Recent Models for Language

AI602: Recent Advances in Deep Learning
Lecture 2

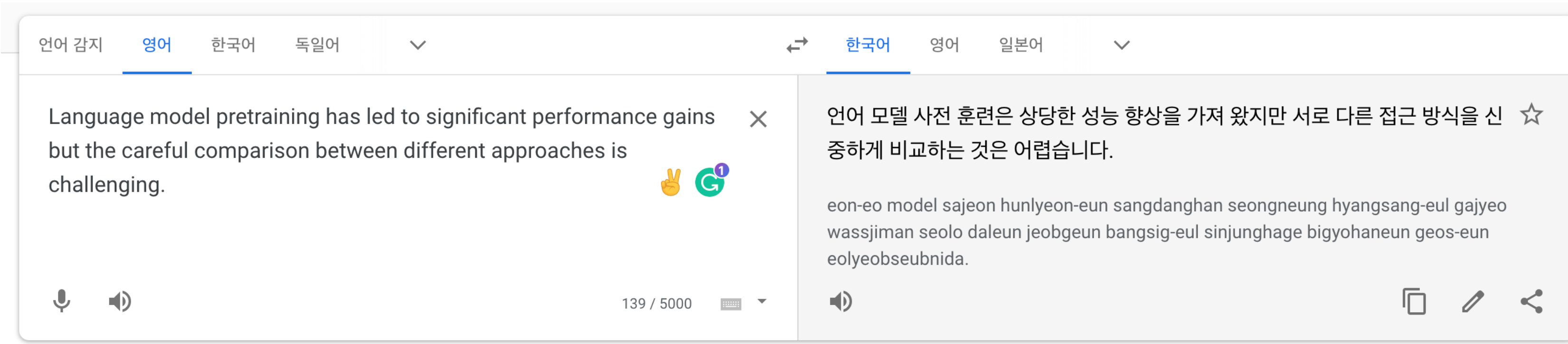
KAIST AI

Motivation: Temporal Data in Real World

- Many real-world data has a **temporal structure** intrinsically
  - Natural language

"Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was __." ⟶ *terrible*

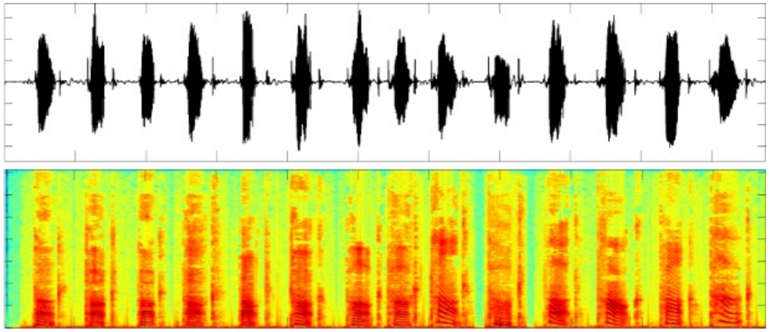
Language modeling



Translation

Motivation: Temporal Data in Real World

- Many real-world data has a **temporal structure** intrinsically
  - Natural language
  - Speech



Motivation: Temporal Data in Real World

- Many real-world data has a **temporal structure** intrinsically
  - Natural language
  - Speech
  - Video

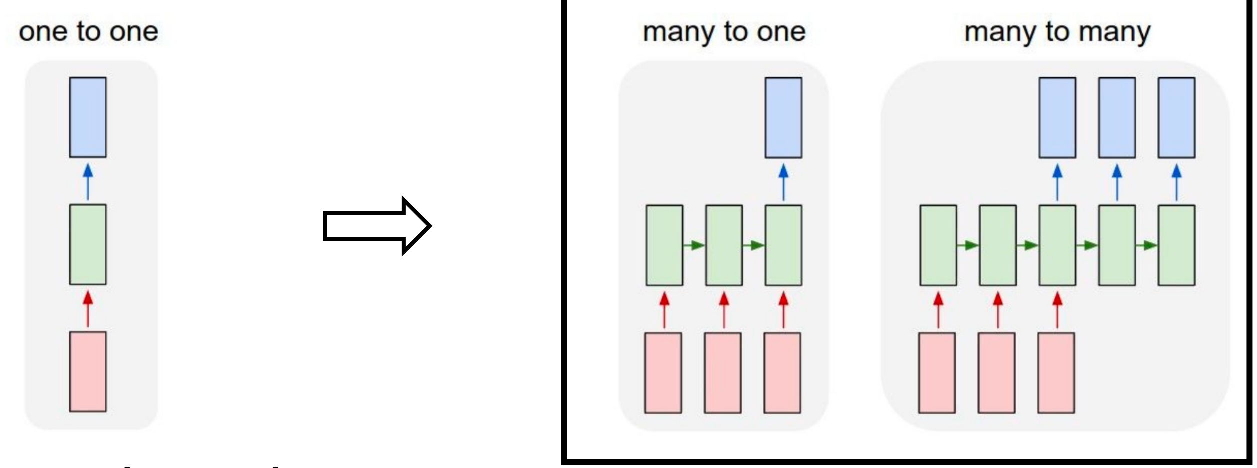
Motivation: Temporal Data in Real World

- Many real-world data has a **temporal structure** intrinsically
  - Natural language
  - Speech
  - Video
  - Stock prices, and etc…

Motivation: Temporal Data in Real World

- Many real-world data has a **temporal structure** intrinsically
  - "**Natural language**"
  - Speech
  - Video
  - Stock prices, and etc…

- In order to solve much complicated real-world problems,
  we need a **better architecture to capture temporal dependency** in the data
  - Specifically, we will focus on the recent models for natural language in this lecture



Vanilla neural network

Overview

Part 1. Basics

- RNN to LSTM
- Sequence-to-sequence Model
- Attention-based NLP Model

Part 2. Advanced Topics

- Transformer (self-attention)
- Pre-training of Transformers and Language Models
- Large Language Models: GPT-3 and emerging properties

Part 3. Recent Advances in Large Language Models

- Large language models beyond GPT-3
- Better training schemes for large language model
- Applications: ChatBot (e.g., ChatGPT)

Overview

Part 1. Basics

- RNN to LSTM
- Sequence-to-sequence Model
- Attention-based NLP Model

Part 2. Advanced Topics

- Transformer (self-attention)
- Pre-training of Transformers and Language Models
- Large Language Models: GPT-3 and emerging properties

Part 3. Recent Advances in Large Language Models

- Large language models beyond GPT-3
- Better training schemes for large language model
- Applications: ChatBot (e.g., ChatGPT)

- Process a sequence of vectors by applying **recurrence formula** at every time step :

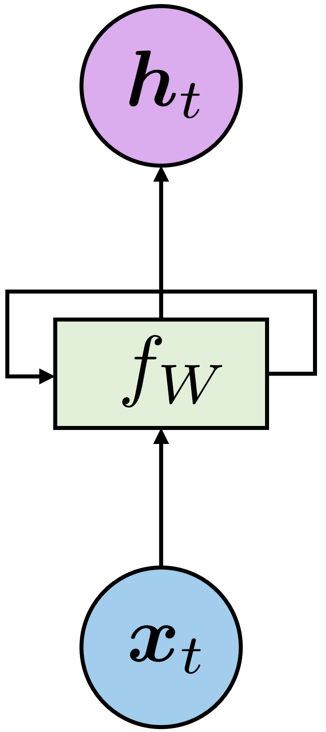
$$\boxed{\boldsymbol{h}_t} = \boxed{f_W}(\boxed{\boldsymbol{h}_{t-1}}, \boxed{\boldsymbol{x}_t})$$

New state

Old state

Input vector at time step t

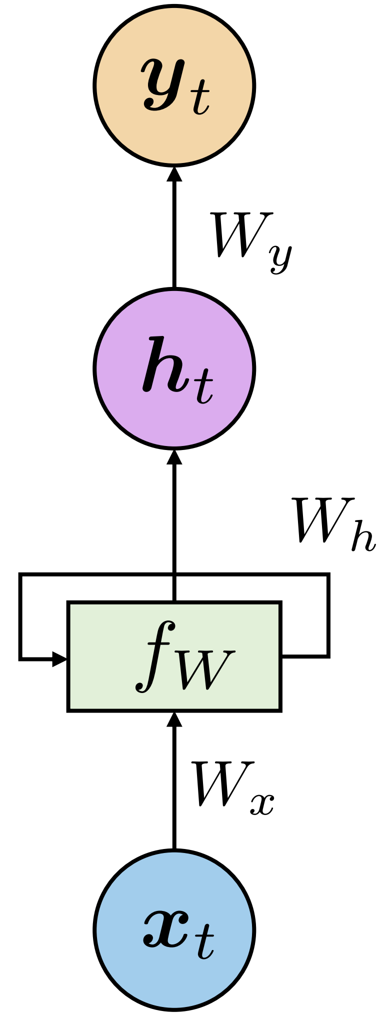
Function parameterized by learnable W



- Vanilla RNN (or sometimes called **Elman RNN**)
    - The state consists of a single "hidden" vector $\mathbf{h}_t$

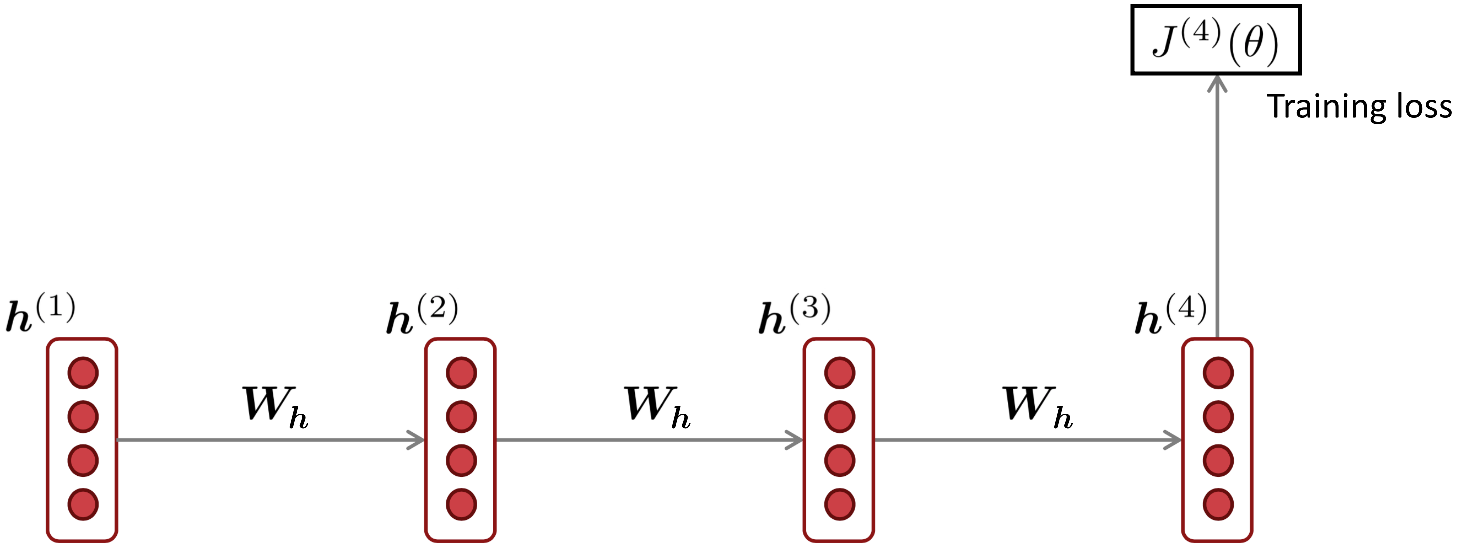
$$\boldsymbol{h}_t = f_W(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$$

$$\boxed{\begin{aligned} \boldsymbol{h}_t &= \tanh(W_h \boldsymbol{h}_{t-1} + W_x \boldsymbol{x}_t) \\ \boldsymbol{y}_t &= W_y \boldsymbol{h}_t \end{aligned}}$$



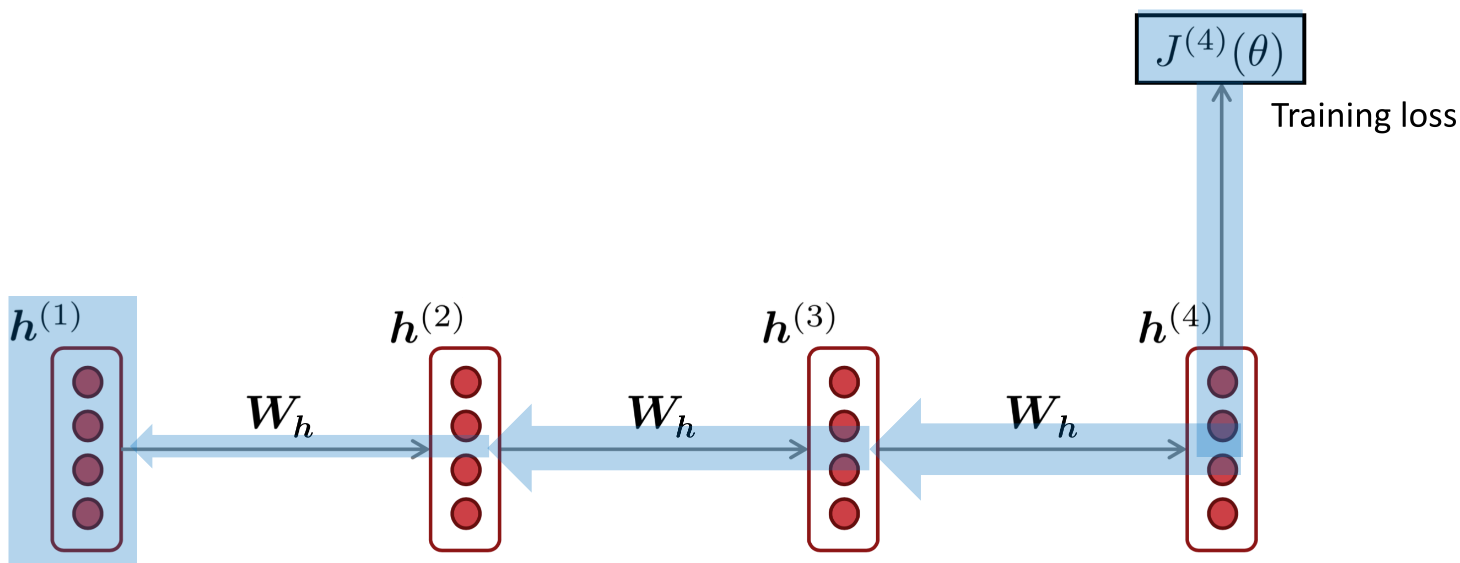
Why Do We Need to Develop RNN Architectures?

- E.g., RNN with a sequence of length 4



Why Do We Need to Develop RNN Architectures?

- E.g., RNN with a sequence of length 4
  - Consider a gradient from the first state $\boldsymbol{h}^{(1)}$

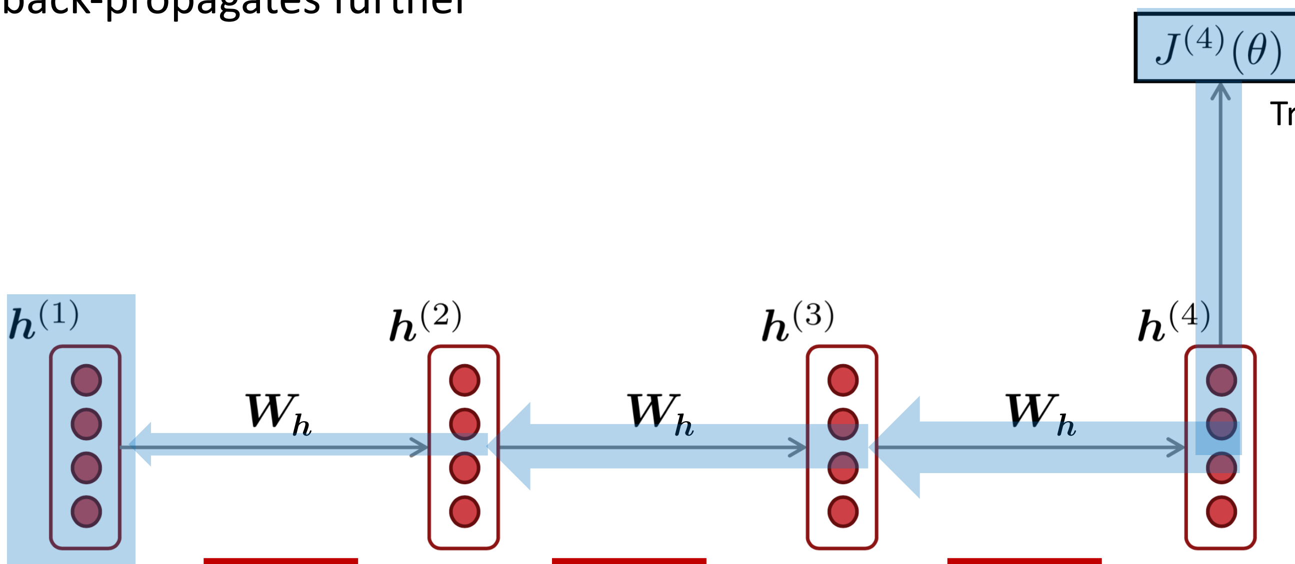


$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}} \times \frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}} \times \frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

Chain rule!

- E.g., RNN with a sequence of length 4
  - Consider a gradient from the first state $\boldsymbol{h}^{(1)}$

- What happens if $\frac{\partial \boldsymbol{h}^{(i+1)}}{\partial \boldsymbol{h}^{(i)}}$ are **too small**? ⇨ **Vanishing gradient problem**
  - When these are small, the gradient signal gets smaller and smaller as it back-propagates further



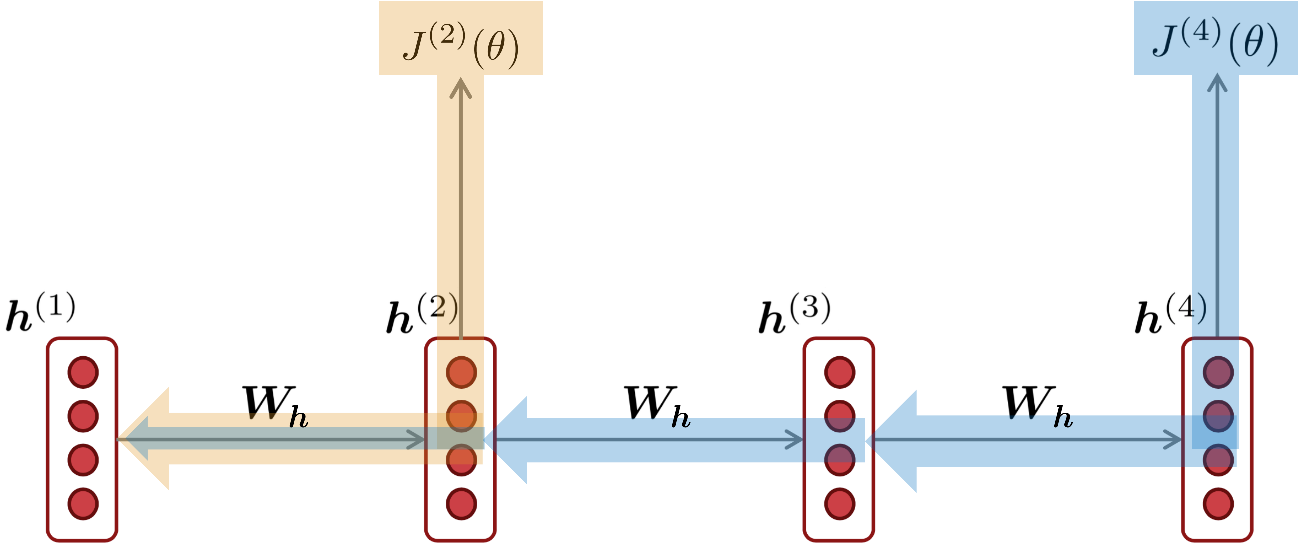
$$\frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(1)}} = \boxed{\frac{\partial \boldsymbol{h}^{(2)}}{\partial \boldsymbol{h}^{(1)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(3)}}{\partial \boldsymbol{h}^{(2)}}} \times \boxed{\frac{\partial \boldsymbol{h}^{(4)}}{\partial \boldsymbol{h}^{(3)}}} \times \frac{\partial J^{(4)}}{\partial \boldsymbol{h}^{(4)}}$$

Chain rule!

Why Do We Need to Develop RNN Architectures?

- E.g., RNN with a sequence of length 4
  - Consider a gradient from the first state $\boldsymbol{h}^{(1)}$

- What happens if $\frac{\partial \boldsymbol{h}^{(i+1)}}{\partial \boldsymbol{h}^{(i)}}$ are **too small**? ⇨ **Vanishing gradient problem**
  - When these are small, the gradient signal gets smaller and smaller as it back-propagates further
  - So, model weight are updated only with respect to near effects, **not** long-term effects.



Why Do We Need to Develop RNN Architectures?

- E.g., RNN with a sequence of length 4
  - Consider a gradient from the first state $\boldsymbol{h}^{(1)}$

- What happens if $\frac{\partial \boldsymbol{h}^{(i+1)}}{\partial \boldsymbol{h}^{(i)}}$ are **too small**? ⇨ **Vanishing gradient problem**
  - When these are small, the gradient signal gets smaller and smaller as it back-propagates further
  - So, model weight are updated only with respect to near effects, **not** long-term effects.

