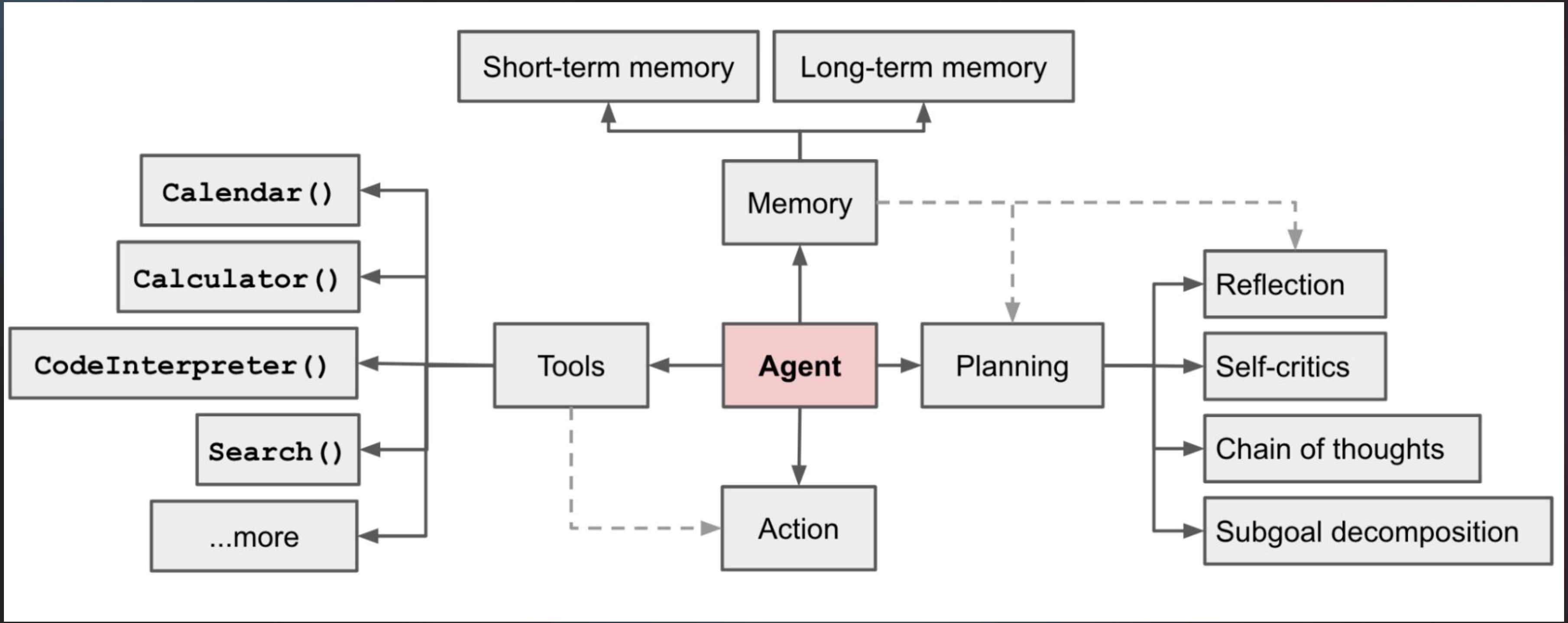
1. **Why?**
   A single call to a large Foundation AI model is not enough. A lot more can be unlocked by building ***AI systems***

2. **How?**
   Using model chaining, reflection & other mechanisms

3. **Ingredients?**
   Memory, context length, personalization, actions, internet access…