- What happens if $\frac{\partial \boldsymbol{h}^{(i+1)}}{\partial \boldsymbol{h}^{(i)}}$ are **too large**? ⇨ **Exploding gradient problem**

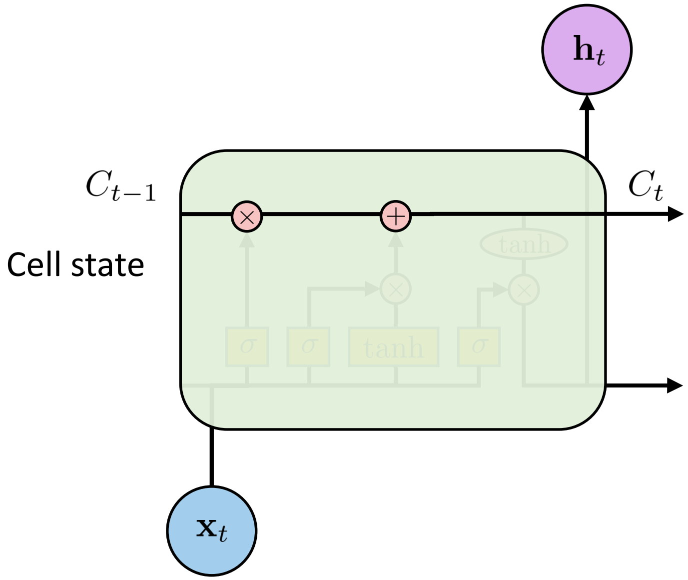
$$\theta^{\text{new}} = \theta^{\text{old}} - \alpha \nabla_\theta J(\theta)$$

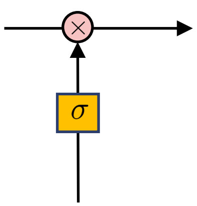
  - This can cause bad updates as the update step of parameters becomes too big
  - In the worst case, this will result in divergence of your network
  - In practice, with a gradient clipping, exploding gradient is **relatively easy to solve**

RNN Architectures: LSTM

- **Long Short-Term Memory (LSTM)** [Hochreiter and Schmidhuber, 1997]
  - A special type of RNN unit, i.e., LSTM networks = RNN composed of LSTM units
  - Explicitly designed RNN to
    - Capture **long-term dependency** $\Longrightarrow$ more robust to vanishing gradient problem

- Core idea behind LSTM
  - With **cell state** (**memory**), it controls **how much to remove or add information**
    - Only linear interactions from the output of each "**gates**" (**prevent vanishing gradient**)



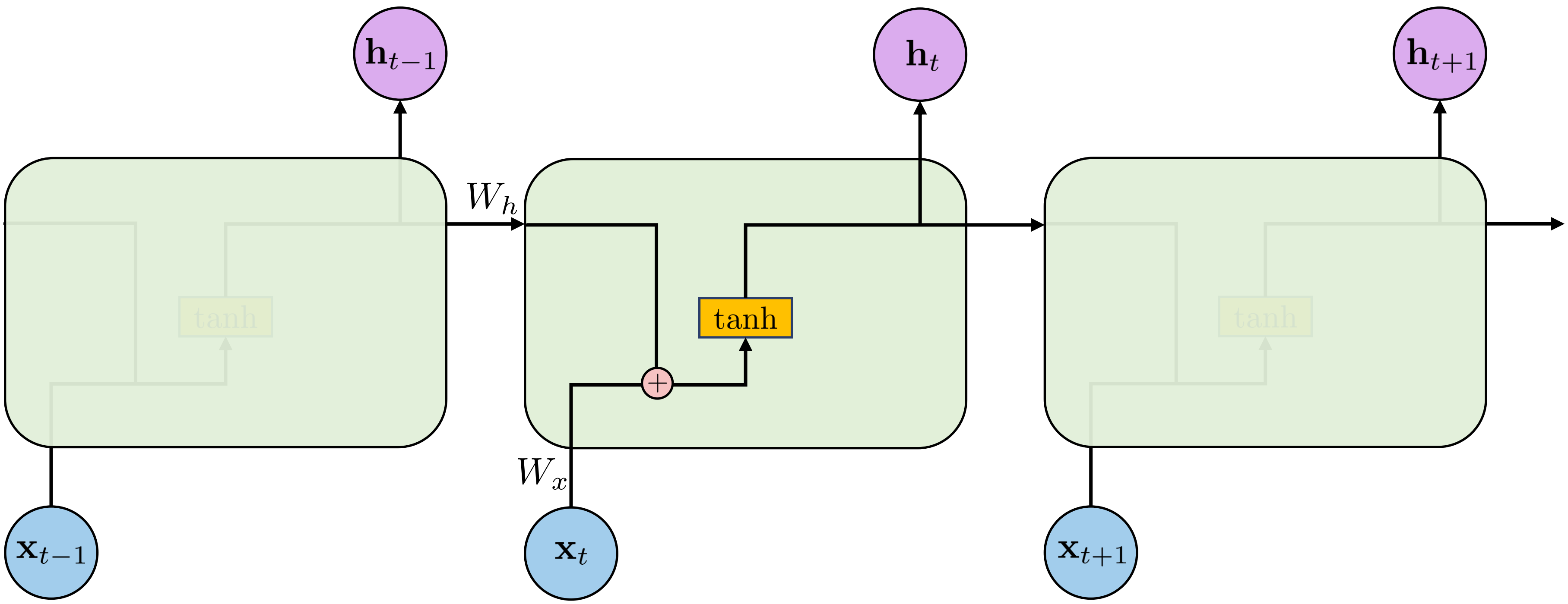


Gates : Way to optionally let information through

RNN Architectures: Vanilla RNN

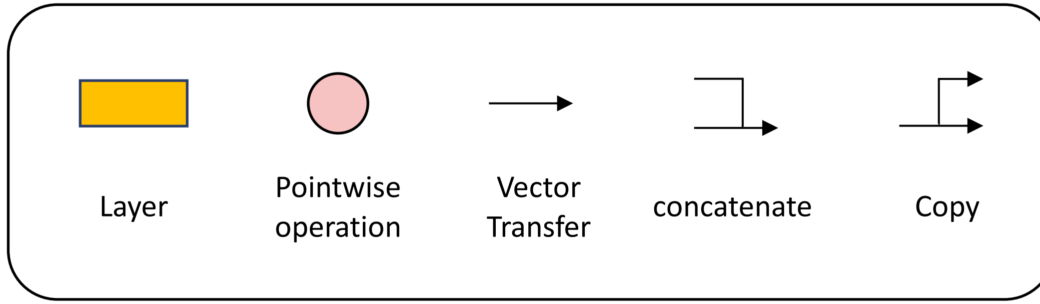
- Repeating modules in **Vanilla RNN** contains a **single layer**

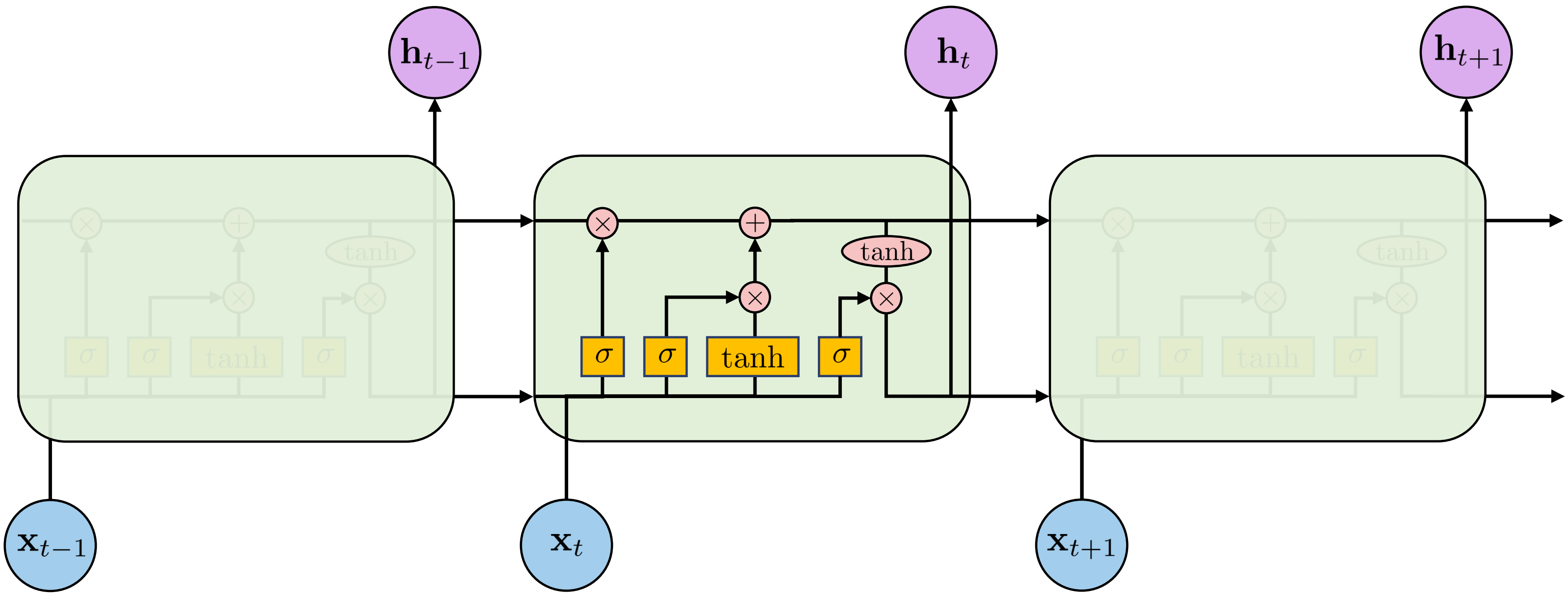
$$\boldsymbol{h}_t = \tanh(W_h \boldsymbol{h}_{t-1} + W_x \boldsymbol{x}_t)$$



RNN Architectures: LSTM

- Repeating modules in **LSTM**

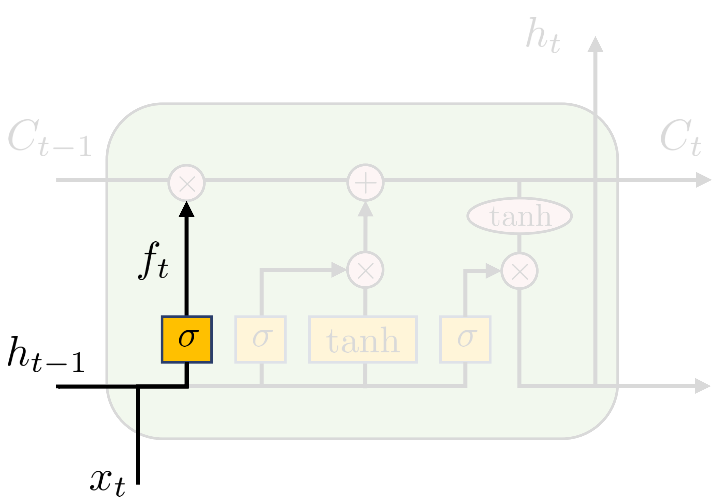




Step 1: Decide what **information** we're going to **throw away** from the **cell state**

- A sigmoid layer called "**Forget gate**" f_t
- Looks at h_{t-1}, x_t and outputs a number between 0 and 1 for each cell state C_{t-1}
  - If 1: completely keep, if 0: completely remove

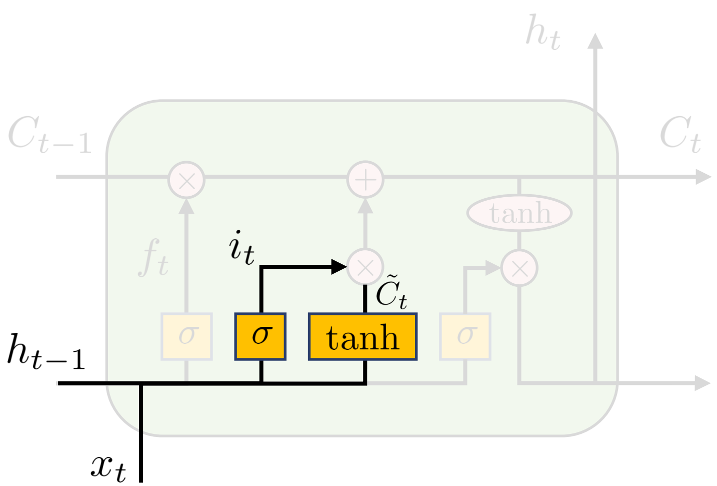
- E.g., language model trying to **predict the next word** based on all previous ones
  - The cell state might include the gender of the present subject so that the correct pronouns can be used
  - When we see a new subject, we want to forget the gender of the old subject



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Step 2: Decide what **information** we're going to **store** in the cell state and **update**

- First, a sigmoid layer called the "**Input gate**" i_t decides which values to update
- Next, a tanh layer creates a **new content** $\tilde{C}_t$ to be written to the



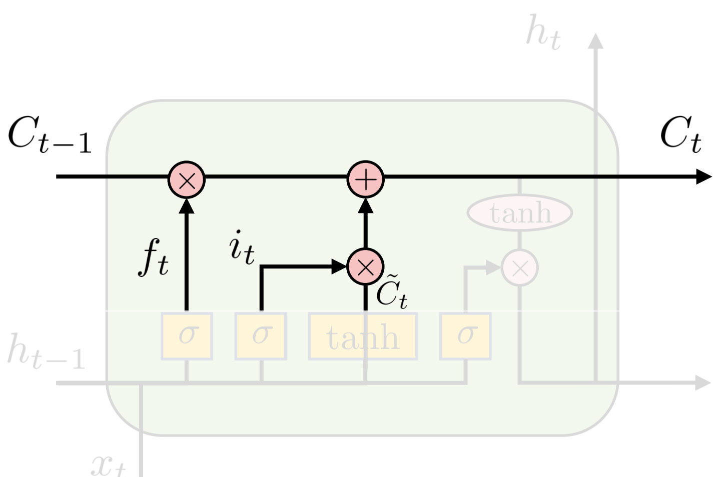
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Step 2: Decide what **information** we're going to **store** in the cell state and **update**

- First, a sigmoid layer called the "**Input gate**" i_t decides which values to update
- Next, a tanh layer creates a **new content** $\tilde{C}_t$ to be written to the

- Then, **update** the old cell state C_{t-1} into the **new cell state** C_t
  - Multiply the old state by f_t (forget gate)
  - Add $i_t * \tilde{C}_t$, new content scaled by how much to update (input gate)



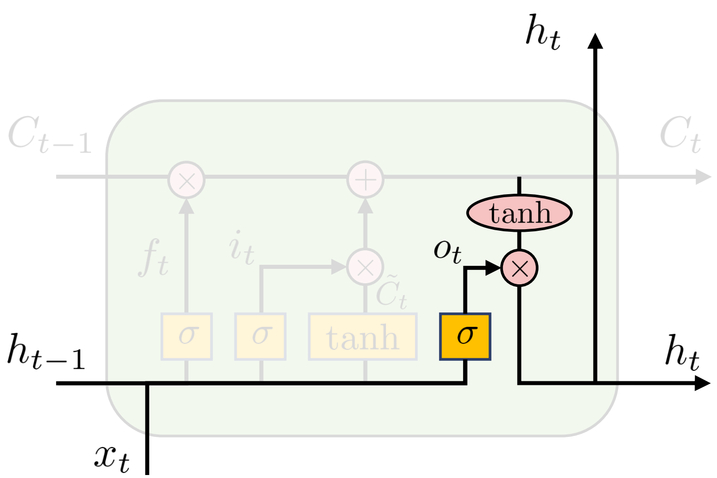
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$\boxed{C_t = f_t * C_{t-1} + i_t * \tilde{C}_t}$$

Step 3: Decide what **information** we're going to **output**

- A sigmoid layer called "**Output gate**" o_t
- First, go through o_t which decides **what parts** of the cell state **to output**
- Then, put the cell state C_t through tanh and multiply it by o_t for hidden state h_t



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

- Overall LSTM operations

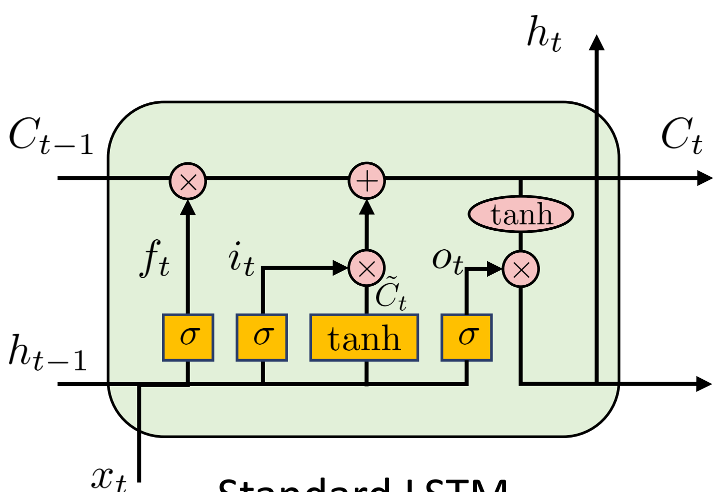
Forget gate: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ Input gate: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

Previous cell state: C_{t-1} New cell content: $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

Updated cell state: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

Output gate: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

Hidden state: $h_t = o_t * \tanh(C_t)$

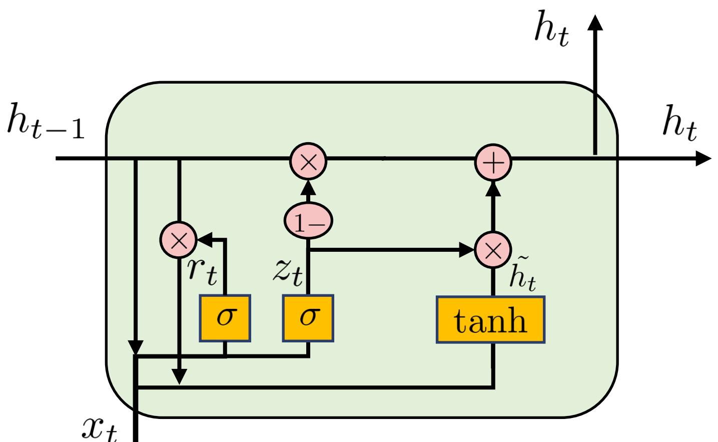


Standard LSTM

RNN Architectures: GRU

- **Gated Recurrent Unit (GRU)** [Cho et.al, 2014]
  - Combines the forget and input gates into a single "**update gate**" z_t
    - Controls the **ratio of information to keep** between previous state and new state
  - **Reset gate** r_t controls how much information to forget when create a new content
  - **Merges** the cell state C_t and hidden state h_t
  - **(+)** Resulting in **simpler model (less weights)** than standard LSTM

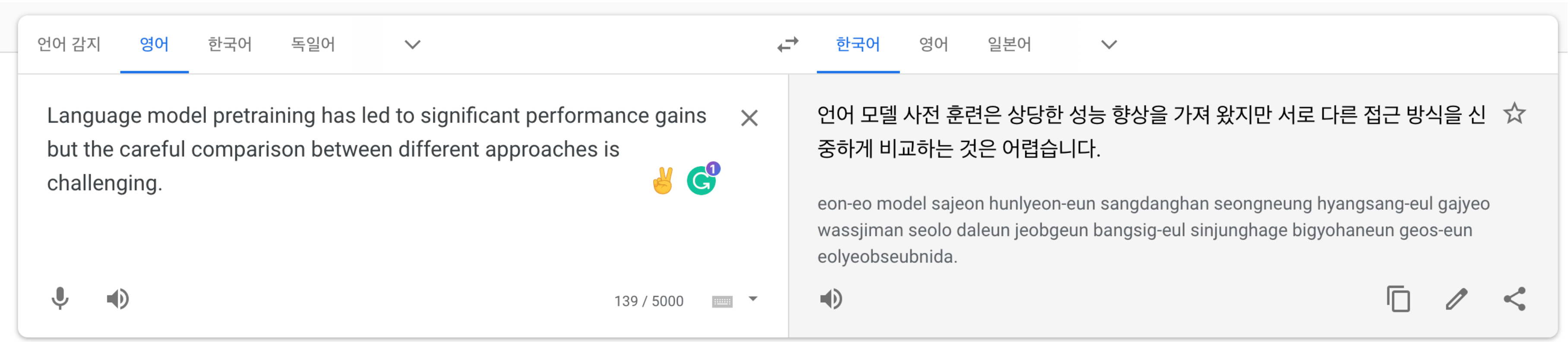
Reset gate: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$	New content: $\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$
Update gate: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$	Hidden state: $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$



Gated Recurrent Unit

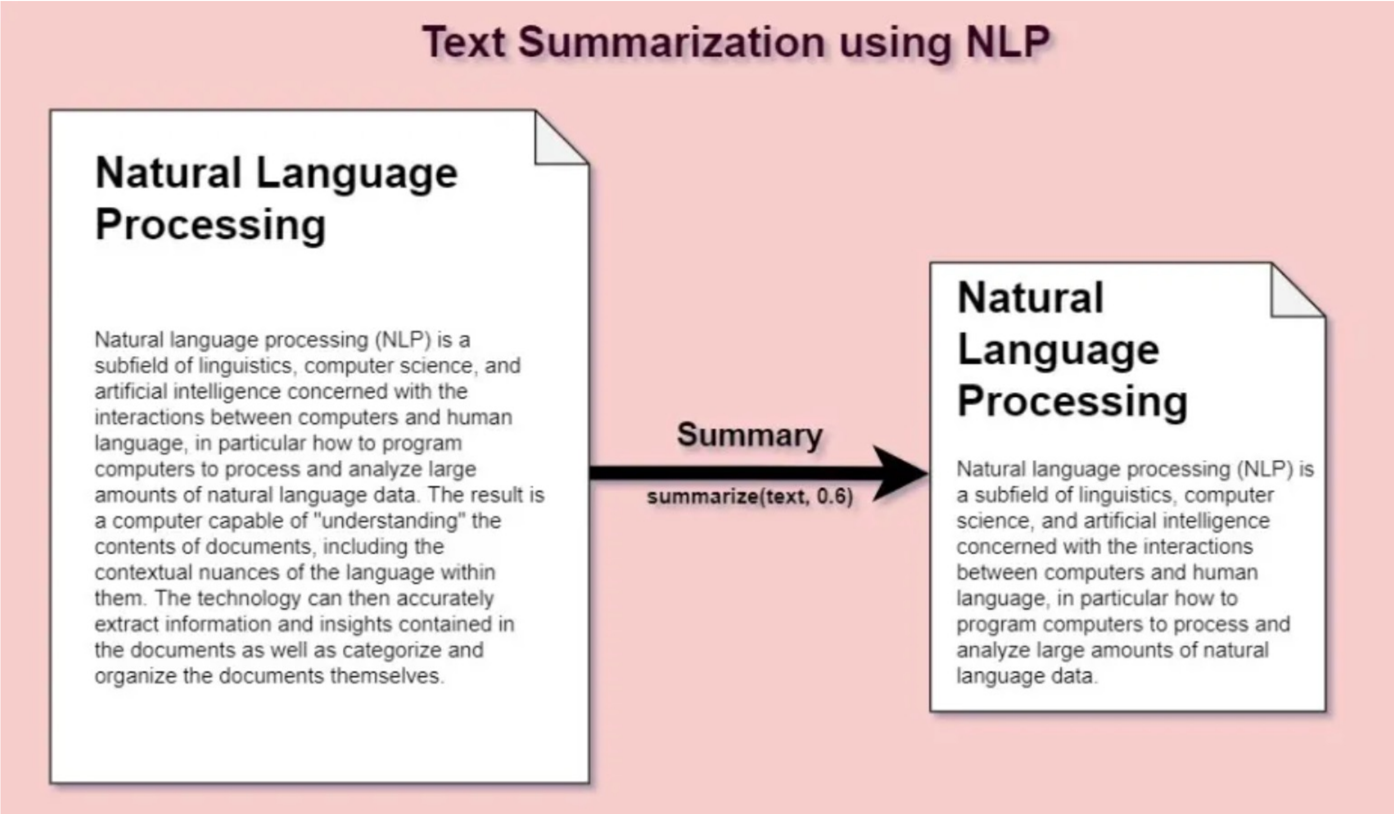
Motivation: Natural Language Processing and Sequence-to-sequence Modeling