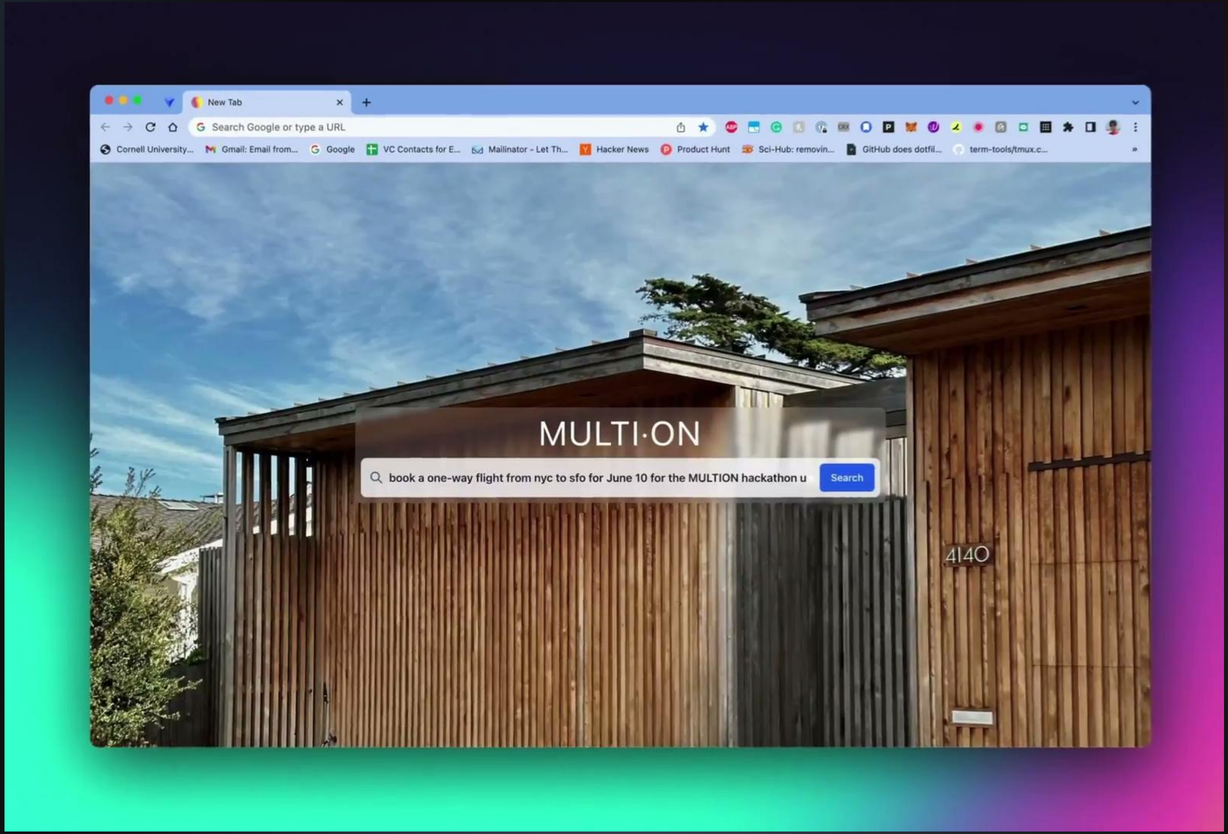
4. **What can they do?**

AI Agents

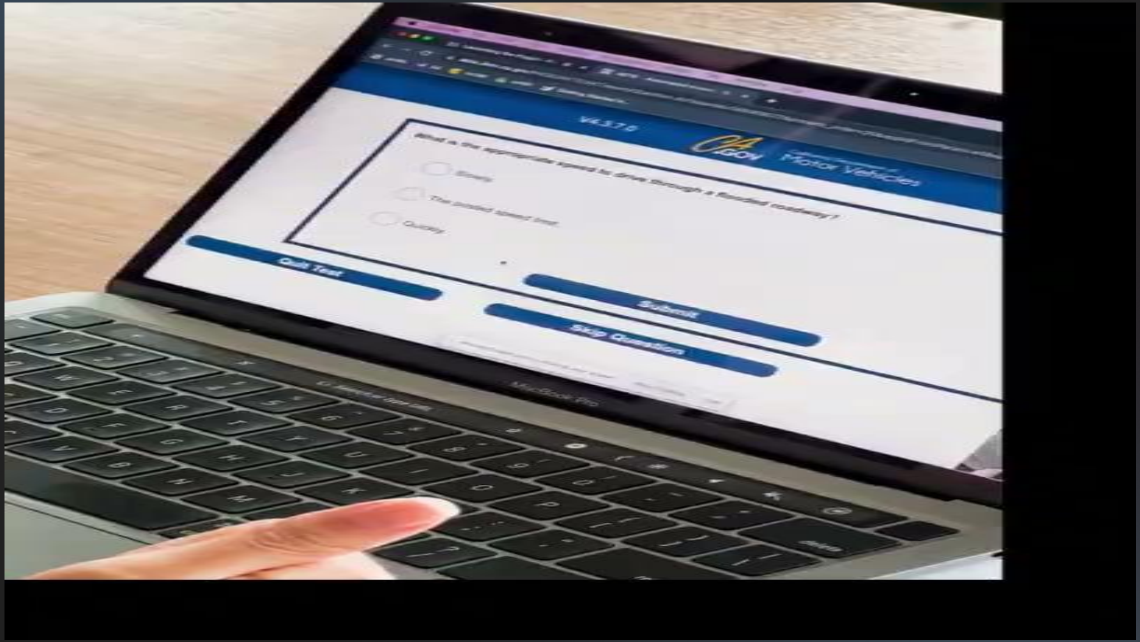


Short-term memory
Long-term memory
Memory
Calendar()
Calculator()
CodeInterpreter()
Search()
...more
Tools
Agent
Planning
Action
Reflection
Self-critics
Chain of thoughts
Subgoal decomposition

The first flight to be fully booked by an AI



AI Agent Passing the Online CA Driving Test



Fully Autonomously passing the official DMV online driving test this week & setting the record as the first AI to obtain a driving permit in CA!

Why **human-like** AI Agents?

1. **Can do what you can do**: Able to use existing interfaces designed for humans & operate outside programmatic boundaries

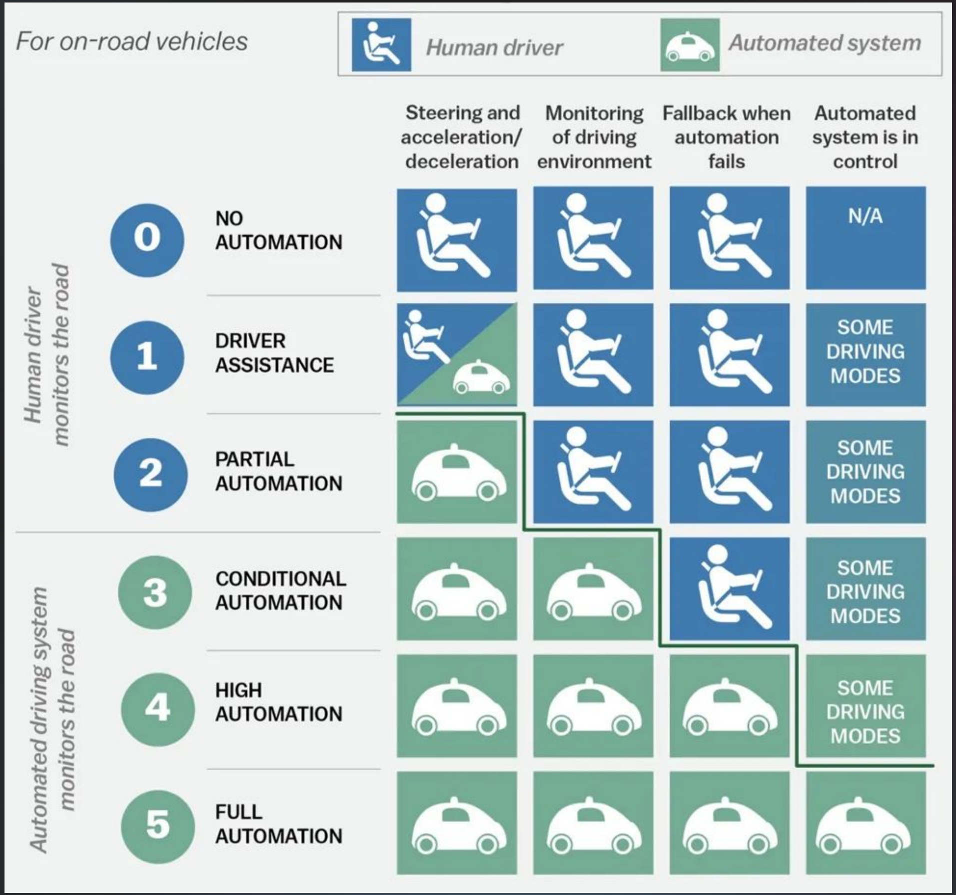
2. **Digital extension of you**: Can act as an extension of the user and act on their behalf

3. **Less-restrictive boundaries**: Can handle logins, payments, etc. and interact with services without any API restrictions

4. **Simple action space**: Need only click & type action primitives

5. **Self-learning**: Can learn from the user and self-improve with more interactions

5 levels of Autonomy



Computer Interactions

Agent Computer Interaction

Two routes

GORILLA

OpenAI ChatGPT Plugins

API (programmatic)

easy to build context
safer & controllable
high variability

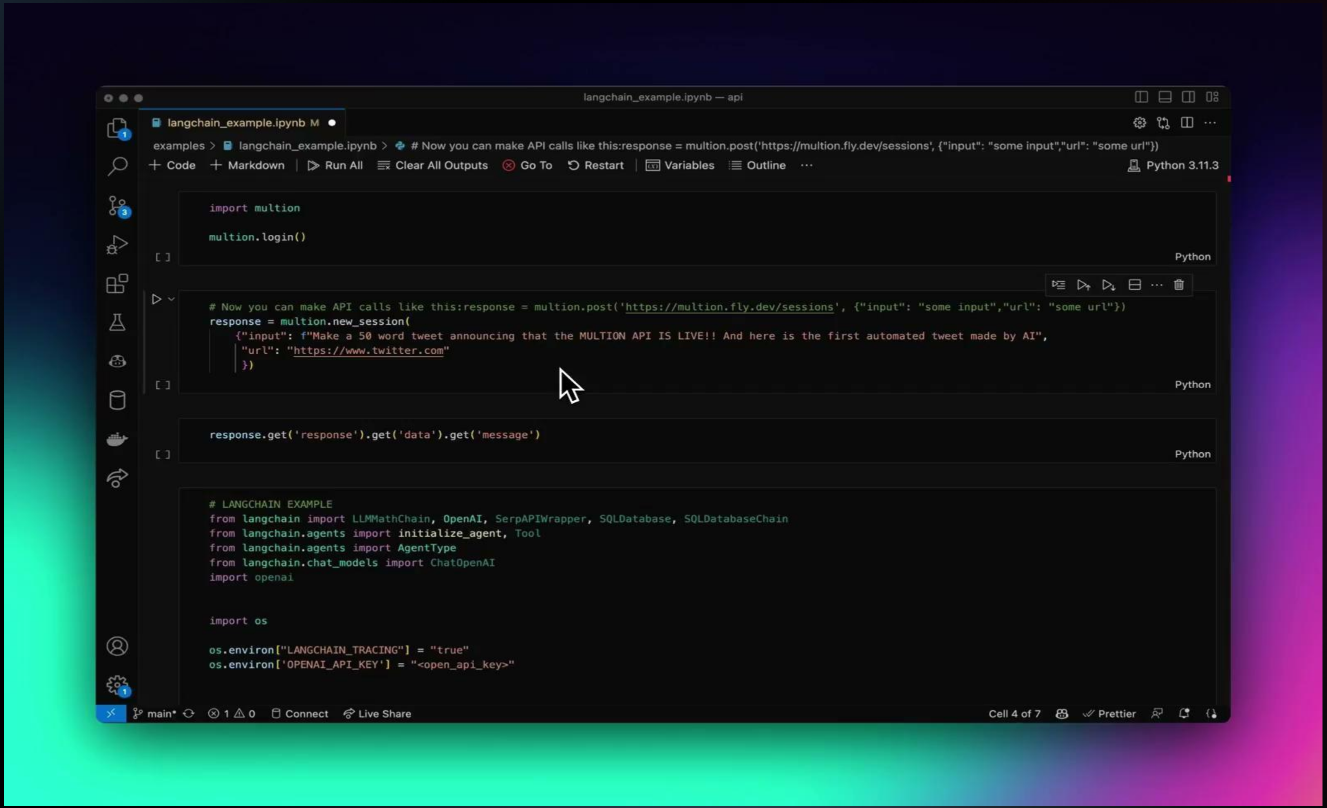
Direct interaction (browser or desktop control)