- Many natural language processing (NLP) tasks are **Sequence-to-sequence**
  - Given an input sequence, turn it into an output sequence
  - Example: **Translation**



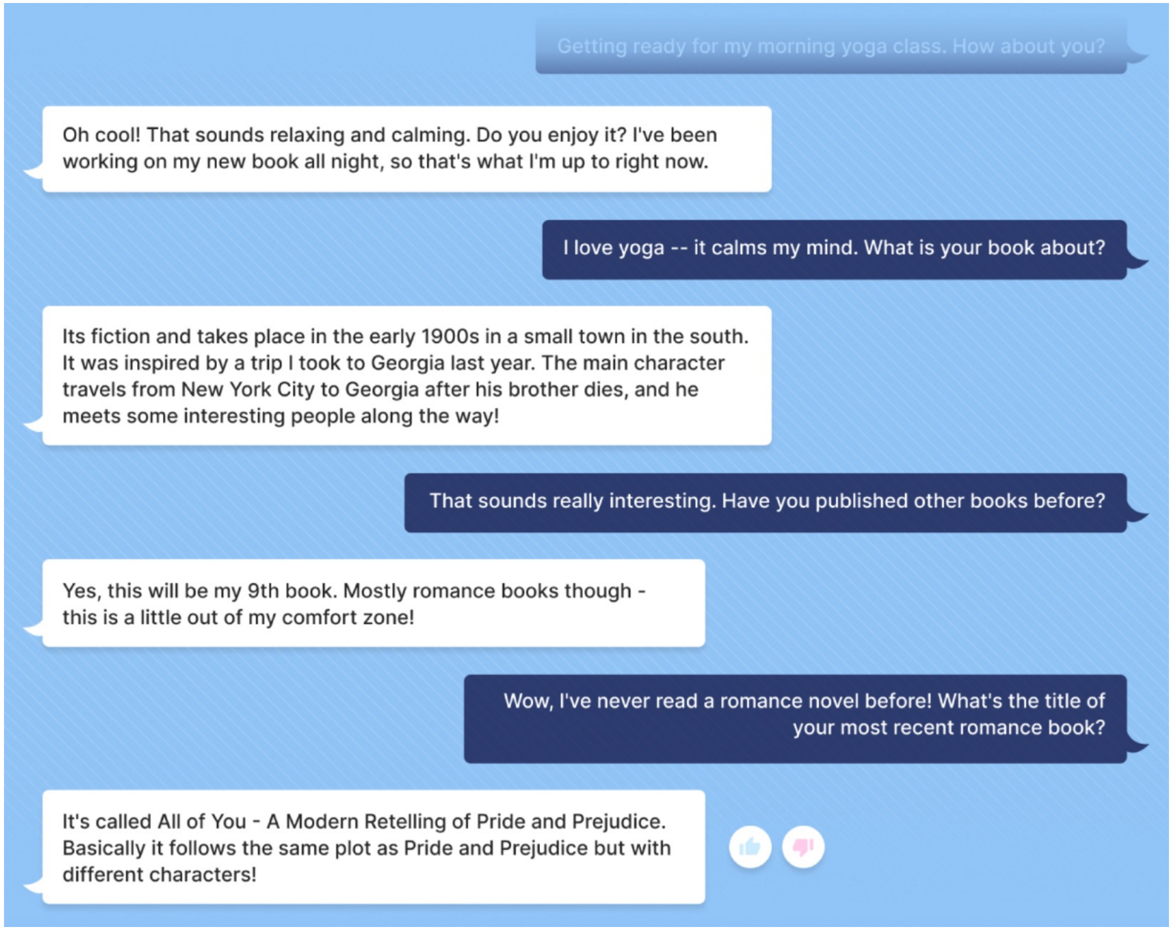
Motivation: Natural Language Processing and Sequence-to-sequence Modeling

- Many natural language processing (NLP) tasks are **Sequence-to-sequence**
  - Given an input sequence, turn it into an output sequence
  - Example: **Text Summarization**



Motivation: Natural Language Processing and Sequence-to-sequence Modeling

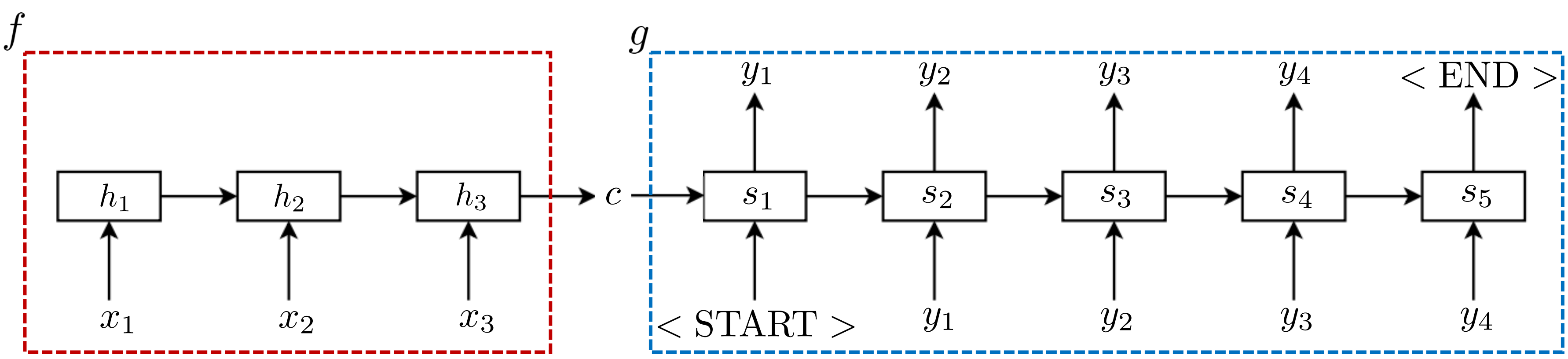
- Many natural language processing (NLP) tasks are **Sequence-to-sequence**
  - Given an input sequence, turn it into an output sequence
  - Example: **ChatBot**



Sequence-to-sequence Model

- Many natural language processing (NLP) tasks is **Sequence-to-sequence**
  - Given an input sequence, turn it into an output sequence

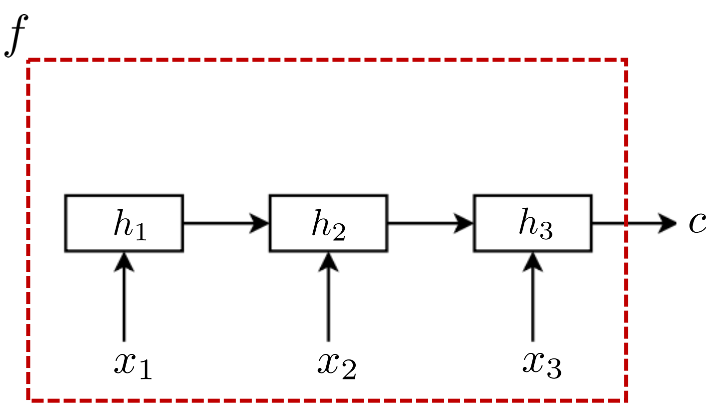
- The core idea of **Sequence-to-sequence** model [Sutskever et al., 2014]
  - **Encoder-Decoder** architecture (input → vector → output)
  - Use one network (**Encoder**) to **read input sequence** at a time for **encoding it into** a fixed-length vector representation **(context)**
  - Use another network (**Decoder**) to extract **output sequence** from context vector



Input sequence $\boldsymbol{x} = (x_1, x_2, x_3)$ and output sequence $\boldsymbol{y} = (y_1, y_2, y_3, y_4)$

Sequence-to-sequence Model

- **Encoder**
  - **Reads the input** sentence $\mathbf{x} = (x_1, \ldots, x_T)$ and **output context** vector c
  - Use RNNs such that $h_t = f(x_t, h_{t-1})$ and $c = q(\{h_1, \ldots, h_T\})$, where f and q are some non-linear functions
  - E.g., LSTMs as f and $q(\{h_1, \ldots, h_T\}) = h_T$ (in the original seq2seq model)



Input sequence $\boldsymbol{x} = (x_1, x_2, x_3)$ and output sequence $\boldsymbol{y} = (y_1, y_2, y_3, y_4)$

Sequence-to-sequence Model

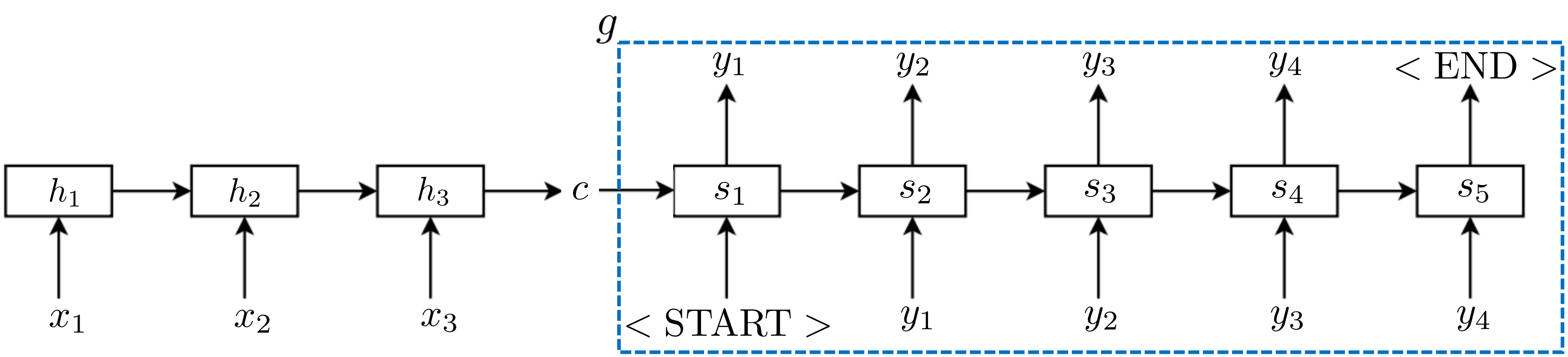
- **Decoder**
  - **Predict the next word** $y_{t'}$ **given the context** vector c and the **previously predicted** words $\{y_1, \dots, y_{t'-1}\}$
  - Defines a probability over the translation $\mathbf{y}$ by **decomposing the joint probability** into the ordered conditionals where $\mathbf{y} = (y_1, \dots, y_T)$.

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t | \{y_1, \dots, y_{t'-1}\}, c),$$

  - The conditional probability is modeled with **another RNN** g as

$$p(y_t | \{y_1, \dots, y_{t'-1}\}, c) = g(y_{t-1}, \underline{s_t}, c),$$

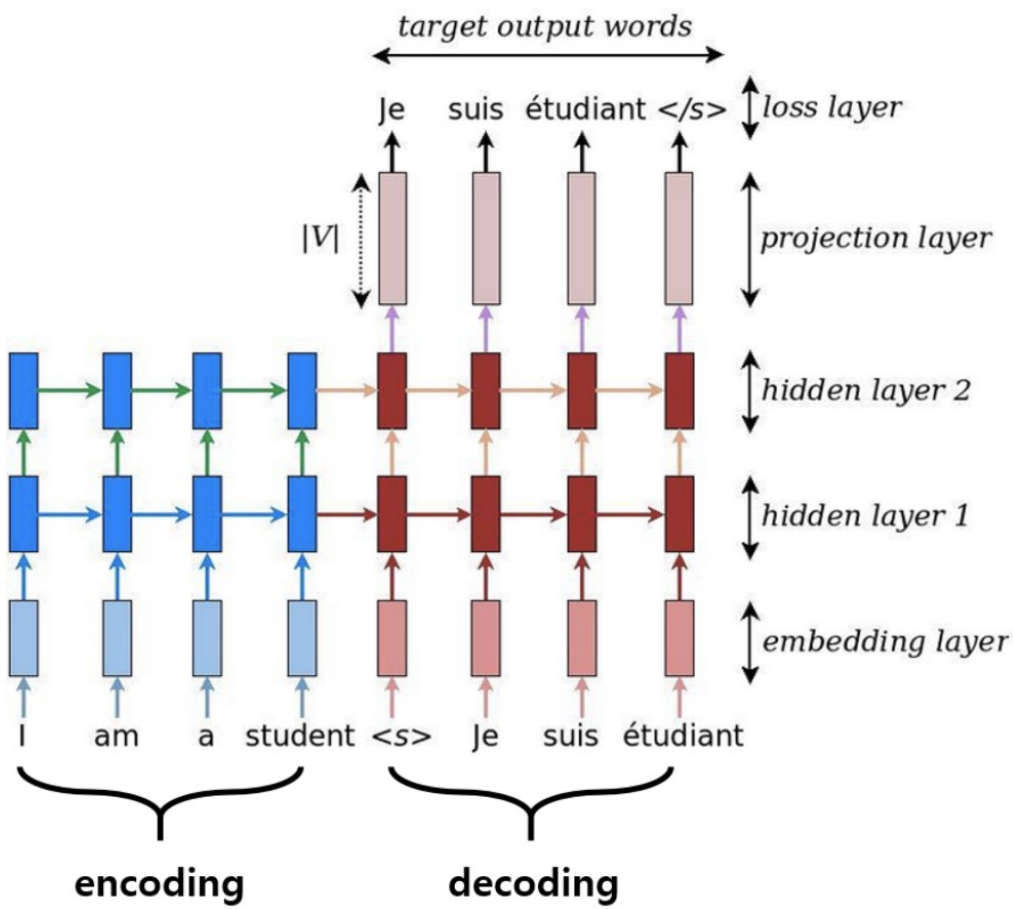
hidden state of the RNN



Input sequence $\boldsymbol{x} = (x_1, x_2, x_3)$ and output sequence $\boldsymbol{y} = (y_1, y_2, y_3, y_4)$

Sequence-to-sequence Model

- Example of the seq2seq model
  - For English → French task
  - With 2-layer LSTM for encoder and encoder



- Results on WMT’14 English to French dataset [Sutskever et al., 2014]
  - Measure : BLEU (Bilingual Evaluation Understudy) score
    - Widely used quantitative measure for MT task
  - On **par with the state-of-the-art** system (without using neural network)
  - Achieved **better results than the previous baselines**

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
State of the art [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	$\sim$45

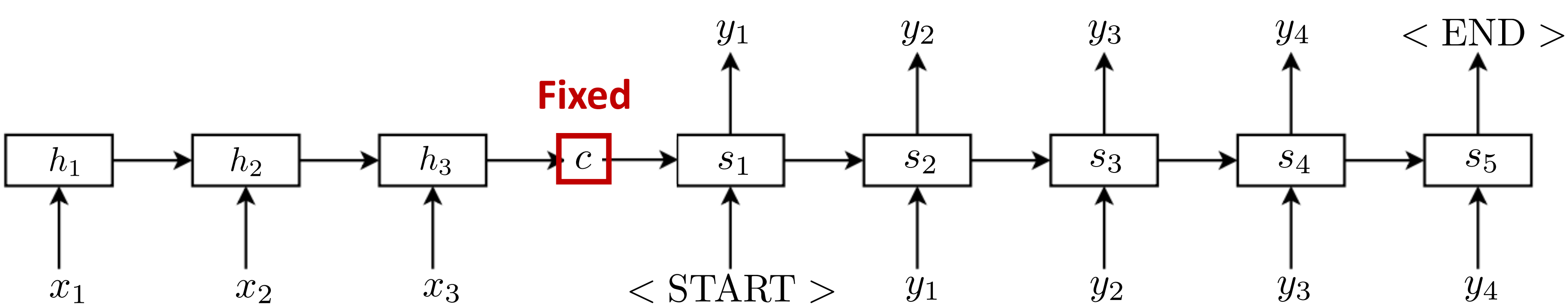
- Seq2seq with RNNs is **simple but very powerful** in MT task

Attention-based Sequence-to-sequence Model

- Problem of original seq2seq (or encoder-decoder) model
  - Need to **compress** all the necessary information of a source sentence into a **fixed context vector**
    - All decoding steps use an identical context along with previous outputs

$$p(y_t|\{y_1,\ldots,y_{t'-1}\},c) = g(y_{t-1}, s_t, \underline{c}),$$

  - But, each step of decoding **requires different part** of the source sequence
    - E.g., Step1: “**I** love you” ⟶ “**나는** 너를 사랑해”
      Step2: “I **love** you” ⟶ “나는 너를 **사랑해**”
    - Hence, difficult to cope with long sentences…

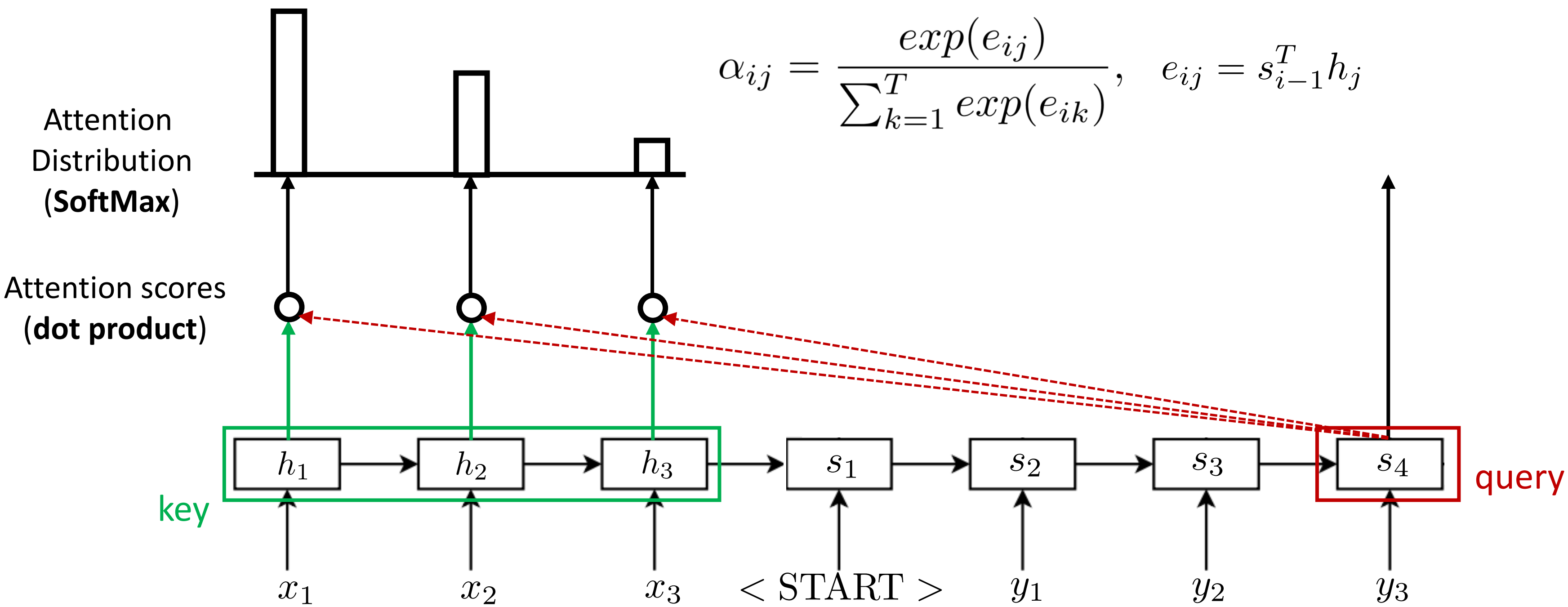


Input sequence $\boldsymbol{x} = (x_1, x_2, x_3)$ and output sequence $\boldsymbol{y} = (y_1, y_2, y_3, y_4)$

Attention-based Sequence-to-sequence Model

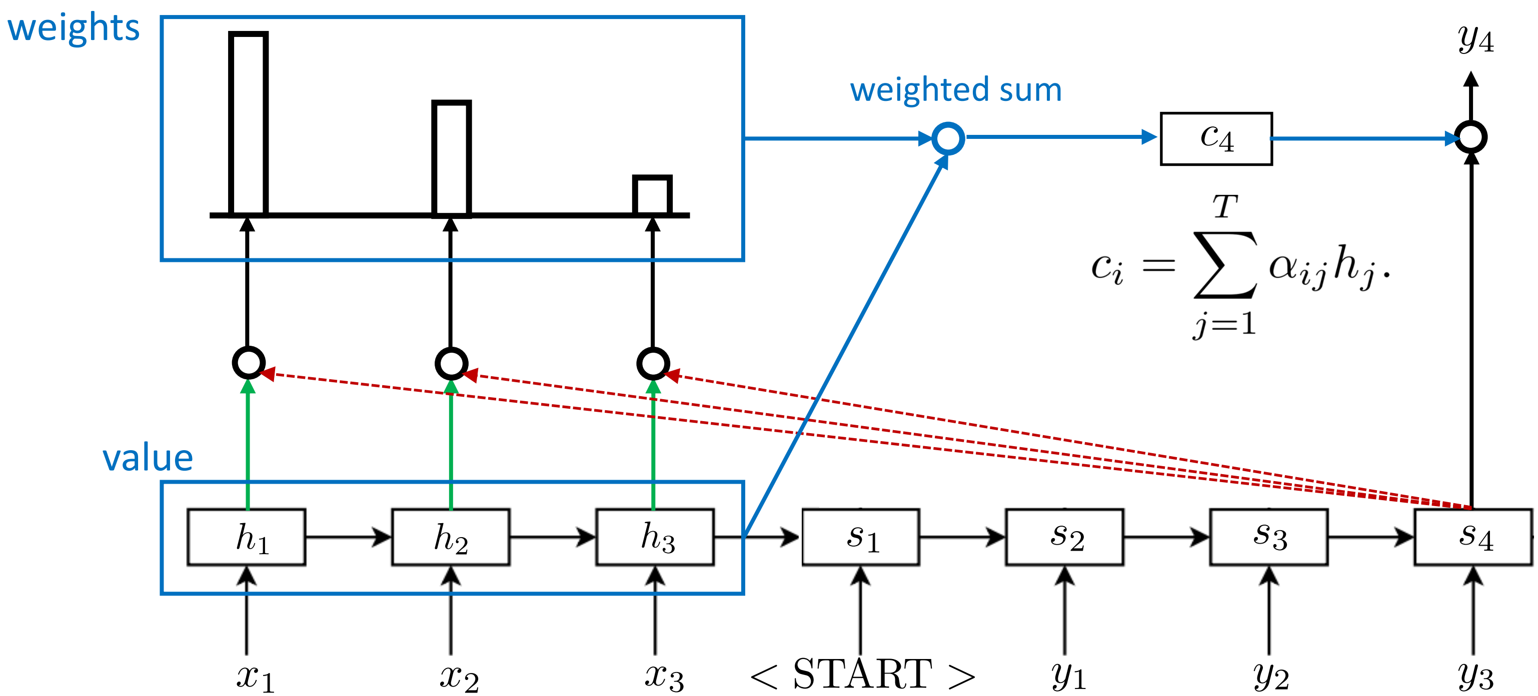
- Extension of seq2seq model with **attention** mechanism [Bahdanau et al., 2015]
  - **Core idea**: on each step of the decoder, **focus on a particular part** of the source sequence using a **direct connection (attention)** to the encoder states
  - Dependent on the query with key, **attention** is a technique to compute a weighted sum of the values
    - Query: decoder's hidden state, key and value: encoder's hidden states
    - α_{ij} is a **relative importance** which means how well the inputs around position i and the output position j match.

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})}, \quad e_{ij} = s_{i-1}^{T} h_j$$



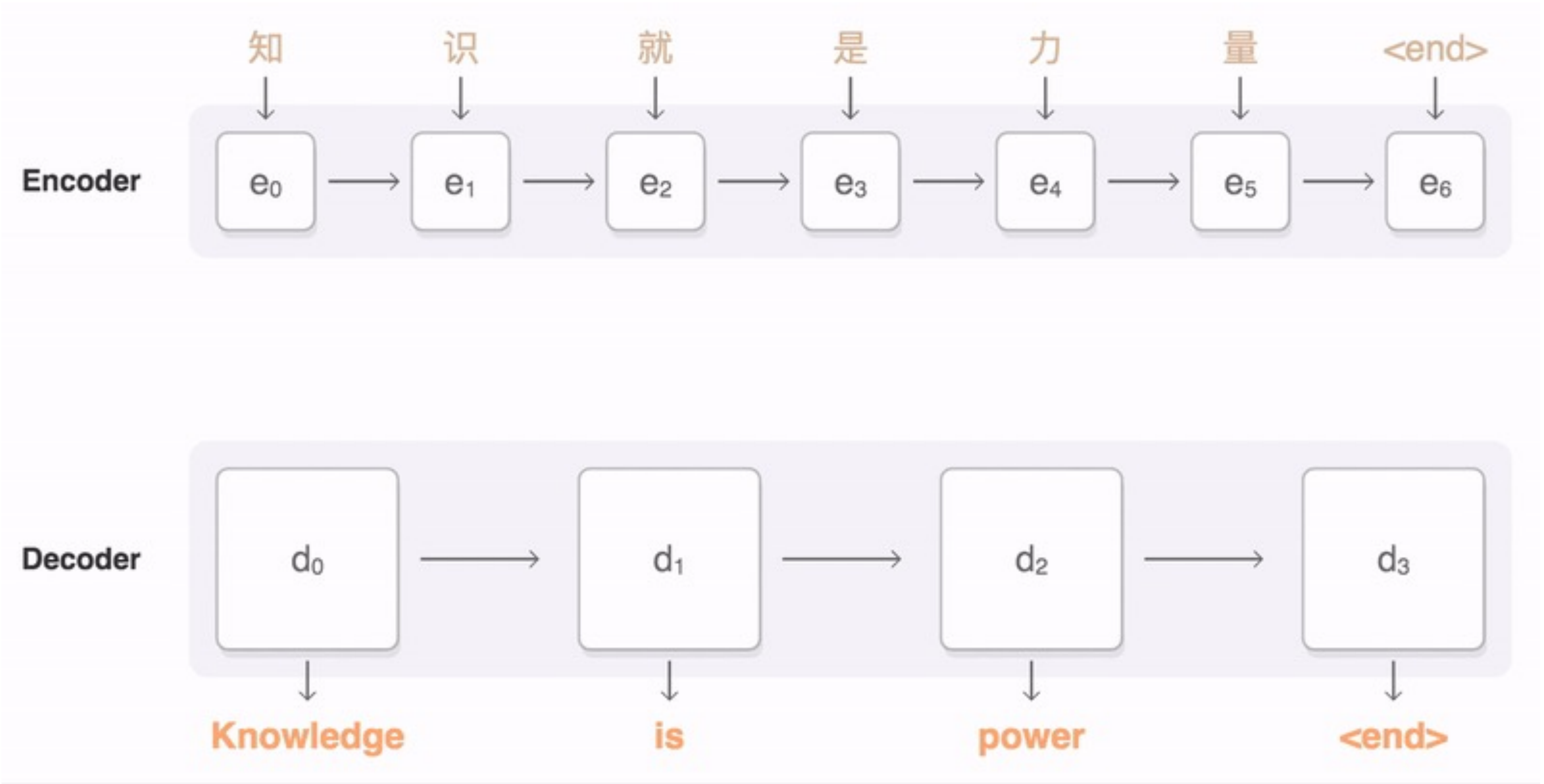
Attention-based Sequence-to-sequence Model

- Extension of seq2seq model with **attention** mechanism [Bahdanau et al., 2015]
  - **Core idea**: on each step of the decoder, **focus on a particular part** of the source sequence using a **direct connection (attention)** to the encoder states
  - Dependent on the query with key, **attention** is a technique to compute a weighted sum of the values
    - Query: decoder's hidden state, key and value: encoder's hidden states
    - The context vector c_i is computed as **weighted sum** of h_i



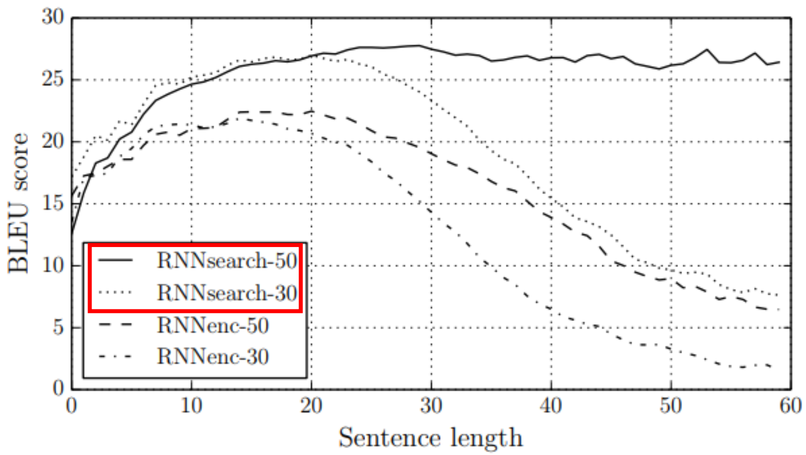
Attention-based Sequence-to-sequence Model

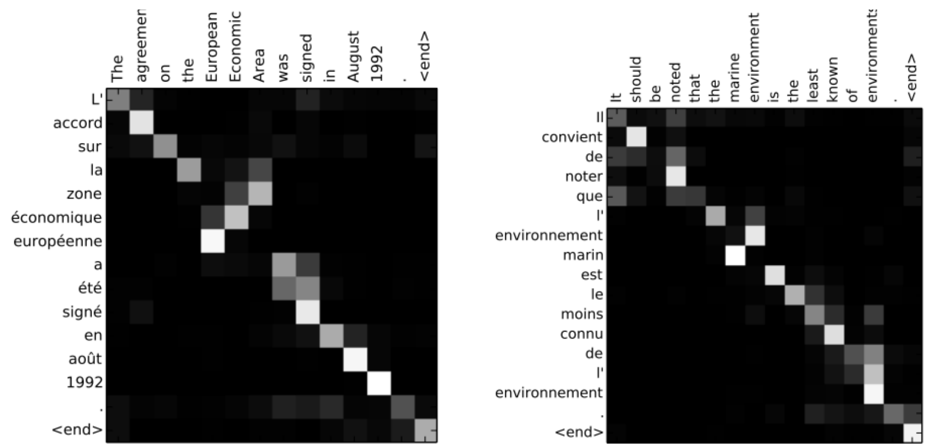
- **Graphical illustration** of seq2seq with **attention**
  - E.g., Chinese to English



Attention-based Sequence-to-sequence Model

- **Results**
  - RNNsearch (with attention) is better than RNNenc (vanilla seq2seq)
  - RNNsearch-50: model trained with sentences of length up to 50 words



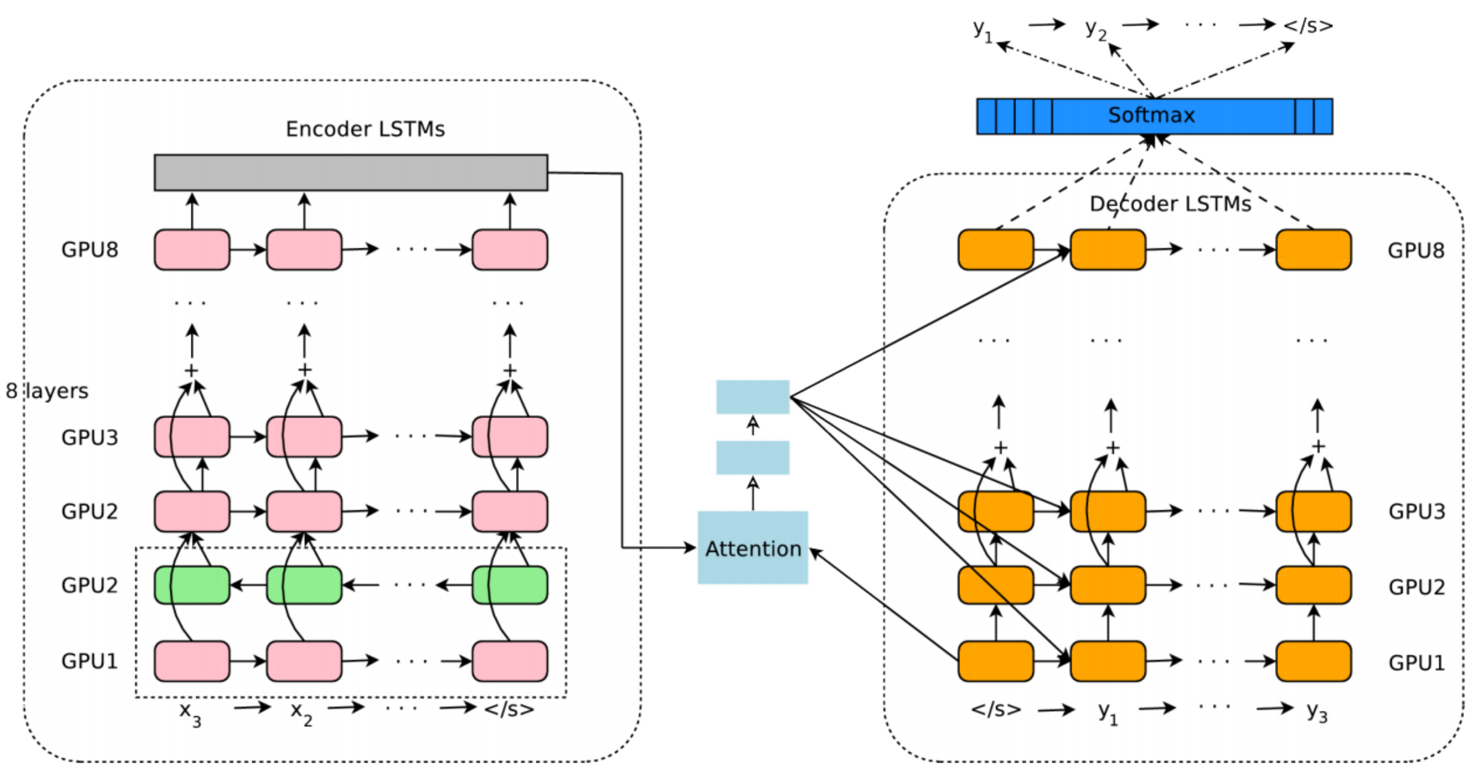


(a) (b)

Sample alignment results (attention map)

Attention-based Sequence-to-sequence Model: Google's NMT

- **Google's NMT** [Wu et al., 2016]
  - Improves over previous NMT systems on **accuracy** and **speed**
  - **8-layer LSTMS** for encoder/decoder with **attention**
    - Achieve **model parallelism** by assigning each LSTM layer into different GPUs
    - Add **residual connections in standard LSTM**
    - … and lots of domain-specific details to apply it to production model



Attention-based Sequence-to-sequence Model: Google's NMT

- **Google's NMT** [Wu et al., 2016]
  - Improves over previous NMT systems on **accuracy** and **speed**
  - **8-layer LSTMS** for encoder/decoder with **attention**
  - State-of-the-art results on various MT datasets and comparable with Human expert

Table 5: Single model results on WMT En→De (newstest2014)

Model	BLEU	CPU decoding time per sentence (s)
Word	23.12	0.2972
Character (512 nodes)	22.62	0.8011
WPM-8K	23.50	0.2079
WPM-16K	24.36	0.1931
WPM-32K	24.61	0.1882
Mixed Word/Character	24.17	0.3268
PBMT [6]	20.7	
RNNSearch [37]	16.5	
RNNSearch-LV [37]	16.9	
RNNSearch-LV [37]	16.9	
Deep-Att [45]	20.6	

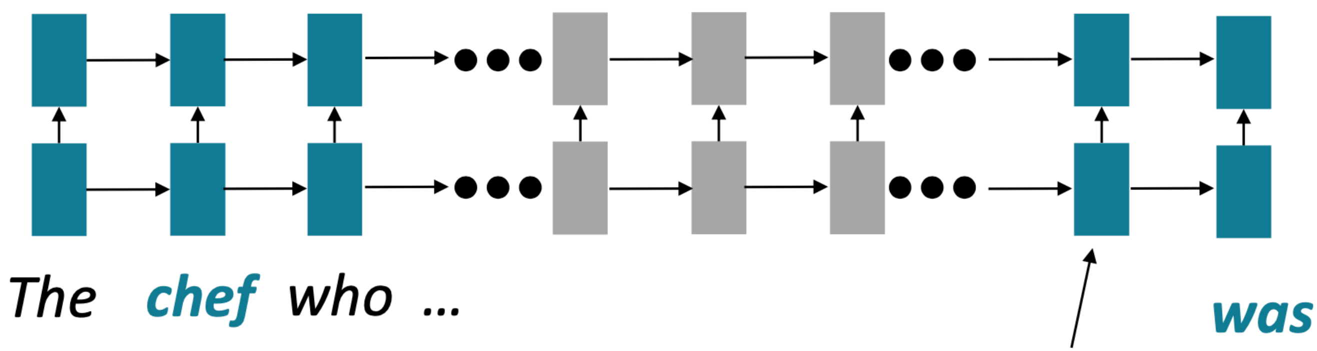
GNMT with different configurations

Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

Limitations with Recurrent Models

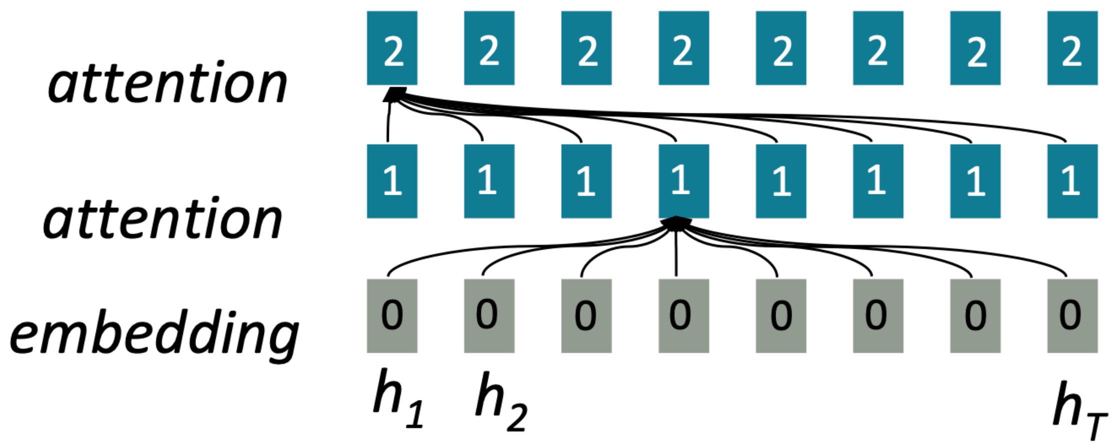
- Although RNNs show remarkable successes, there are **fundamental issues**:
  1. **O(sequence length)** steps for distant word pairs to interact means
     - Hard to learn long-distance dependencies because of gradient problems
  2. Forward/backward passes have **O(sequence length)** unparallelizable operations
     - Future RNN hidden states can't be computed before past states have been computed
     - This aspect inhibits training on the very large datasets



Info of **chef** has gone through **O(sequence length)** many layers

- Although RNNs show remarkable successes, there are **fundamental issues**:
  1. **O(sequence length)** steps for distant word pairs to interact means
  2. Forward/backward passes have **O(sequence length)** unparallelizable operations

- In contrast, **attention has some advantages** in these aspects:
  1. Maximum interaction distance: **O(1)**
     - Since all words interact at each layer
  2. Number of unparallelizable operations does **not increase with respect to length**



All words can attend to all words in previous layer

Limitations with Recurrent Models

- Although RNNs show remarkable successes, there are **fundamental issues**:
  1. **O(sequence length)** steps for distant word pairs to interact means
  2. Forward/backward passes have **O(sequence length)** unparallelizable operations

- In contrast, **attention has some advantages** in these aspects:
  1. Maximum interaction distance: **O(1)**
     - Since all words interact at each layer
  2. Number of unparallelizable operations does **not increase with respect to length**

Q. Then, can we design an architecture **only using attention** modules?

- Remark. We saw attention from the **decoder to the encoder**;
  but here, we'll think about attention **within a single sentence**.

Overview

Part 1. Basics

- RNN to LSTM
- Sequence-to-sequence Model
- Attention-based NLP Model

Part 2. Advanced Topics

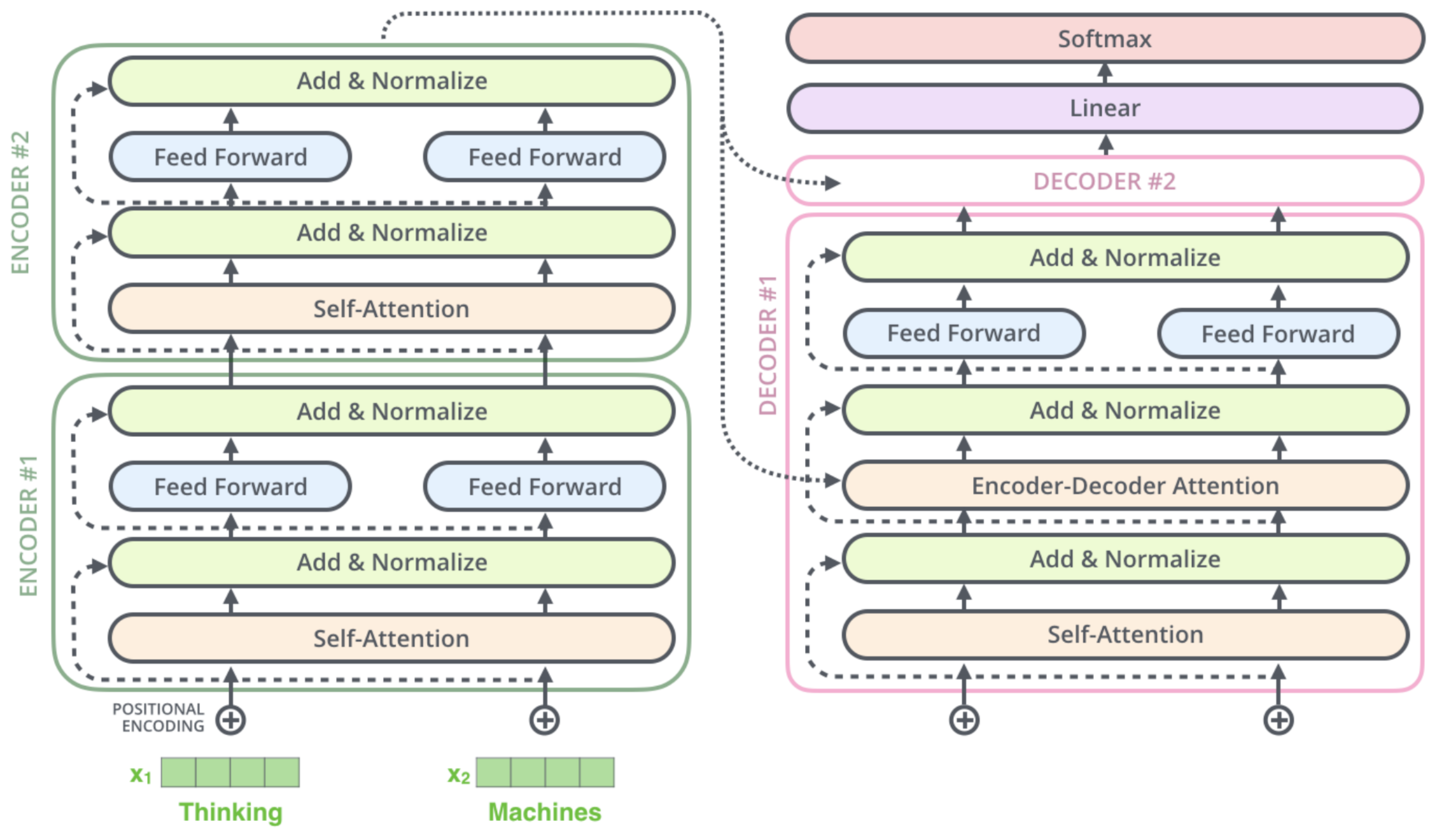
- Transformer (self-attention)
- Pre-training of Transformers and Language Models
- Large Language Models: GPT-3 and emerging properties