easy to take actions
free-form interactions
need to provide guarantees

MULTI·ON

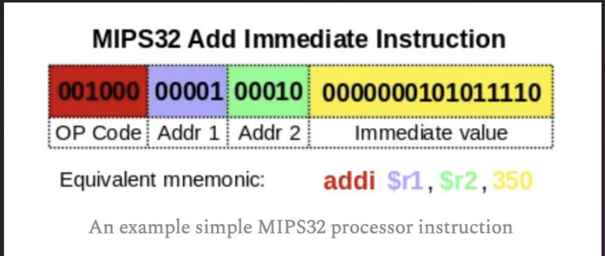
ADEPT

Agent Action API: An universal API for computer interaction



Memory & Personalization

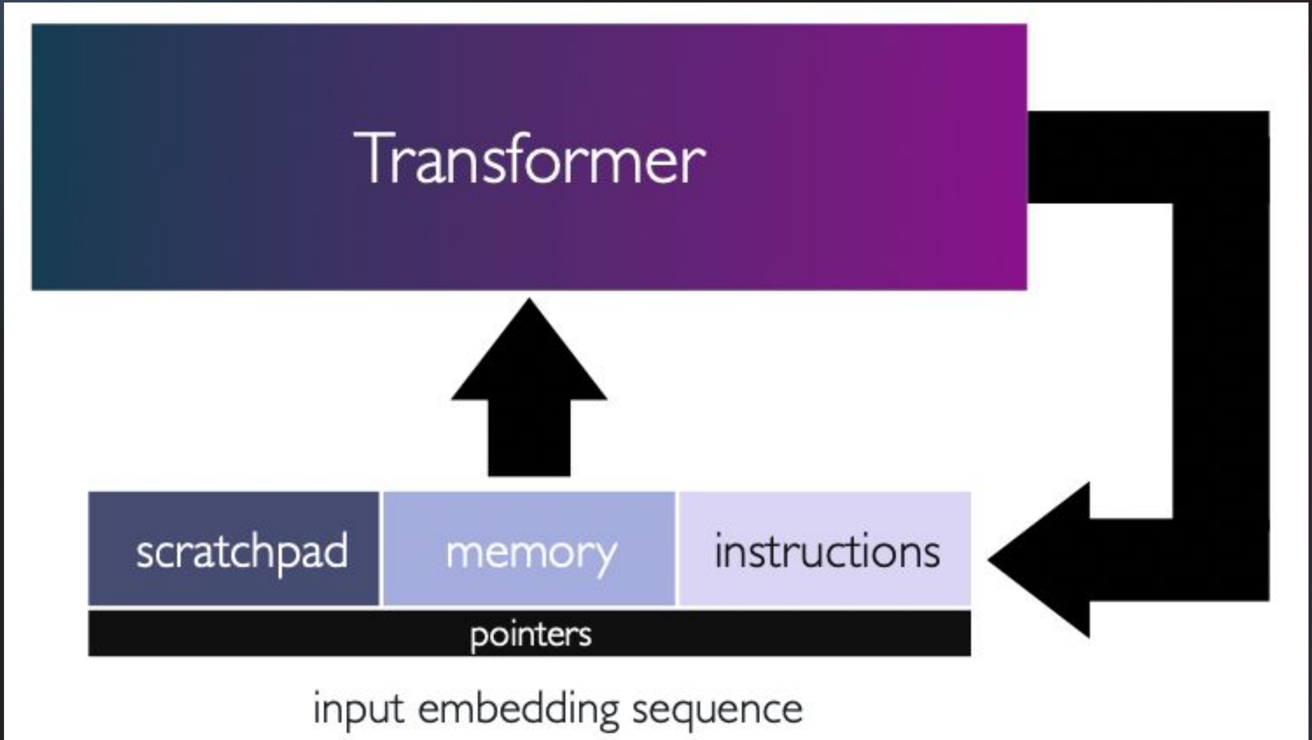
AI Models as Neural Compute Unit



Input Tokens
(max token size)

Output Tokens
(max token size)

AI Models as Neural Compute Unit



Looped Transformers

Long-term Memory

- **Works similar to disk (long-lived & persistent)**
- **Mechanisms**
  - Embeddings
  - Retrieval models
- **Open Questions:**
  - Hierarchy
  - Temporal Coherence
  - Structure
  - Online adaptation

Personalization

- **User-Agent Alignment Problem:** Enable agent to take actions that are aligned with the user preferences

- Everyone has different prefs & likes/dislikes:
  - **Explicit**: allergies, favourite dishes, flight seat prefs, …
  - **Implicit**: choice between brands, out of 10 items in a listing which user likes better

Challenges

- **Collecting user data & preferences:**
  - actively asking for preferences
  - passive learning from interactions

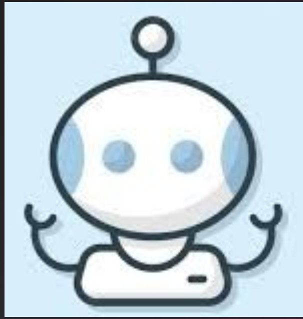
- **Learning from user preferences:** supervised fine-tuning vs human-feedback