Part 3. Recent Advances in Large Language Models

- Large language models beyond GPT-3
- Better training schemes for large language model
- Applications: ChatBot (e.g., ChatGPT)

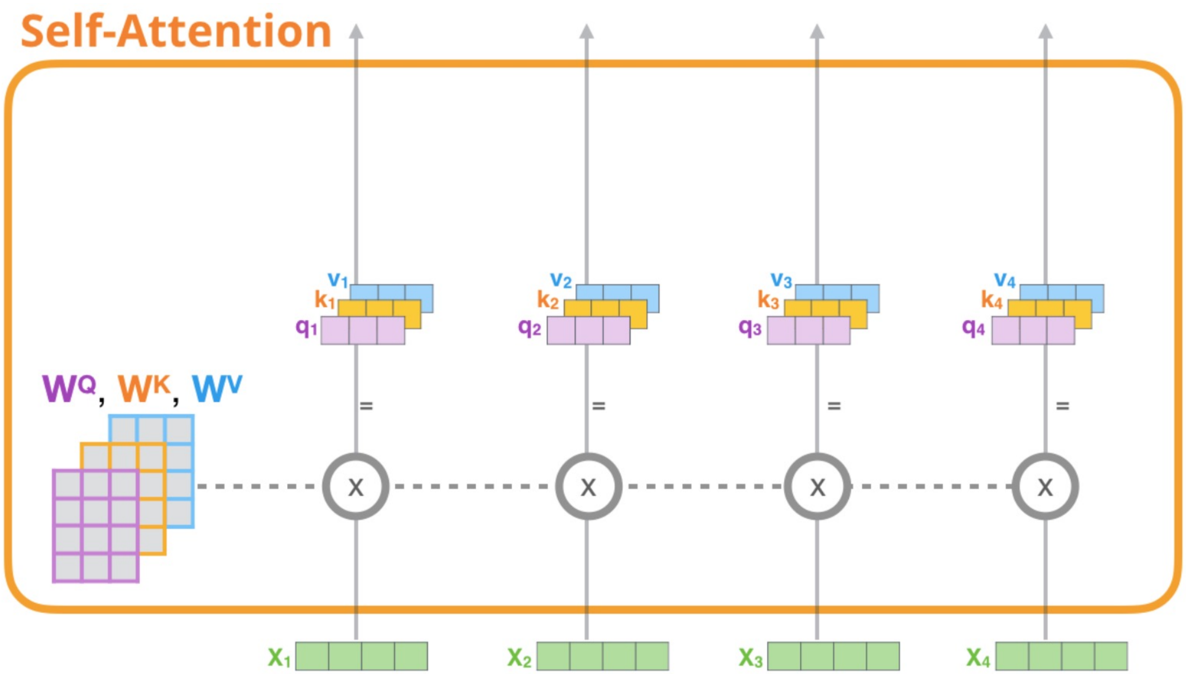
Transformer (Self-attention)

- Transformer [Vaswani et al., 2017] has an **encoder-decoder** structure and they are composed of multiple block with **multi-head (self) attention** module



- **Self-attention**
  - **Recall**: Attention operates on query, key, and value
    - Query is decoder's hidden state, key and value are encoder's hidden states in seq2seq
  - In self-attention, the query, key, and value are drawn from the **same source**
    1. For each input x_i, create query, key, and value vectors q_i, k_i, v_i by multiplying **learnable** weight matrices

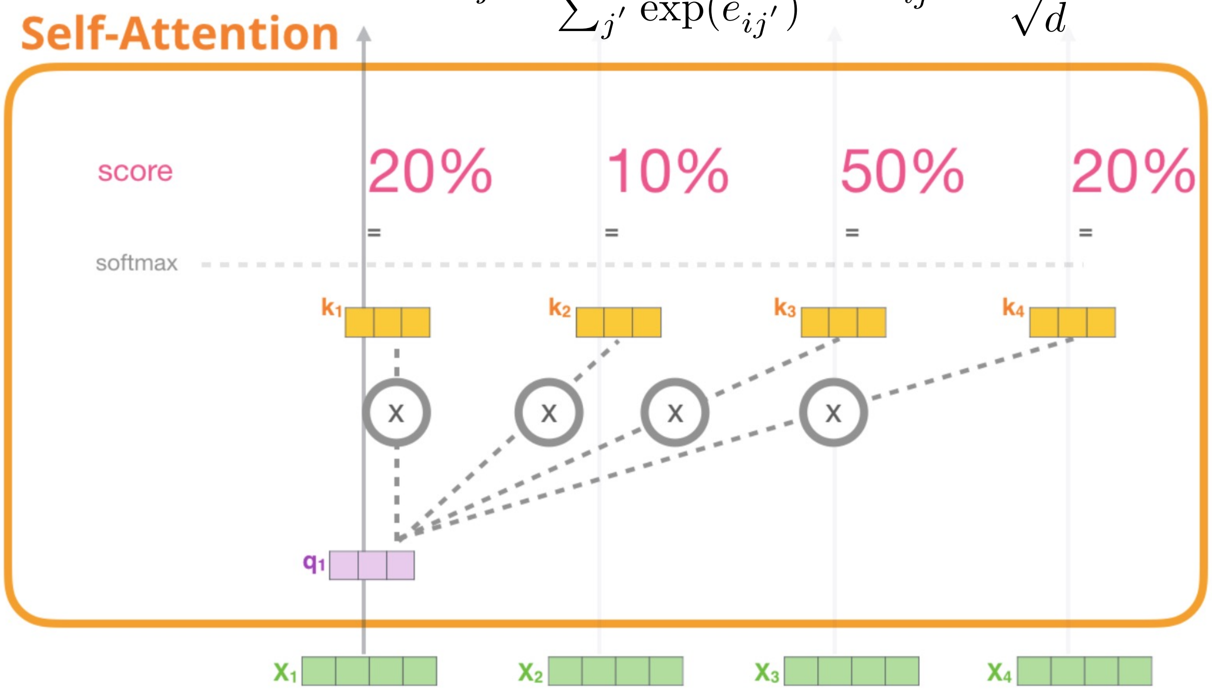
$$q_i = W^Q x_i, k_i = W^k x_i, v_i = W^V x_i$$



Transformer (Self-attention)

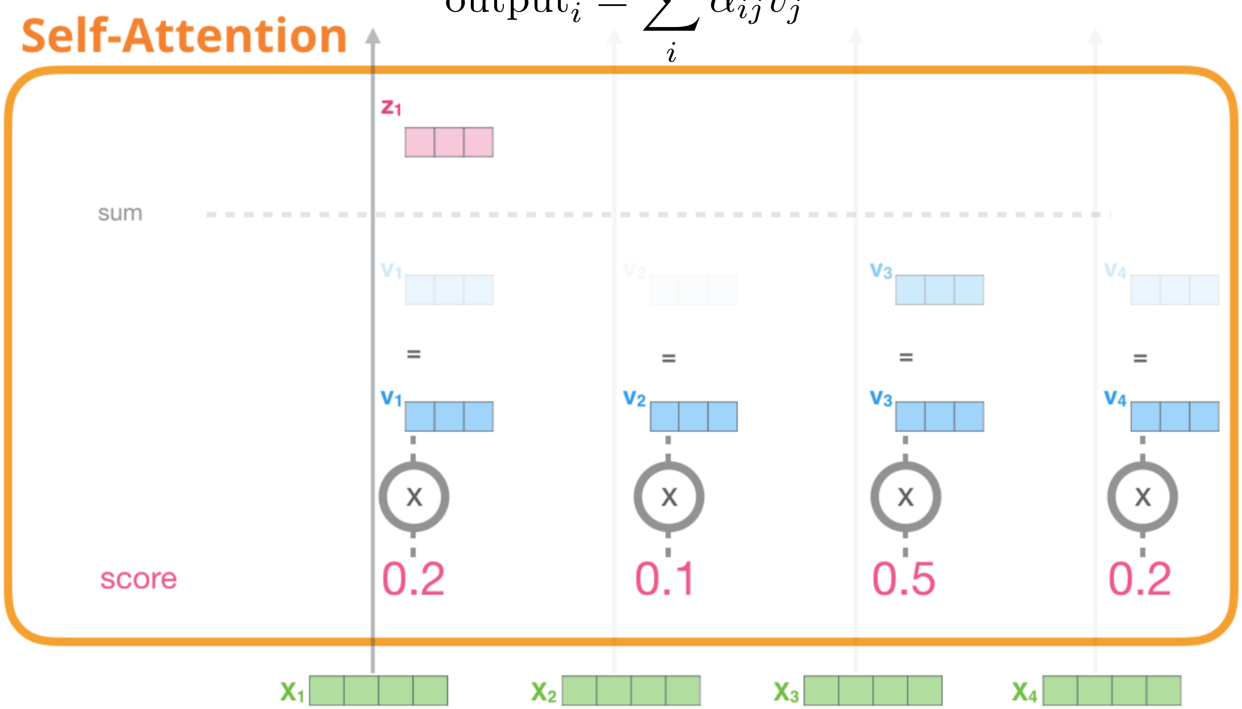
- **Self-attention**
  - **Recall**: Attention operates on query, key, and value
    - Query is decoder's hidden state, key and value are encoder's hidden states in seq2seq
  - In self-attention, the query, key, and value are drawn from the **same source**
    1. For each input x_i, create query, key, and value vectors q_i, k_i, v_i
    2. Multiply (**dot product**) the current query vector, by all the key vectors, to get a **score** α_{ij} of **how well they match**

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{j'} \exp(e_{ij'})} \qquad e_{ij} = \frac{q_i^T k_j}{\sqrt{d}}$$



- **Self-attention**
  - **Recall**: Attention operates on query, key, and value
    - Query is decoder's hidden state, key and value are encoder's hidden states in seq2seq
  - In self-attention, the query, key, and value are drawn from the **same source**
    1. For each input x_i, create query, key, and value vectors q_i, k_i, v_i
    2. Multiply (**dot product**) the current query vector, by all the key vectors, to get a **score** α_{ij}
    3. Multiply the value vectors by the scores, then **sum up**

$$\text{output}_i = \sum_i \alpha_{ij} v_j$$

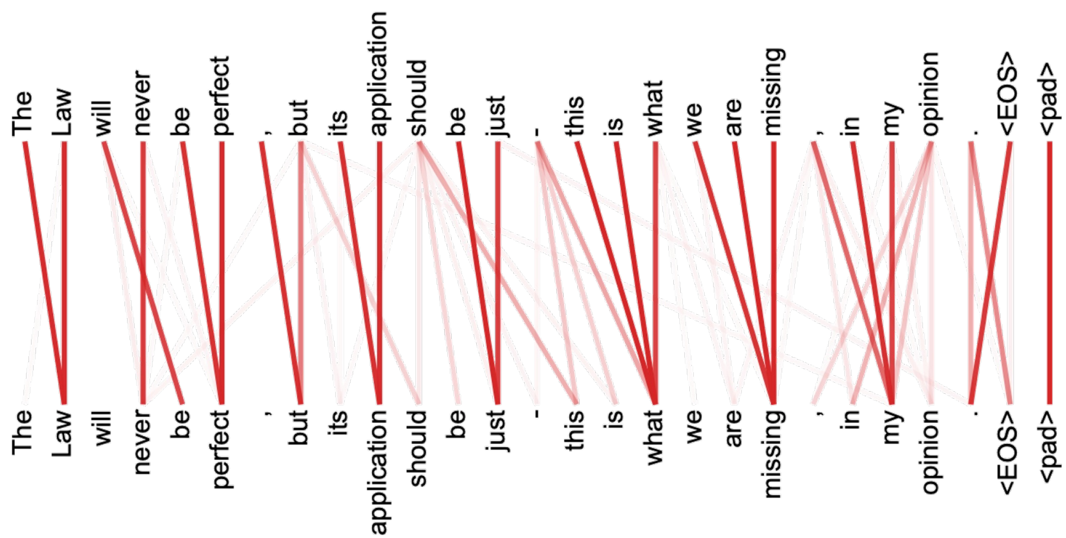


- **Self-attention**
  - **Recall**: Attention operates on query, key, and value
    - Query is decoder's hidden state, key and value are encoder's hidden states in seq2seq
  - In self-attention, the query, key, and value are drawn from the **same source**
    1. For each input x_i , create query, key, and value vectors q_i, k_i, v_i
    2. Multiply (**dot product**) the current query vector, by all the key vectors, to get a **score** α_{ij}
    3. Multiply the value vectors by the scores, then **sum up**
  - Hence, self-attention is **effective to learn the context** within given sentence
    - It's easier than recurrent layer to be parallelized and model the long-term dependency

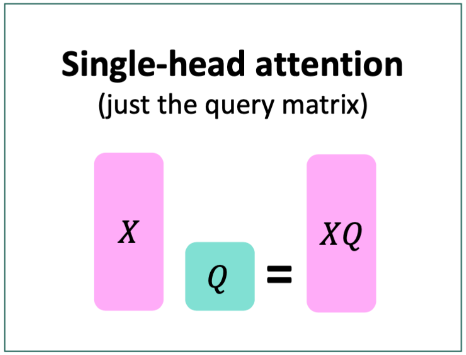
Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

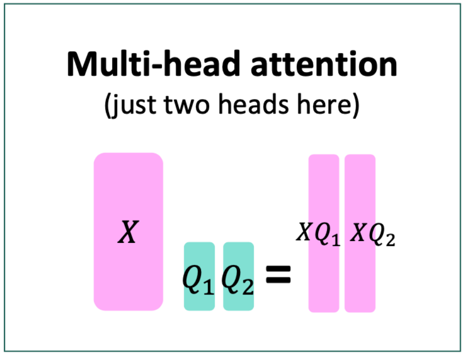
Transformer (Self-attention)

- **Self-attention**
  - **Recall**: Attention operates on query, key, and value
    - Query is decoder's hidden state, key and value are encoder's hidden states in seq2seq
  - In self-attention, the query, key, and value are drawn from the **same source**
    1. For each input x_i, create query, key, and value vectors q_i, k_i, v_i
    2. Multiply (**dot product**) the current query vector, by all the key vectors, to get a **score** α_{ij}
    3. Multiply the value vectors by the scores, then **sum up**
  - Hence, self-attention is **effective to learn the context** within given sentence
    - It's easier than recurrent layer to be parallelized and model the long-term dependency
    - It also provides an **interpretability** of learned representation

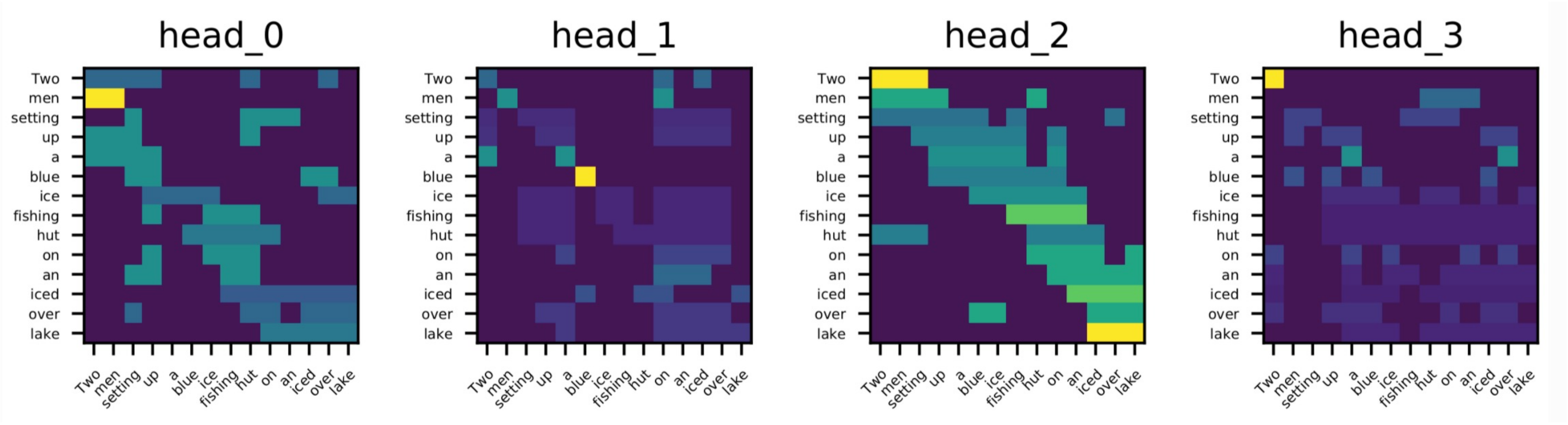


- **Multi-head attention**
  - Applying **multiple attentions at once** to look in multiple places in the sentence
    - To prevent the increase of computation, original attentions weights are **divided**

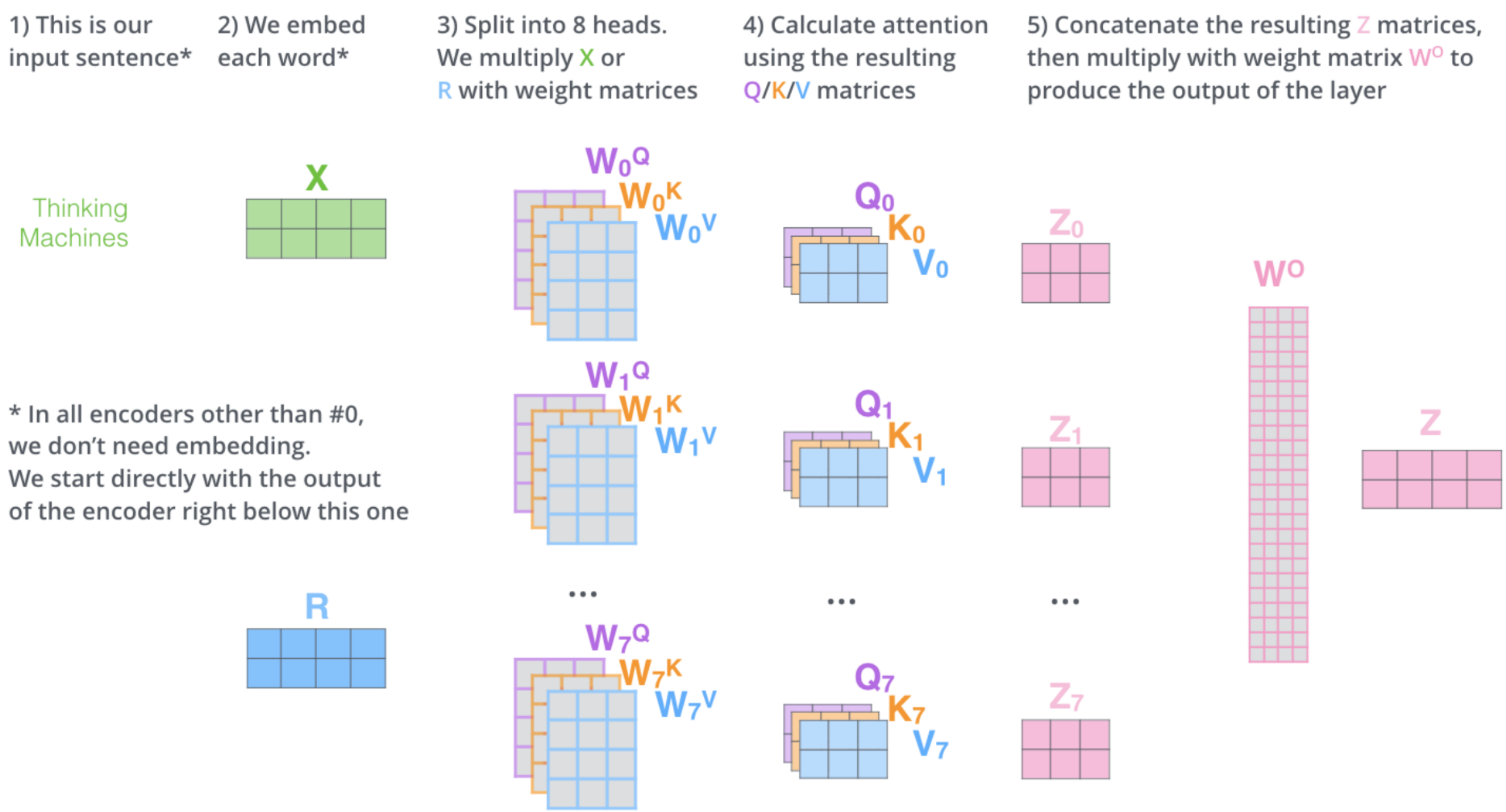




Same amount of computation as single-head self-attention



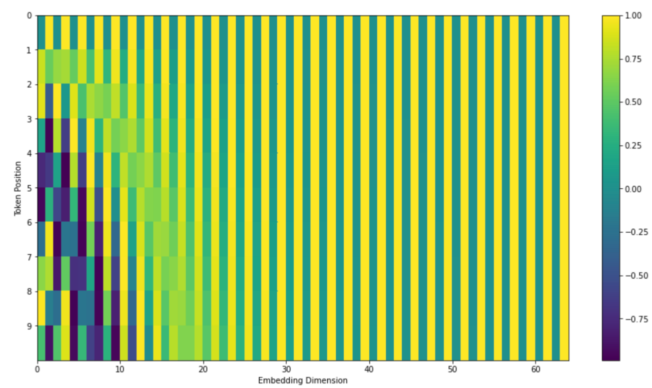
- **Multi-head attention**
  - Applying **multiple attentions at once** to look in multiple places in the sentence

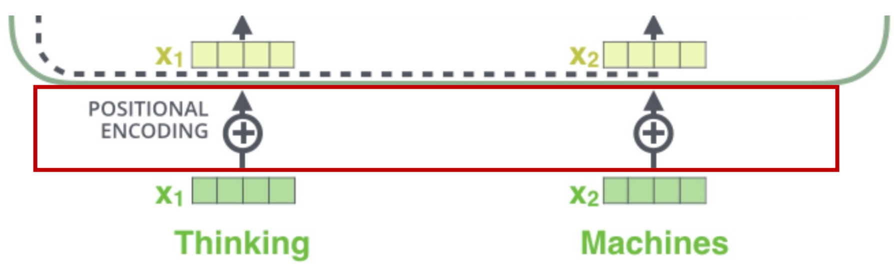


- **Encoder**
  - Self-attention is **invariant to order** of input sequence
    - To represent the order of sequence, **positional encoding** is added to input embeddings at the **bottoms of the encoder and decoder stacks**
  - Fixed sine and cosine functions are used for each position pos and dimension i

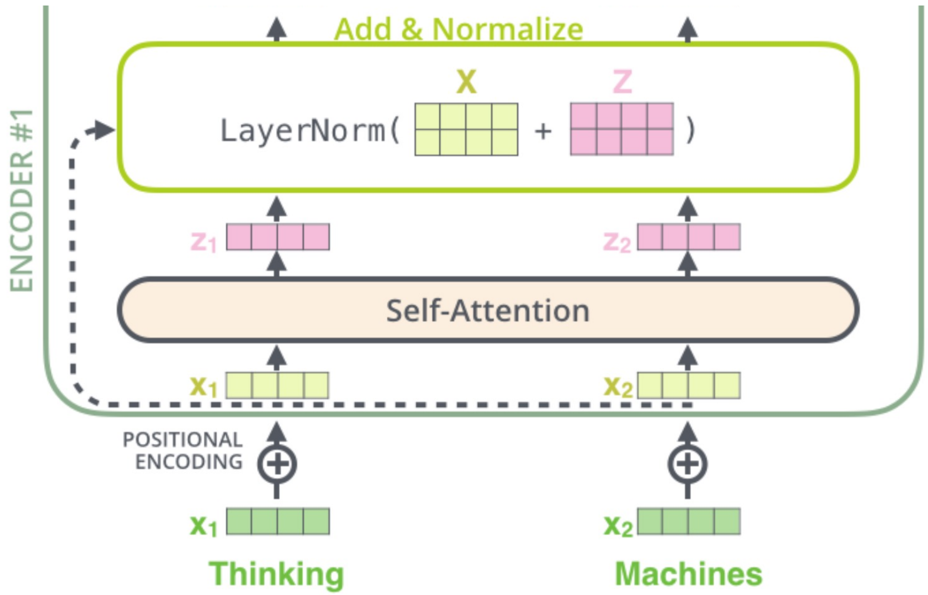
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}}) \quad PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

    - PE_{pos+k} can be derived as a linear function of PE_{pos} → easier to learn a relative position
    - Compare to learning encoding, it's better for **extrapolation** (not encountered in training)

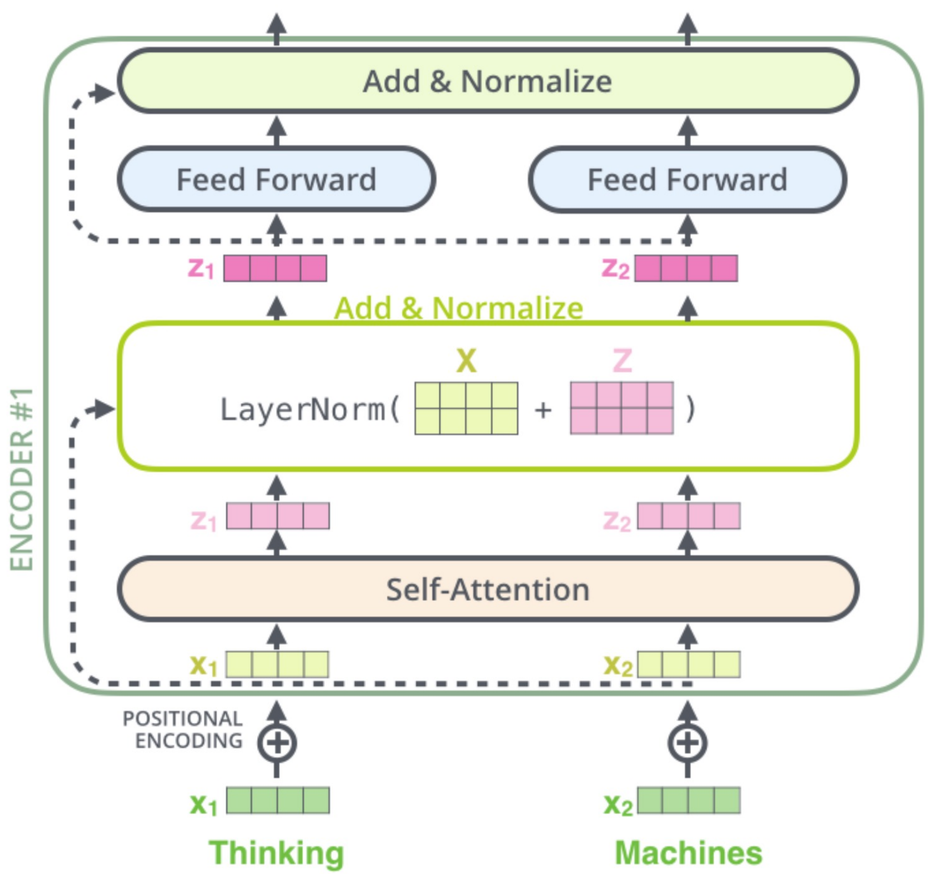




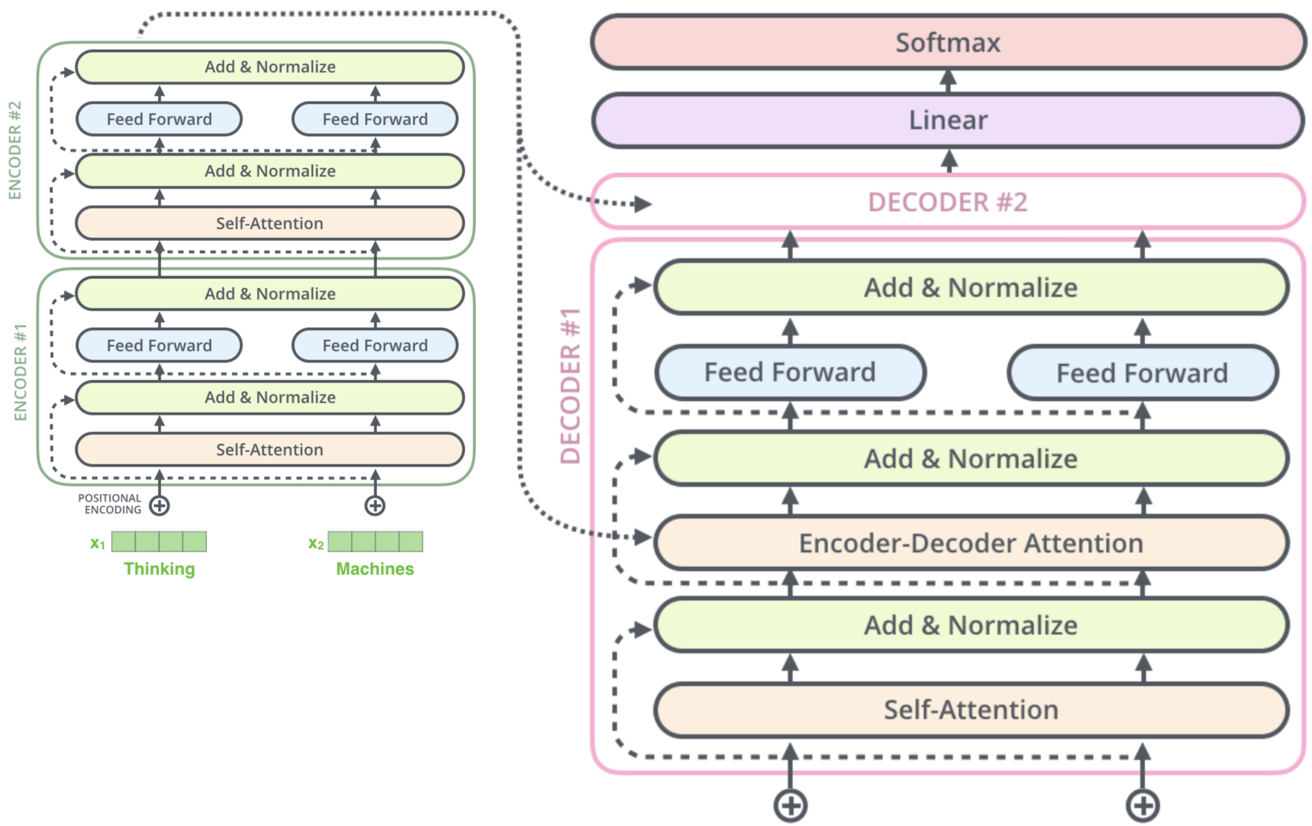
- **Encoder**
  - Self-attention is **invariant to order** of input sequence ⟶ **positional encoding**
  - **Residual connections** (dotted) and **layer normalization** are used to help training



- **Encoder**
  - Self-attention is **invariant to order** of input sequence ⟶ **positional encoding**
  - **Residual connections** (dotted) and **layer normalization** are used to help training
  - Non-linearity is imposed by adding position-wise **feed-forward networks**

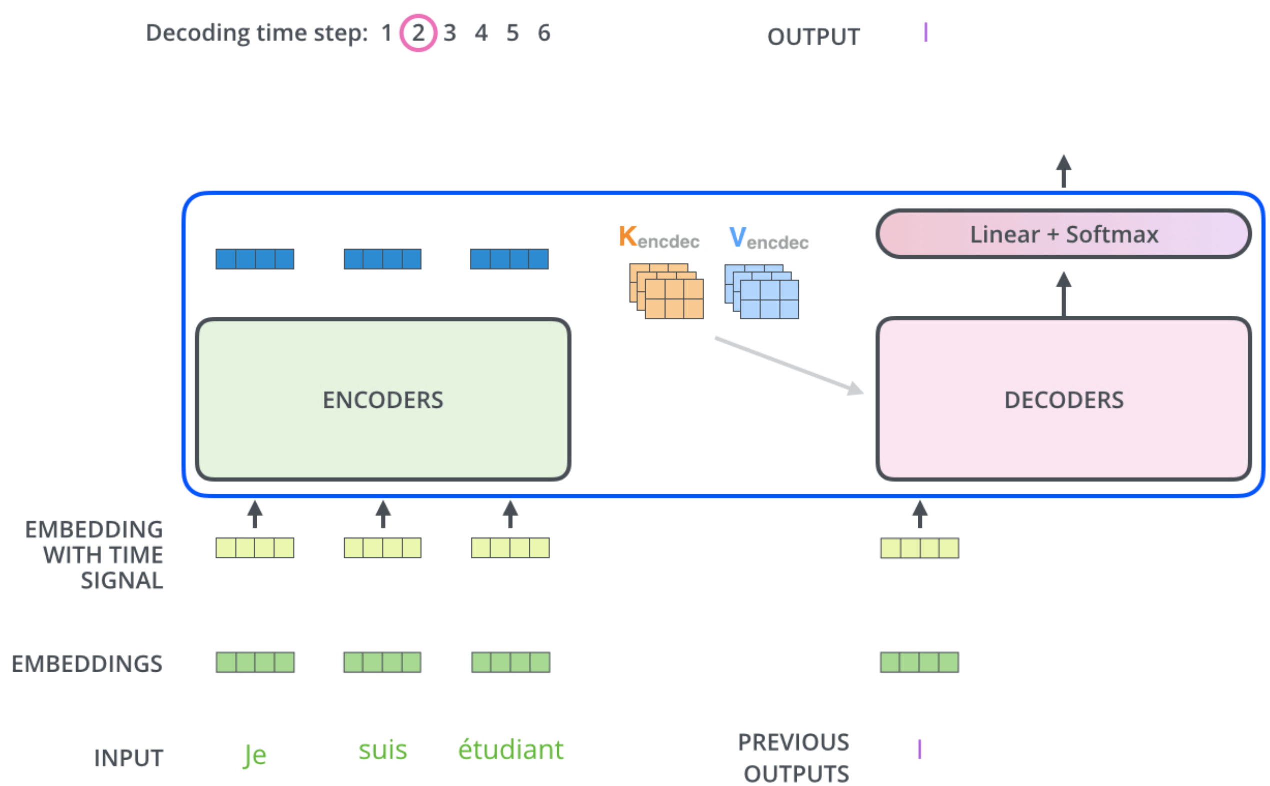


- **Decoder**
  - Most parts are same with encoder except **encoder-decoder(cross) attention**
  - This cross attention is previously used in seq2seq model
    - Queries are drawn from the **decoder**
    - Keys and values are drawn from the **encoder** (like context vector)



Transformer (Self-attention)

- **Decoder**
  - Most parts are same with encoder except **encoder-decoder(cross) attention**
  - This cross attention is previously used in seq2seq model
    - Queries are drawn from the **decoder**
    - Keys and values are drawn from the **encoder** (like context vector)



Transformer (Self-attention)

- Success of Transformer: **Machine Translation (MT)**
  - Initially, Transformer shows **better results at a fraction of the training cost**

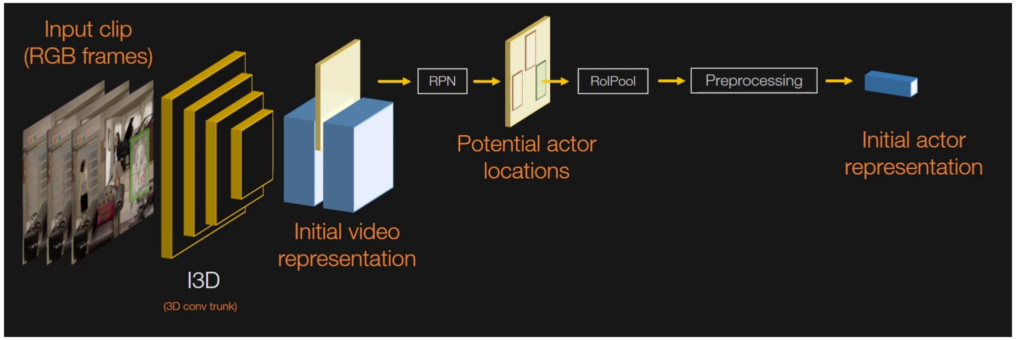
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

  - Nowadays, Transformer is still a standard for MT with additional techniques

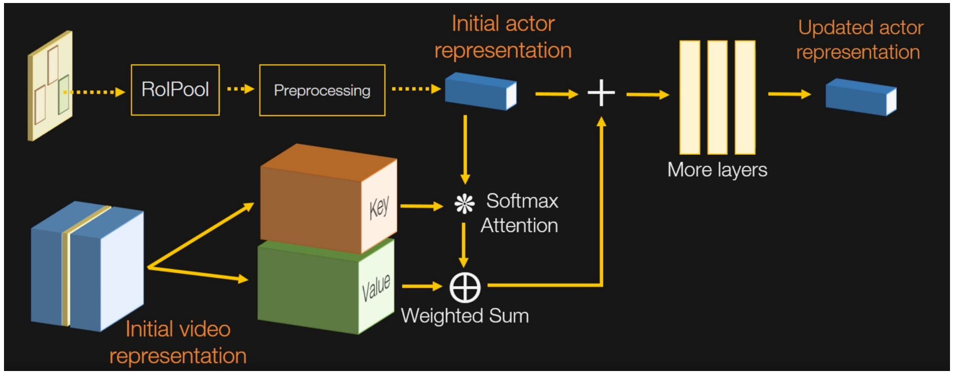
System	En→De	
	news2017	**news2018**
baseline	30.90	45.40
+ langid filtering	30.78	46.43
+ ffn 8192	31.15	46.28
+ BT	33.62	46.66
+ fine tuning	-	47.61
+ ensemble	-	49.27
+ reranking	-	50.63
WMT'18 submission	-	46.10
WMT'19 submission	**42.7**	

Transformer (Self-attention)

- Success of Transformer: **Video action recognition** [Girdhar et al., 2018]
  - **Goal**: localize the atomic action in space and time
  - Previous approaches just use the feature of key frame with object detection
    - But, it's hard to model the interaction between frames

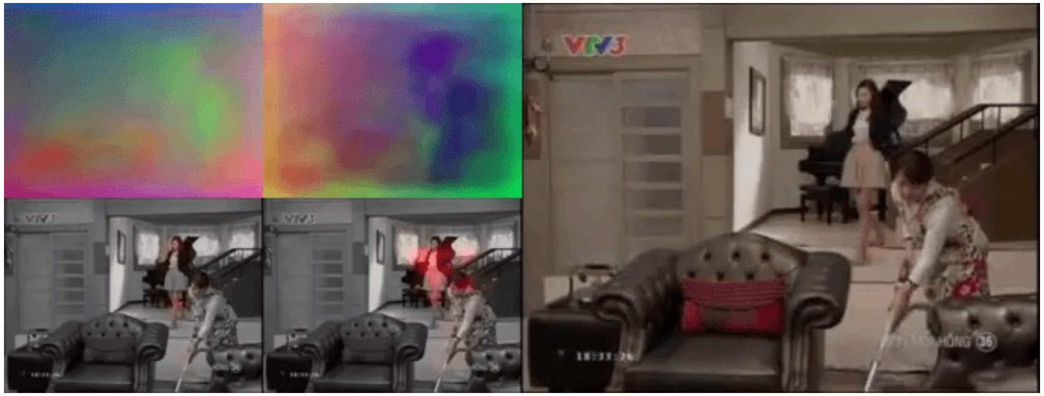


- **Self-attention is an effective way** to resolve this issue



Transformer (Self-attention)

- Success of Transformer: **Video action recognition** [Girdhar et al., 2018]
  - **Qualitative results of learned attention**



  - **Winner of AVA challenge in 2019: > 3.5 %** than previous challenge winner

Method	Modalities	Architecture	Val mAP	Test mAP
Single frame [16]	RGB, Flow	R-50, FRCNN	14.7	-
AVA baseline [16]	RGB, Flow	I3D, FRCNN, R-50	15.6	-
ARCN [42]	RGB, Flow	S3D-G, RN	17.4	-
Fudan University	-	-	-	17.16
YH Technologies [52]	RGB, Flow	P3D, FRCNN	-	19.60
Tsinghua/Megvii [23]	RGB, Flow	I3D, FRCNN, NL, TSN, C2D, P3D, C3D, FPN	-	21.08
Ours (Tx-only head)	RGB	I3D, Tx	24.4	24.30
Ours (Tx+I3D head)	RGB	I3D, Tx	24.9	24.60
Ours (Tx+I3D+96f)	RGB	I3D, Tx	**25.0**	**24.93**

Transformer (Self-attention)

- Success of Transformer: **Music generation** [Huang et al., 2018]
  - **Goal**: generate music which contains structure at multiple timescales (short to long)
  - Performance RNN (LSTM): lack of long-term structure

  - Music transformer; able to continue playing with consistent style

Pre-training and Fine-tuning Paradigm with Transformers

- **Motivation**
  - Many success of computer vision comes from ImageNet-pretrained networks
    - Simple fine-tuning improves the performance than training from scratch
  - **Q**. Then, can we train a similar universal pre-trained network for NLP tasks?
    - As labeling of NLP task is more ambiguous, **unsupervised pre-training is essential**
  - **Language modeling** is simple yet effective pre-training method without label
    - i.e., predicting what will be the next word
    - With diverse examples, model can learn the useful knowledge about the world

"Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was __." ⟶ *terrible*

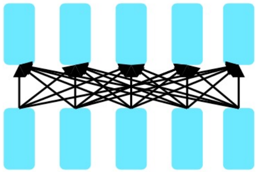
"I wat thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, __" ⟶ *34*

"I went to the ocean to see the fish, turtles, seals, and __" ⟶ *sand*

Pre-training and Fine-tuning Paradigm with Transformers

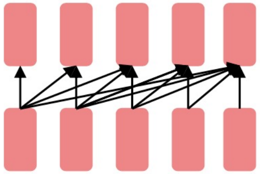
- **Motivation**
  - Many success of computer vision comes from ImageNet-pretrained networks
    - Simple fine-tuning improves the performance than training from scratch
  - **Q**. Then, can we train a similar universal pre-trained network for NLP tasks?
    - As labeling of NLP task is more ambiguous, **unsupervised pre-training is essential**
  - **Language modeling** is simple yet effective pre-training method without label
    - i.e., predicting what will be the next word
    - With diverse examples, model can learn the useful knowledge about the world

- **Pre-training for two types of architectures**
  - Architecture influences the type of pre-training, and specific use cases



Encoders

- E.g. **BERT**
- Pre-training with **masked** language modeling
- Better use for **discriminative** tasks (classification)

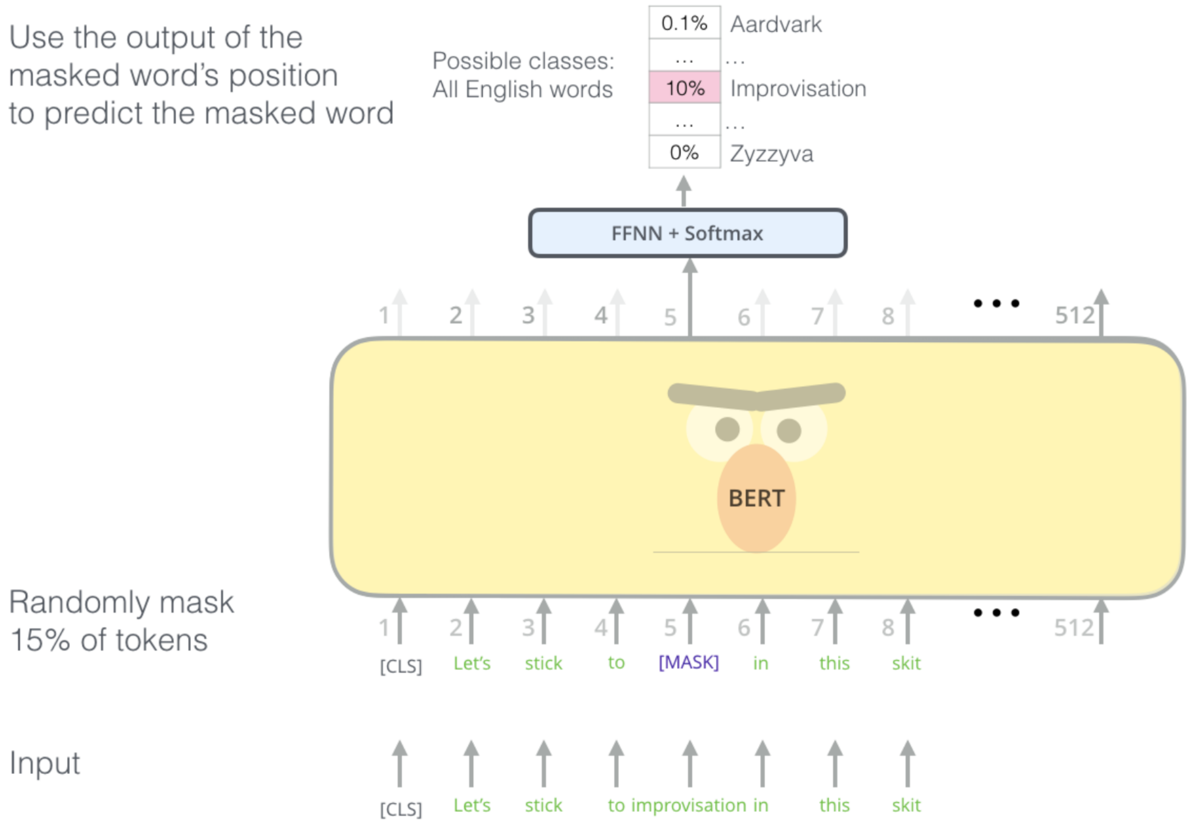


Decoders

- E.g. **GPT**
- Pre-training with **normal** language modeling
- Better use for **generation** tasks