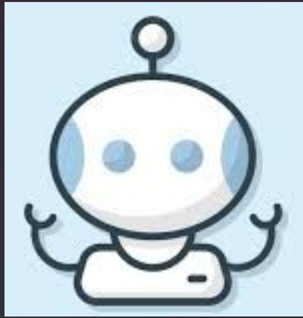
- **On-fly adaptation**

- **Privacy**

Agent-to-Agent Communication

Multi Agent Autonomous AI systems





Why Multi-agent Systems

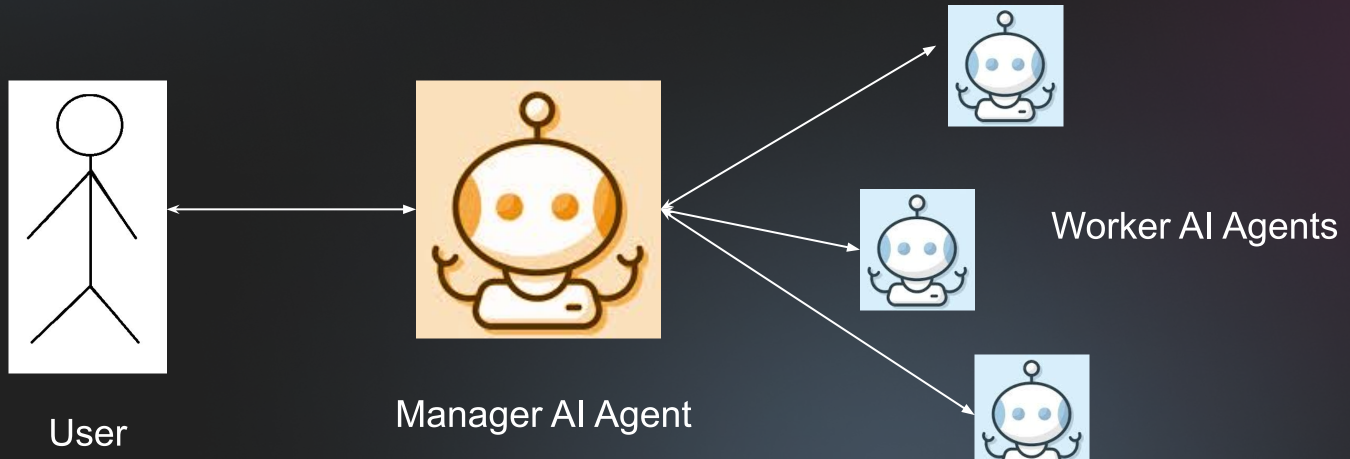
1. **Parallelization unlock**: Breaking a task into smaller chunks and dividing between agents to improve efficiency & speeds

2. **Task Specialization**: An AI agent might seat between the user and each service: e.g. a spreadsheet AI agent, a slack AI agent, a web-browser AI agent, …

3. **Challenges**:
   a. **Agent to Agent Communication**: one AI might want to exchange or request info from another AI agent finish a task

Agent to Agent Communication

- Exchanging info between fleets of agents
- Hierarchies
- Syncing primitives



Agent to Agent Communication

- **Robust communication protocols & Syncing primitives:** Natural language is ambiguous, need mechanisms to reduce miscommunication!

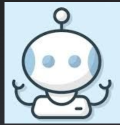
Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**

Manager state

Task X:
(status: not done)

Manager AI Agent



Worker AI Agents

Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**

Do Task X:
- plan
- context

Worker State

Task X:
(status: not done)

Worker AI Agents

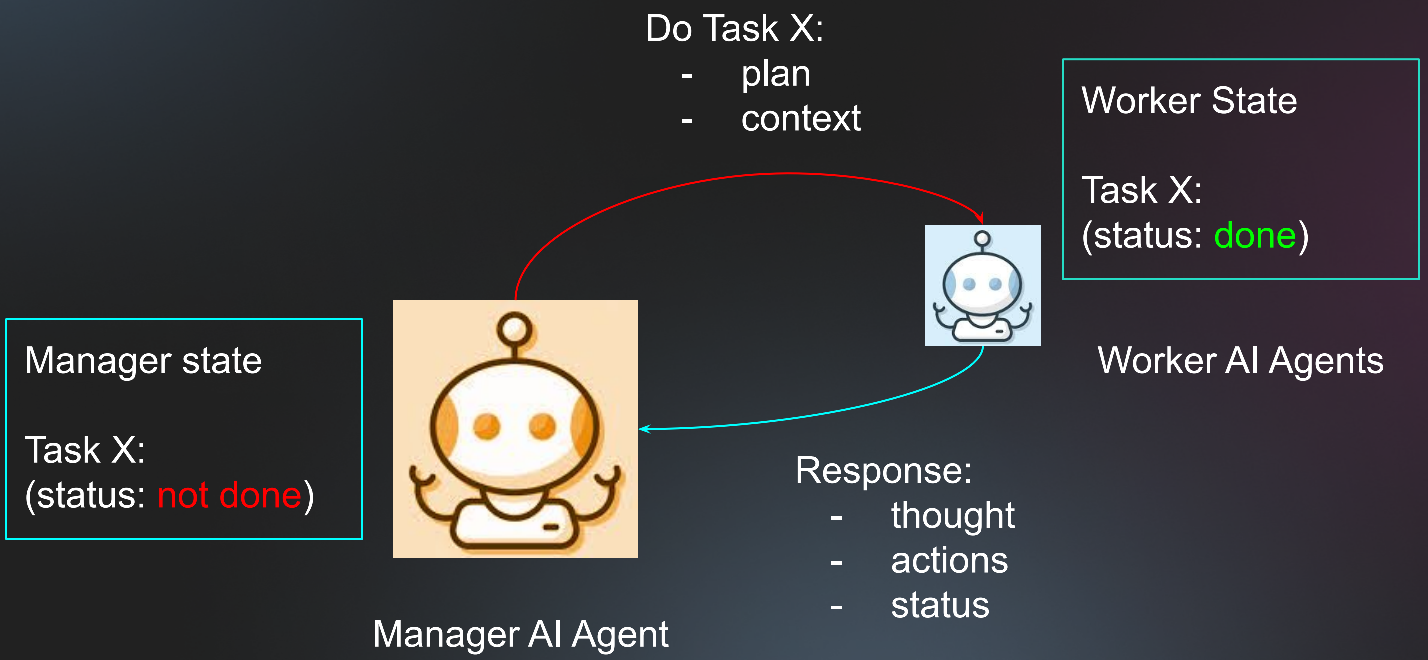
Manager state

Task X:
(status: not done)