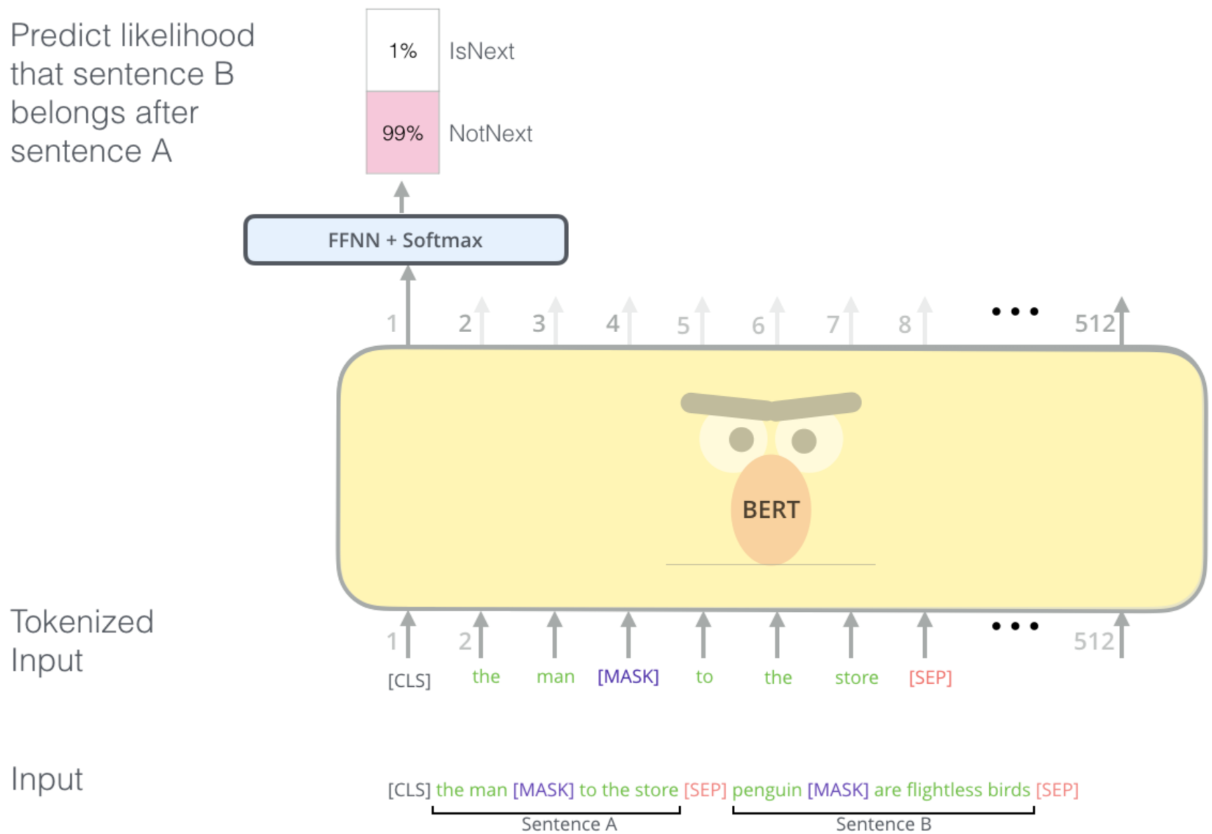
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

- **BERT**: Bidirectional Encoder Representations from Transformers [Devlin et al., 2018]
  - As **encoders get bidirectional context**, original language modeling is suboptimal
    - Not only left-to-right, but also right-to-left modeling is possible
  - Hence, **masked language modeling** is used for pre-training
    - Replace some fraction of words (15%) in the input, then predict these words

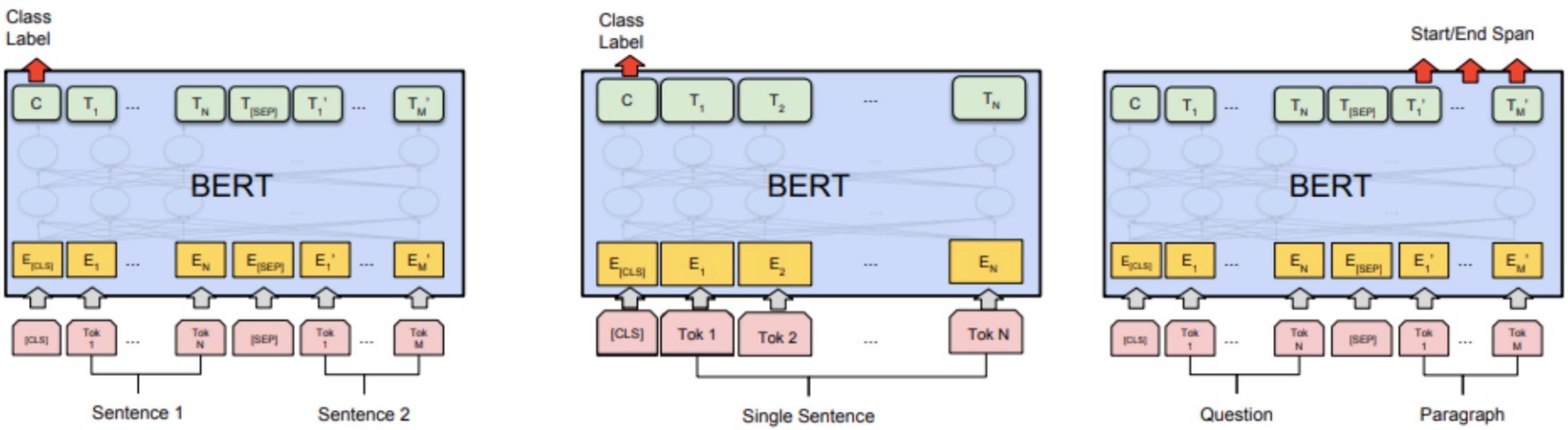


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

- **BERT**: Bidirectional Encoder Representations from Transformers [Devlin et al., 2018]
  - As **encoders get bidirectional context**, original language modeling is suboptimal
  - Hence, **masked language modeling** is used for pre-training
  - Additionally, **next sentence prediction** (NSP) task is used for pre-training
    - Decide whether two input sentences are **consecutive or not**



- **BERT**: Bidirectional Encoder Representations from Transformers [Devlin et al., 2018]
  - Even **without** task-specific complex architectures, BERT achieves **SOTA** for **11 NLP tasks**, including classification, question answering, tagging, etc.
    - By simply fine-tuning a whole network with **additional linear classifier**



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

- **BERT**: Bidirectional Encoder Representations from Transformers [Devlin et al., 2018]
  - Even **without** task-specific complex architectures, BERT achieves **SOTA** for **11 NLP tasks**, including classification, question answering, tagging, etc.
    - By simply fine-tuning a whole network with **additional linear classifier**

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

RoBERTa: A Robustly Optimized BERT Pre-training Approach

- **RoBERTa** [Liu et al., 2019]
  - Simply modifying BERT design choices and training strategies with alternatives
    - Using **dynamic masking** instead of static masking in BERT
    - **Removing NSP task** and generate training data in single document instead
    - Much **larger data** for pre-training: 16GB ⟶ 160GB, and etc…
  - But, it leads a huge improvement in many downstream tasks

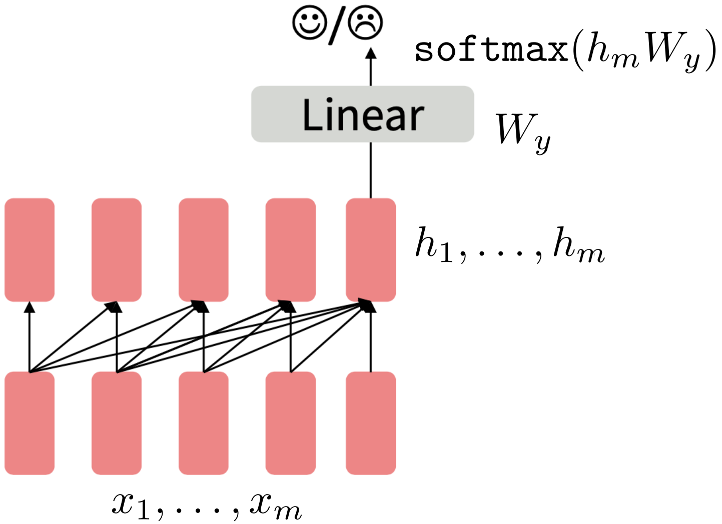
Model	**data**	**bsz**	**steps**	**SQuAD** (v1.1/2.0)	**MNLI-m**	**SST-2**
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

GPT: Generative Pre-Training with Transformer's Decoder

- **GPT** [Radford et al., 2018]

$$\arg\max_{\theta} \ \log p(\boldsymbol{x}) = \sum_{n} p_\theta(x_n | x_1, \ldots, x_{n-1})$$

  - **Pre-training** by language modeling over 7000 unique books (**unlabeled data**)
    - Contains long spans of contiguous text, for learning long-distance dependencies
  - **Fine-tuning** by training a classifier with target task-specific **labeled data**
    - Classifier is added on the final transformer block's last word's hidden state



- **GPT** [Radford et al., 2018]

$$\arg\max_{\theta} \log p(\boldsymbol{x}) = \sum_{n} p_\theta(x_n | x_1, \ldots, x_{n-1})$$

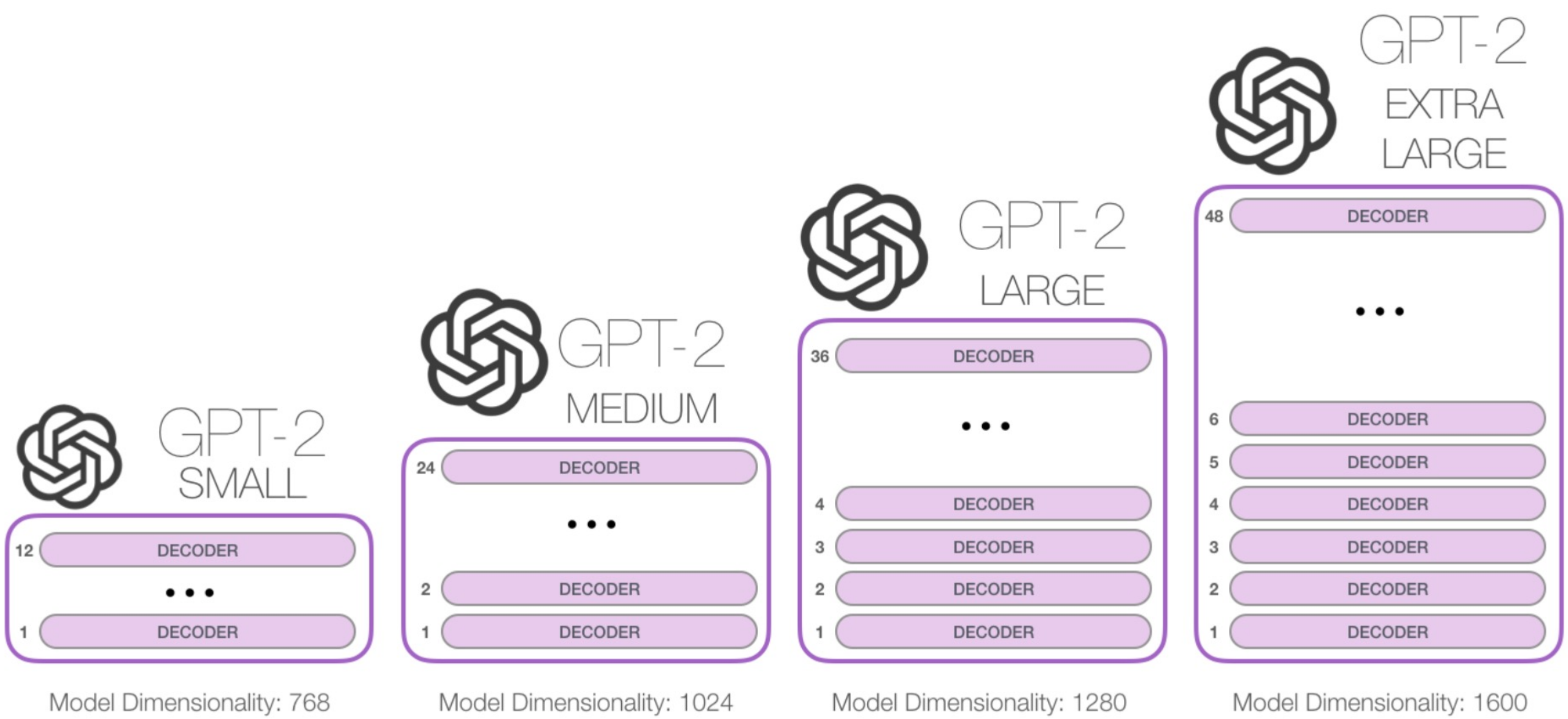
  - **Pre-training** by language modeling over 7000 unique books (**unlabeled data**)
    - Contains long spans of contiguous text, for learning long-distance dependencies
  - **Fine-tuning** by training a classifier with target task-specific **labeled data**
    - Classifier is added on the final transformer block's last word's hidden state

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

GPT's results on various *natural language inference* datasets

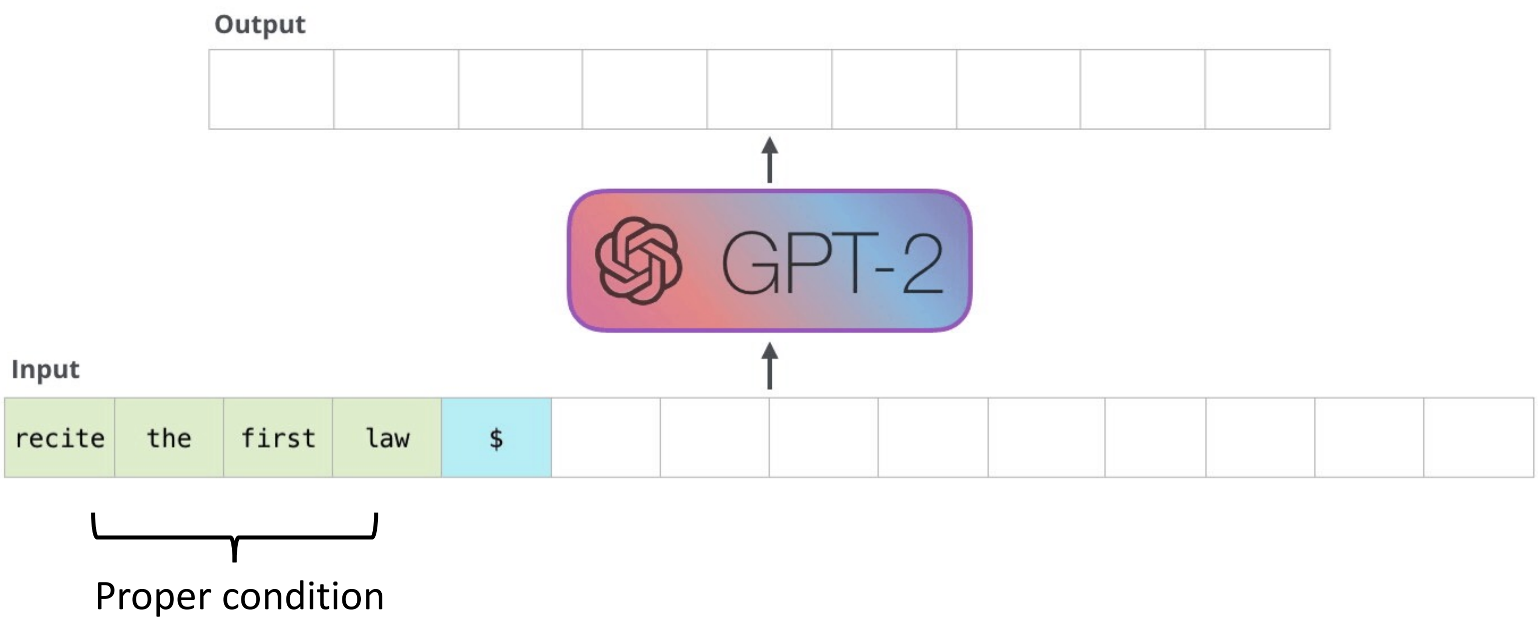
GPT-2: Language Models are Unsupervised Multitask Learners

- **GPT-2** [Radford et al., 2019]
  - **Pre-training** by language modeling as same as previous GPT-1, but **training with..**
    - Much **larger datasets**; 8 million documents from web (40 GB of text)
    - Much **larger model size**; # of parameters: 117M (GPT-1) ⟶ 1542M (extra-large GPT-2)



GPT-2: Language Models are Unsupervised Multitask Learners

- **GPT-2** [Radford et al., 2019]
  - **Pre-training** by language modeling as same as previous GPT-1, but **training with..**
    - Much **larger datasets**; 8 million documents from web (40 GB of text)
    - Much **larger model size**; # of parameters: 117M (GPT-1) ⟶ 1542M (extra-large GPT-2)
  - GPT-2 can perform down-stream tasks in a **zero-shot setting**
    - Via conditional generation without any parameter or architecture modification



GPT-2: Language Models are Unsupervised Multitask Learners

- **GPT-2** [Radford et al., 2019]
  - **Pre-training** by language modeling as same as previous GPT-1, but **training with..**
    - Much **larger datasets**; 8 million documents from web (40 GB of text)
    - Much **larger model size**; # of parameters: 117M (GPT-1) ⟶ 1542M (extra-large GPT-2)
  - GPT-2 can perform down-stream tasks in a **zero-shot setting**
    - Via conditional generation without any parameter or architecture modification
  - **Remark**. Largest model still underfits.. ⟶ larger model for better performance?

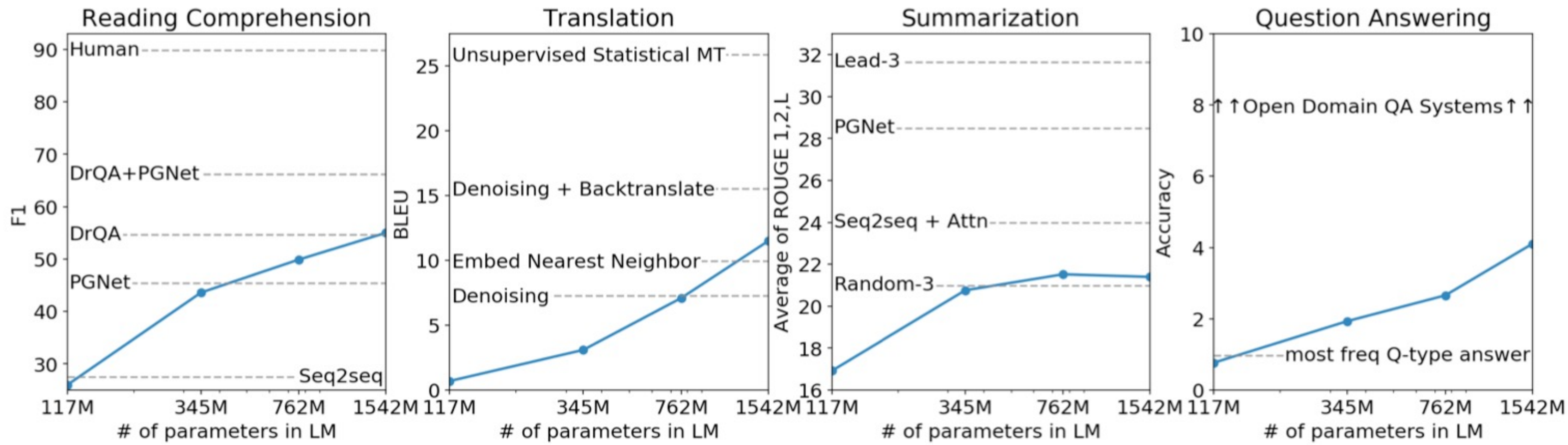
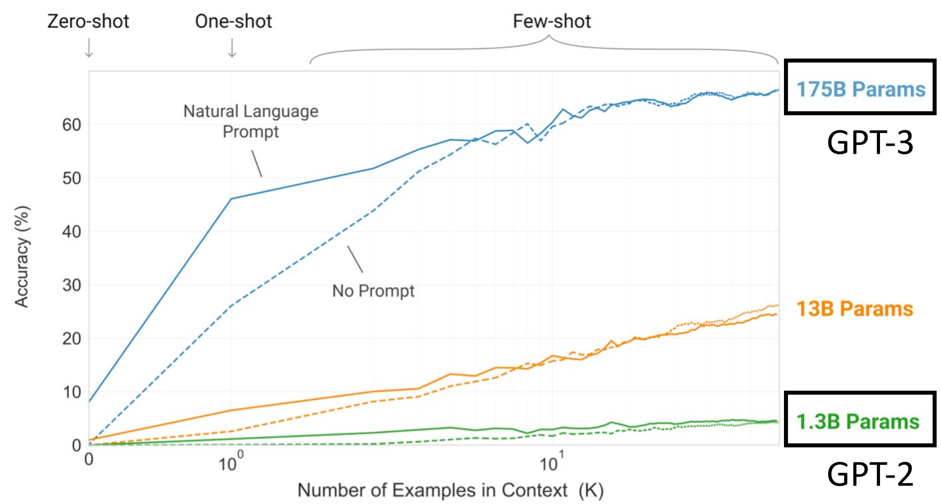