Manager AI Agent

Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**



Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**

Verify if task was correctly done & follows all specifications

Verify Task X:
- required spec

Worker State

Task X:
(status: done)

Manager state

Task X:
(status: verify done)

Manager AI Agent

Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**

Scenario 1:
Task was correctly done &
follows all specifications

Verify Task X:
- required spec

Worker State

Task X:
(status: done)

Manager state

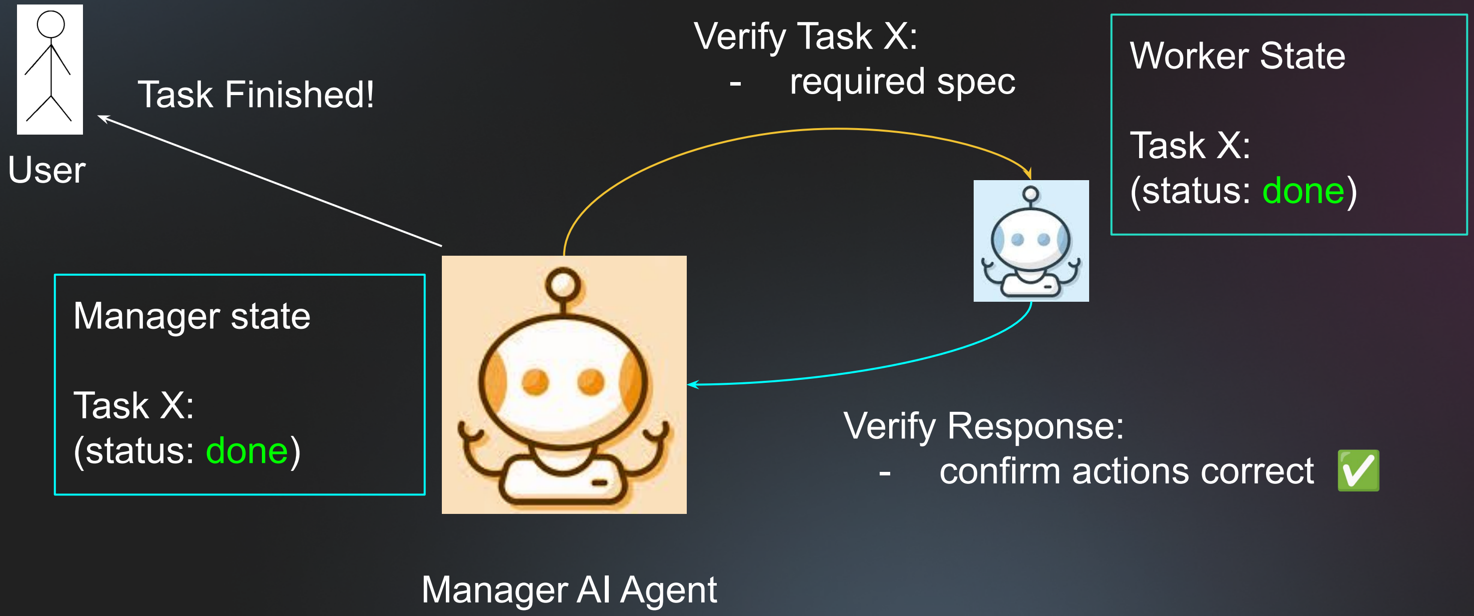
Task X:
(status: verify done)

Verify Response:
- confirm actions correct ✅

Manager AI Agent

Agent to Agent Communication

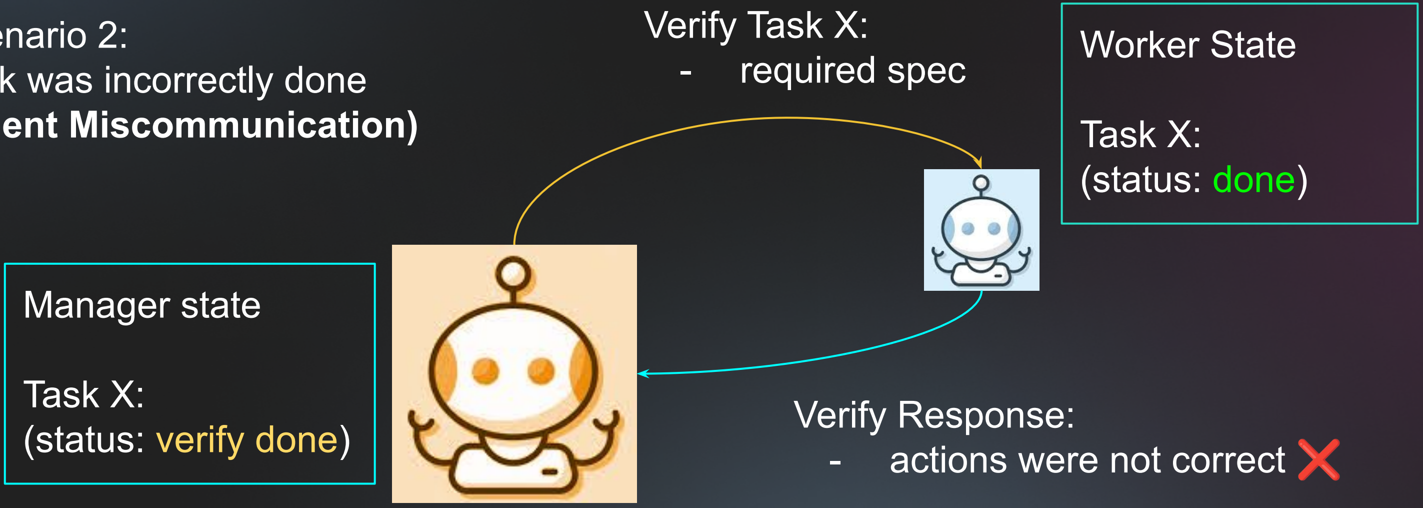
- **Robust communication protocols & Syncing primitives**



Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**

Scenario 2:
Task was incorrectly done
(Agent Miscommunication)



Manager AI Agent

Agent to Agent Communication

- **Robust communication protocols & Syncing primitives**

Re-do Task X:
- plan
- context
- feedback/corrections

Scenario 2:
Task was incorrectly done
(Agent Miscommunication)

Worker State

Task X:
(status: not done)

Manager state

Task X:
(status: not done)

Manager AI Agent

Future Directions

Key Issues with Autonomous Agents

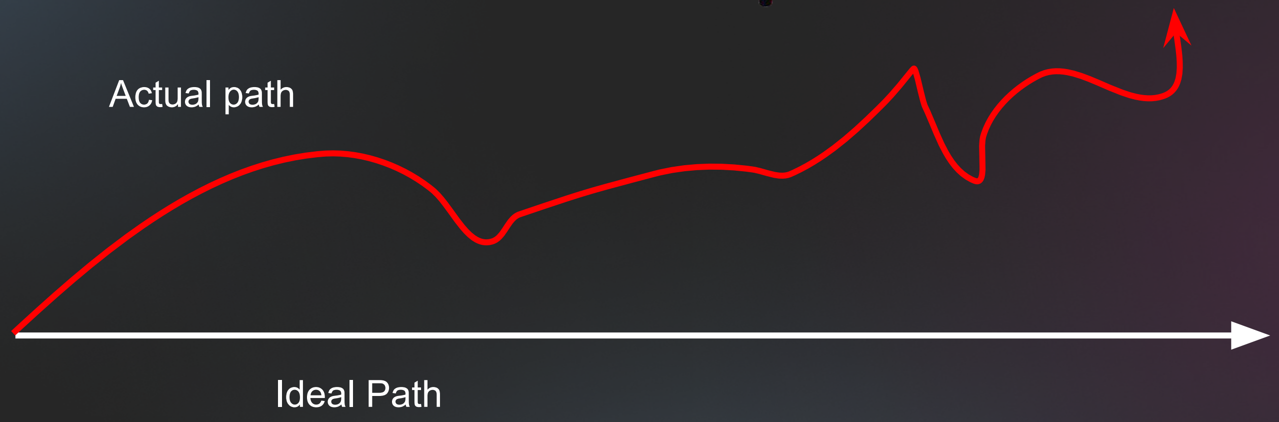
1. **Reliability**

2. **Looping & Plan Divergence**

3. **Testing & Benchmarking**

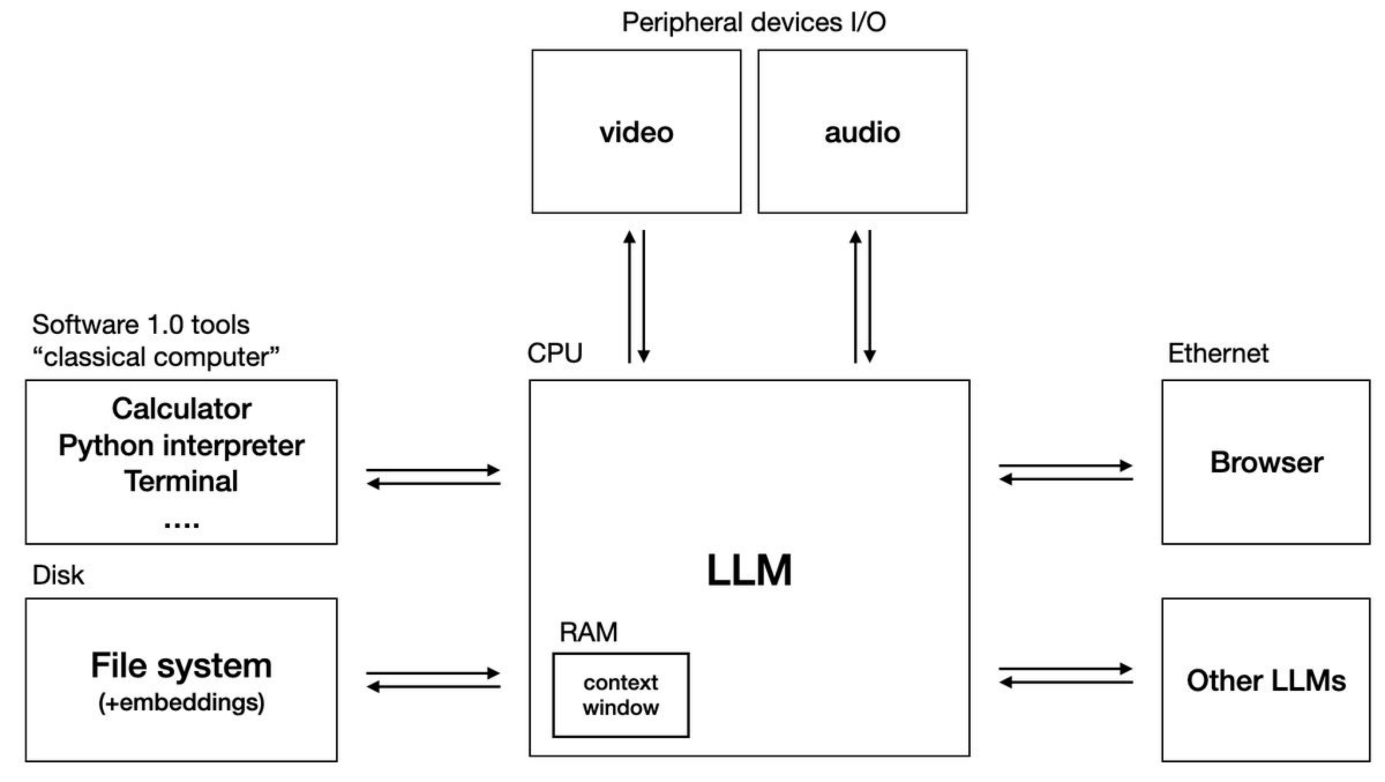
4. **Real world-deployment & Observability**
   a. How do we trust a fully autonomous AI system
   b. How do we build in human overrides

Plan Divergence

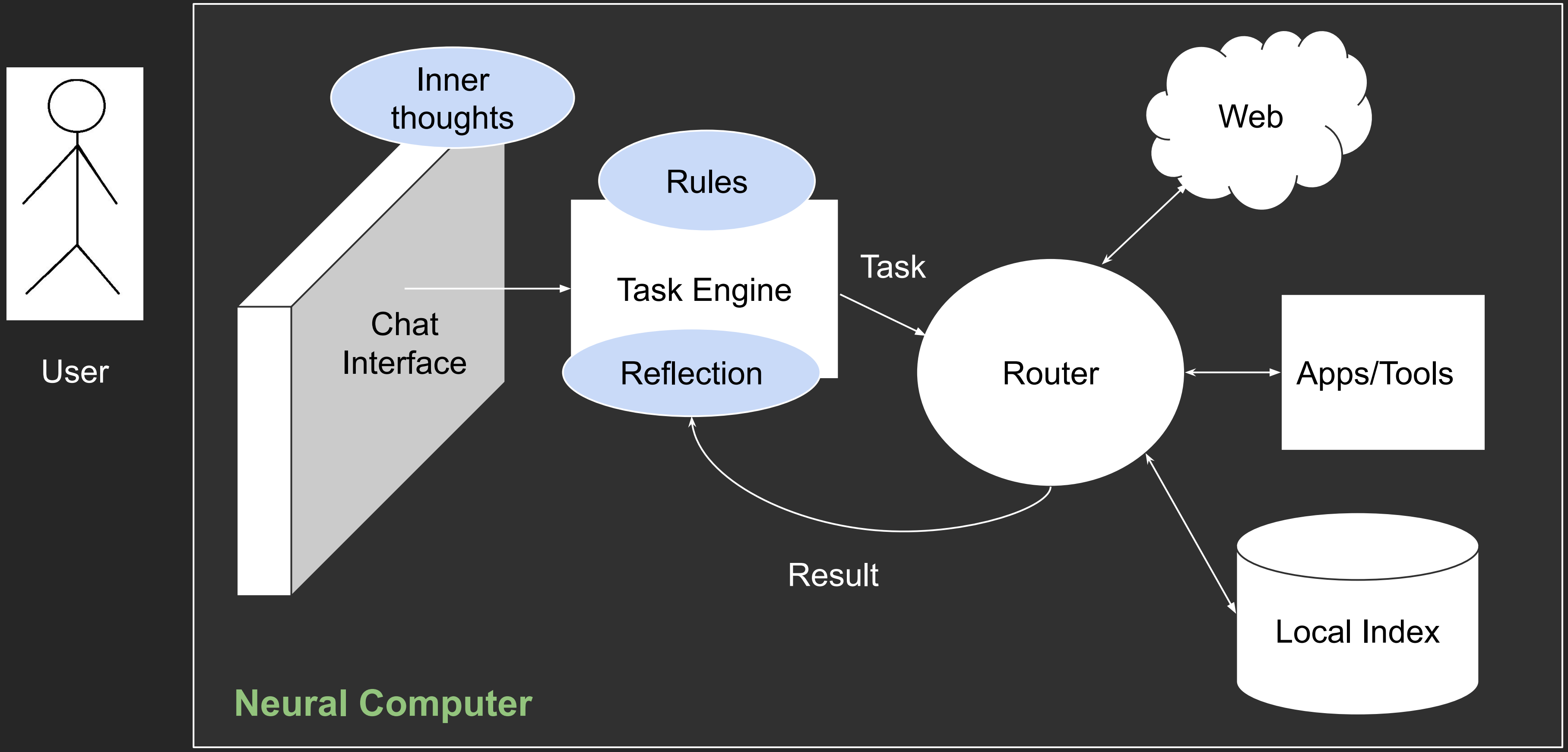


AI Agents like AutoGPT don’t know how to correct on making a mistake!

Karpathy - LLM OS



Building Generalized AI Systems



Future needs for AI agents

- Error correction mechanisms & better agent frameworks
- Security & user permission models
- Sandboxing & deployment in risky settings


That's all Folks!