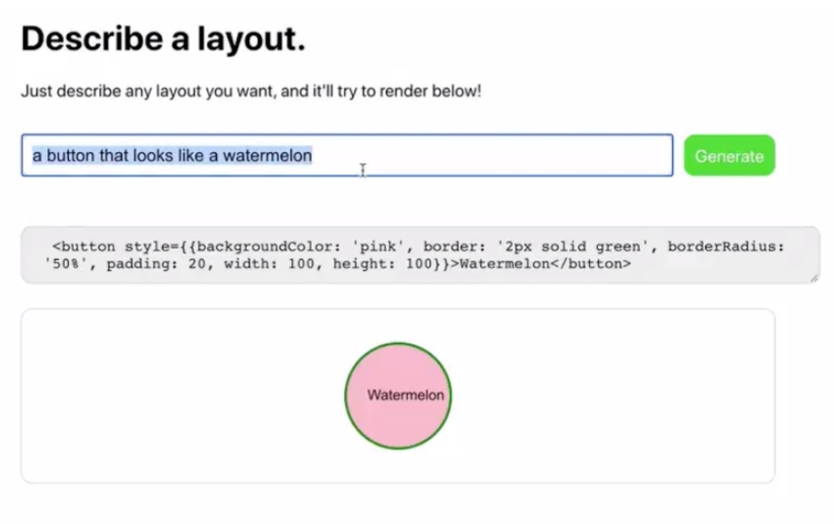
Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

- **GPT-3**: Language Models are Few-shot Learners [Brown et al., 2020]
  - **First very large** language models (1B → 175B parameters)
  - With this scale-up, new capability of LMs suddenly emerges
  - E.g., it can adapt to new tasks **perform in-context learning** **without fine-tuning**
    - **In-context learning (prompting)**; adapting to task from examples with some context

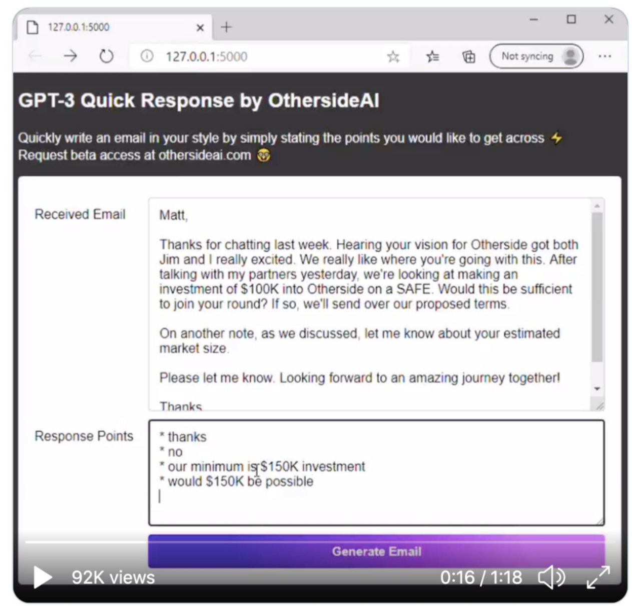


Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

- **GPT-3**: Language Models are Few-shot Learners [Brown et al., 2020]
  - **First very large** language models (**LLMs**, 1B → 175B parameters)
  - With this scale-up, new capability of LMs suddenly emerges
  - E.g., it can adapt to new tasks **perform in-context learning** **without fine-tuning**
  - It enables us to do a lot of interesting applications!
  - E.g.,



Simple code generation



Email response

Overview

Part 1. Basics

- RNN to LSTM
- Sequence-to-sequence Model
- Attention-based NLP Model

Part 2. Advanced Topics

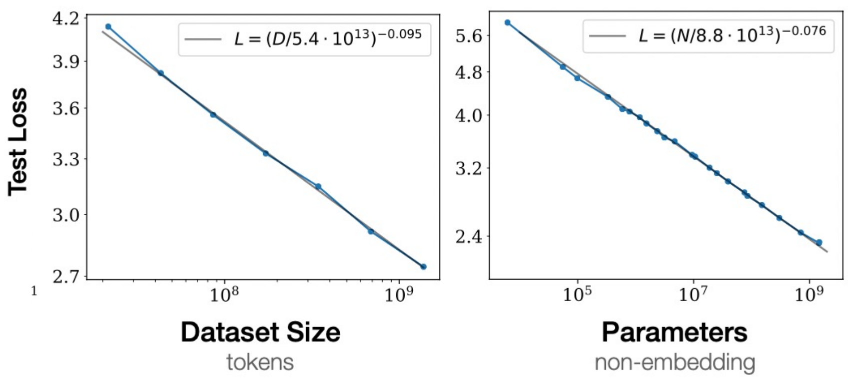
- Transformer (self-attention)
- Pre-training of Transformers and Language Models
- Large Language Models: GPT-3 and emerging properties

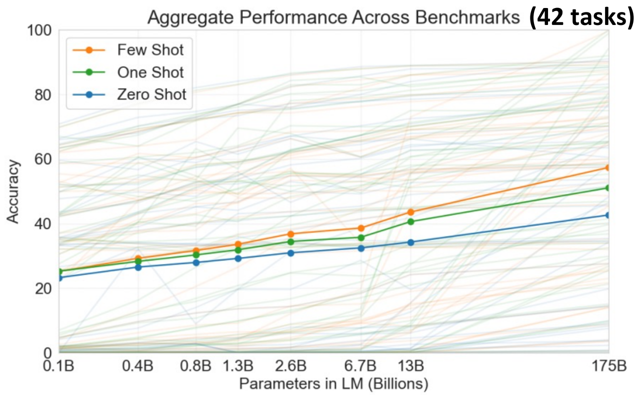
Part 3. Recent Advances in Large Language Models

- Large language models beyond GPT-3
- Better training schemes for large language model
- Applications: ChatBot (e.g., ChatGPT)

Beyond GPT-3

- GPT-3 reveals the **effectiveness of large-scale language models and datasets**
  - Performance improves as the size of model and dataset increase [Kaplan et al., 2020]
  - Few-shot adaptability to new task (in-context learning) is also significantly improved





- Success of GPT-3 opens up the following research questions:
  1. Can we develop better LLMs via **scaling up**?
  2. What is a **better training scheme** for these LLMs than language modeling?
  3. Are these LLMs really useful for **real world application**?

- **Gopher** [Rae et al., 2022]
  - **280 billion parameters**: 80 Transformer layers with 16,384 hidden dimensions
  - **Model modification**: (1) RMSNorm and (2) relative positional encoding
    - RMSNorm [Zhang et al., 2019] removes unnecessary scaling term in LayerNorm

LayerNorm: $$\bar{a}_i = \frac{a_i - \mu}{\sigma} g_i \qquad \mu = \frac{1}{n}\sum_{i=1}^{n} a_i \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \mu)^2}$$

RMSNorm: $$\bar{a}_i = \frac{a_i}{\mathbf{RMS}(\mathbf{a})} g_i \qquad \mathbf{RMS}(\mathbf{a}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} a_i^2}$$

- Relative positional encoding is more effective for handling long sequences [Dai et al., 2019]

Model	$r = 0.1$	$r = 0.5$	$r = 1.0$
Transformer-XL 151M	**900**	**800**	**700**
QRNN	500	400	300
LSTM	400	300	200
Transformer-XL 128M	**700**	**600**	**500**
- use Shaw et al. (2018) encoding	400	400	300
- remove recurrence	300	300	300
Transformer	128	128	128

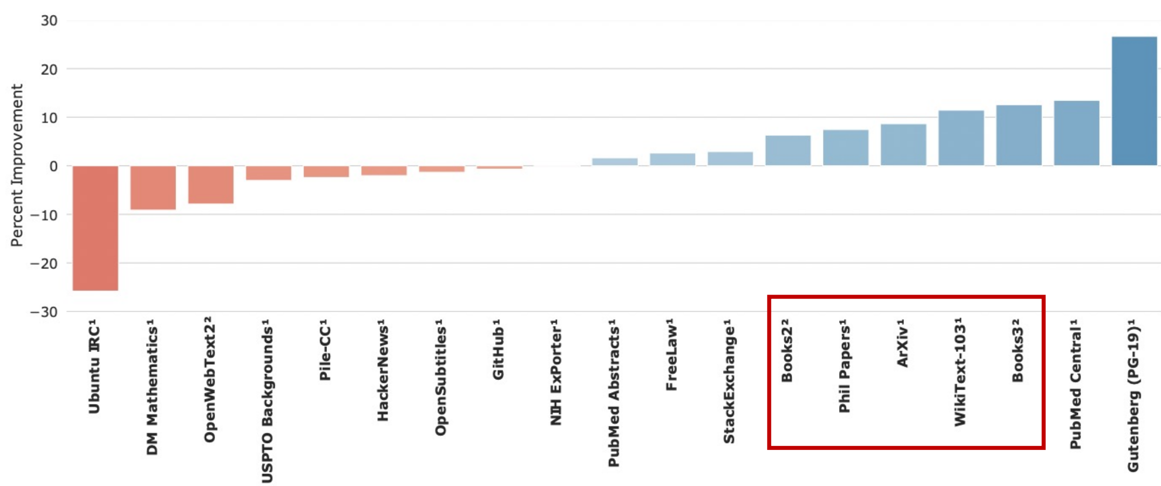
Relative Effective Context Length

Language Models beyond GPT-3 : Gopher

- **Gopher** [Rae et al., 2022]
  - Pre-training on new large text dataset: **MassiveText**
    - Number of tokens in datasets: **2350 B (Gopher)** vs 333.7 B (MT-NLG)

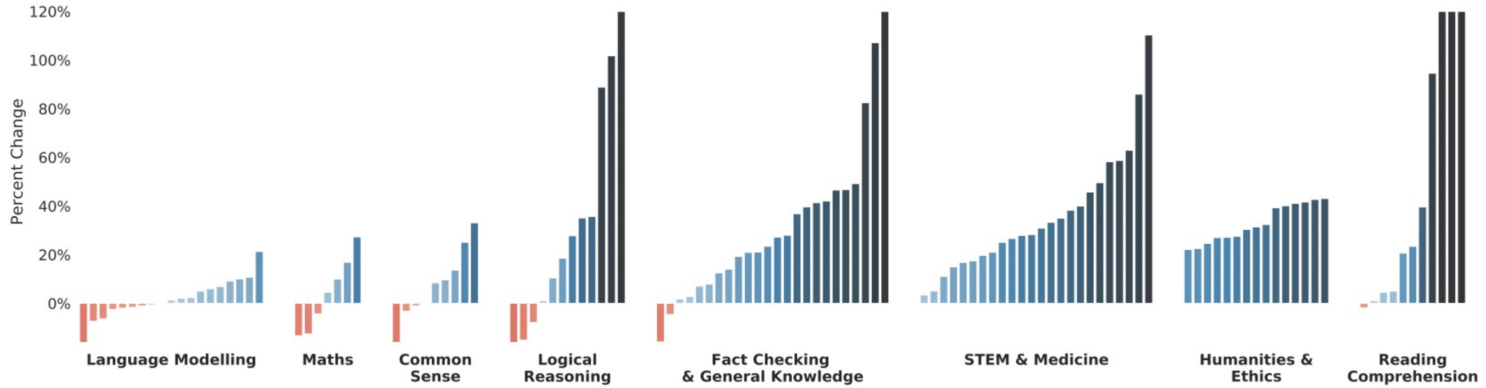
	Disk Size	Documents	Tokens	Sampling proportion
MassiveWeb	1.9 TB	604M	506B	48%
Books	2.1 TB	4M	560B	27%
C4	0.75 TB	361M	182B	10%
News	2.7 TB	1.1B	676B	10%
GitHub	3.1 TB	142M	422B	3%
Wikipedia	0.001 TB	6M	4B	2%

    - Sampling portion affect to performance → Gopher is much effective on Books like tasks



Language Models beyond GPT-3 : Gopher

- **Gopher** [Rae et al., 2022]
  - Pre-training on new large text dataset: **MassiveText**
  - Overall, **Gopher outperforms** the existing SOTA LMs
    - Performance improvement compared to the best among {GPT-3, Jurrasic-1, MT-NLG}
    - Gopher improves the performance across 100 / 124 tasks

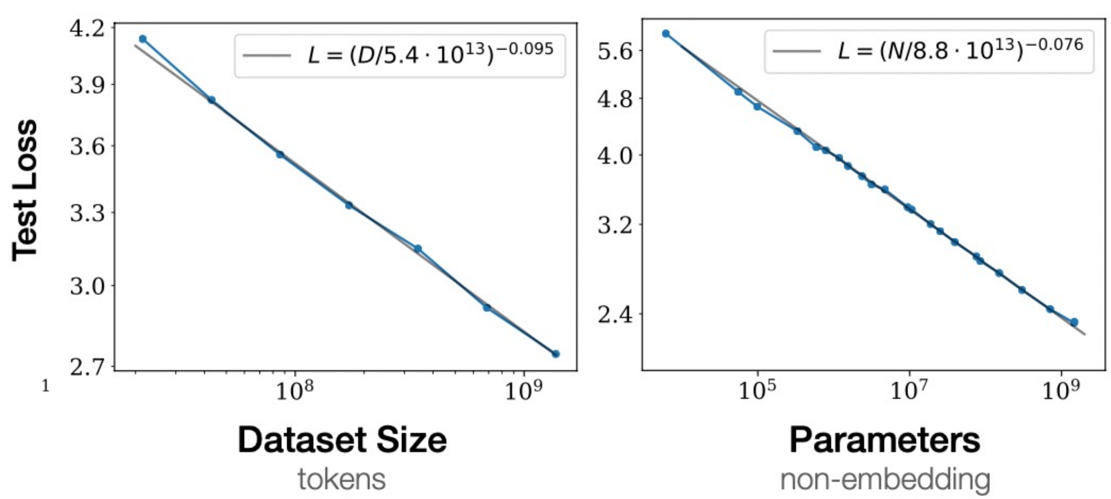


	417M	1.4B	7.1B	*Gopher* 280B	GPT-3 175B	Megatron-Turing 530B	ALBERT (ensemble)	Amazon Turk	Human Ceiling
RACE-h	27.2	26.0	30.6	**71.6**	46.8	47.9	<u>90.5</u>	69.4	94.2
RACE-m	26.2	25.0	31.8	**75.1**	58.1	n/a	<u>93.6</u>	85.1	95.4

Results on reading comprehension tasks

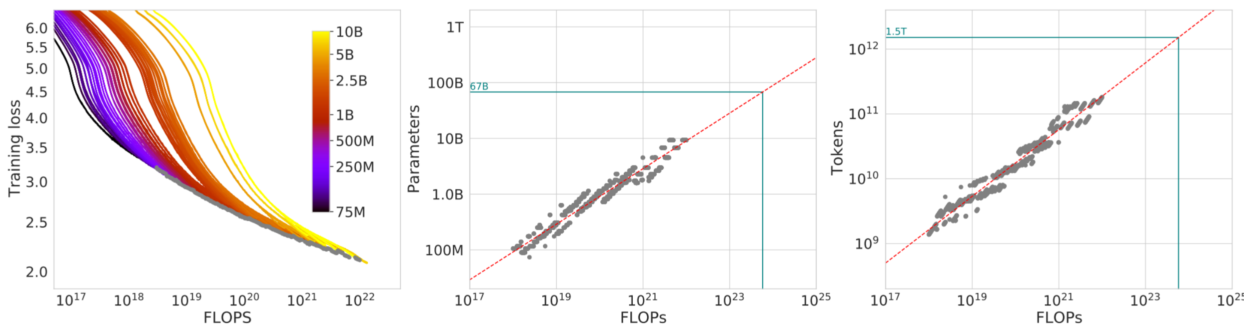
Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]
  - **Motivation:** current large language models are **significantly undertrained**
    - Due to recent focus on scaling LMs whilst keeping the amount of training data constant
      → But, performance also critically depends on **number of trained tokens** [Kaplan et al., 2020]
    - **Q**. Given a FLOPs budget, *how should one trade-off model size and the number of tokens*?



Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]
  - **Motivation:** current large language models are **significantly undertrained**
  - Multiple approaches reveal **new optimal parameter/training tokens trade-off**
    - *Approach 1.* Fix model sizes and vary number of training tokens

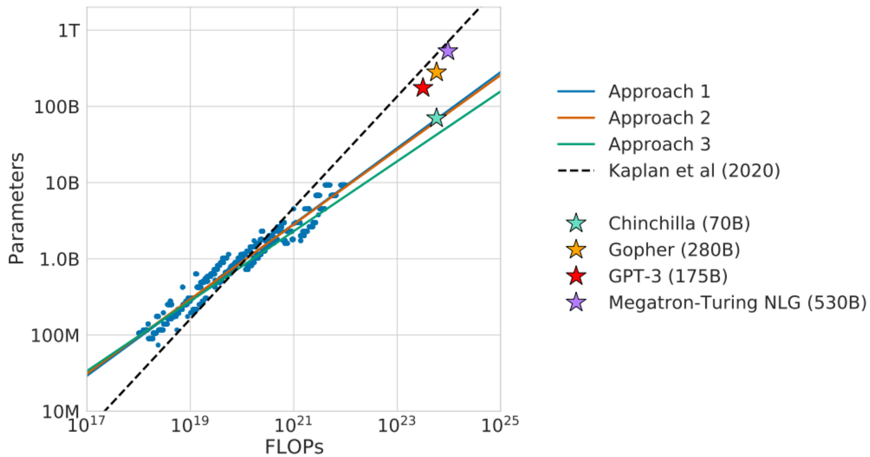


    - *Approach 2.* IsoFLOP profiles (i.e., same FLOP by varying the trade-off)
    - *Approach 3.* Fitting a parametric loss function (with multiple models on different trade-off)

Approach	Coeff. a where $N_{opt} \propto C^a$	Coeff. b where $D_{opt} \propto C^b$
1. Minimum over training curves	0.50 (0.488, 0.502)	0.50 (0.501, 0.512)
2. IsoFLOP profiles	0.49 (0.462, 0.534)	0.51 (0.483, 0.529)
3. Parametric modelling of the loss	0.46 (0.454, 0.455)	0.54 (0.542, 0.543)
Kaplan et al. (2020)	0.73	0.27

Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]
  - **Motivation:** current large language models are **significantly undertrained**
  - Multiple approaches reveal **new optimal parameter/training tokens trade-off**
    - Previous LLMs follow the previous optimal trade-off
    - Chinchilla follows new optimal by reducing the model size while increasing training tokens (to keep **same total FLOPs**)



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

Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

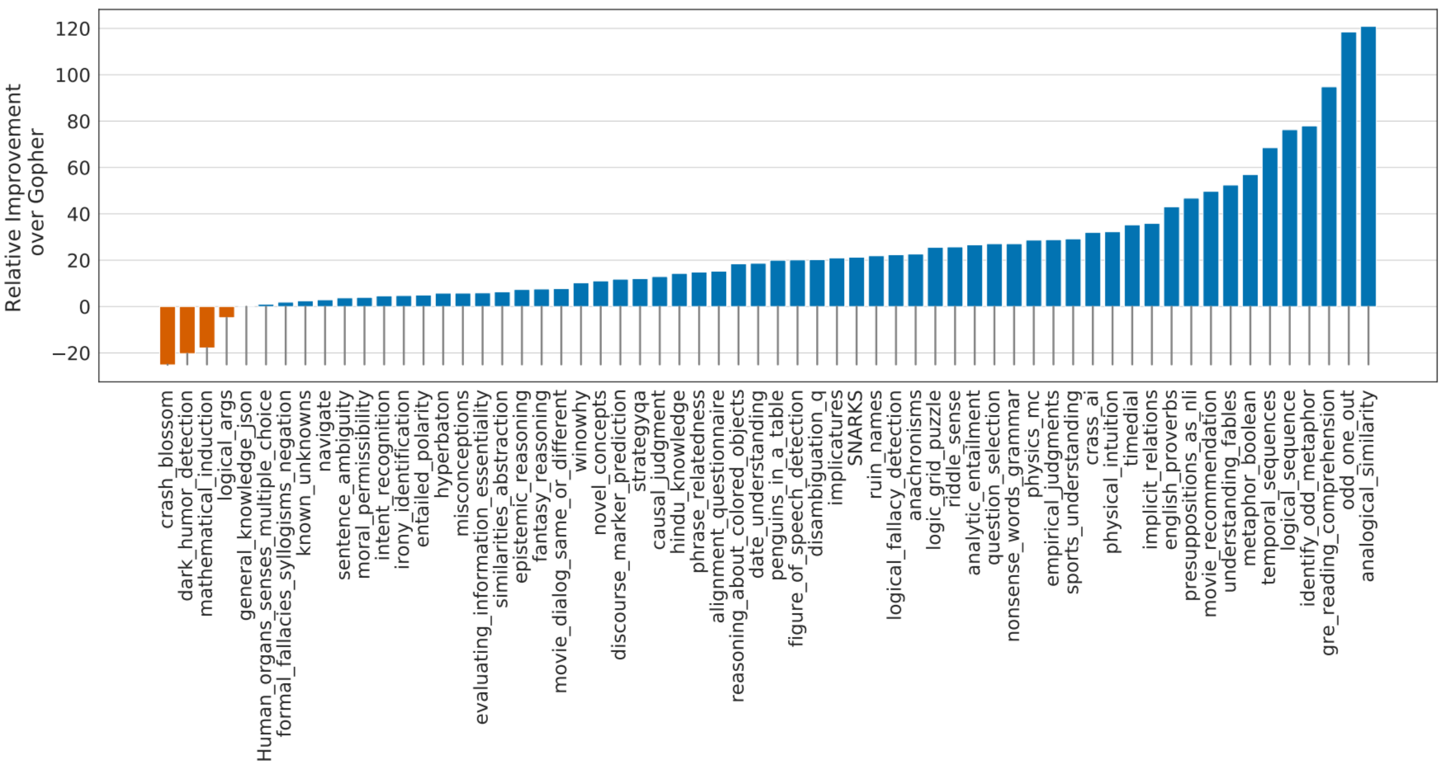
Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]
  - Chinchilla significantly outperforms the previous LLMs
  - **Results on MMLU** [Hendrycks et al., 2020] (Massive Multitask Language Understanding)
    - MMLU consists of **57** different tasks
    - **7.6%** average improvement → (vs Gopher) 51 wins, 2 ties, 4 loses on 57 tasks

Random	25.0%
Average human rater	34.5%
GPT-3 5-shot	43.9%
Gopher 5-shot	60.0%
***Chinchilla* 5-shot**	**67.6%**
Average human expert performance	*89.8%*
June 2022 Forecast	57.1%
June 2023 Forecast	63.4%

Language Models beyond GPT-3 : Chinchilla

- **Chinchilla** [Hoffmann et al., 2022]
  - Chinchilla significantly outperforms the previous LLMs
  - **Results on BIG-bench** [Rae et al., 2021]
    - BIG-bench consists of **62** different tasks
    - **10.7%** average improvement → (vs Gopher) 57 wins, 1tie, 4 loses on 62 tasks



Language Models beyond GPT-3 : PaLM

- **PaLM** (<u>**Pa**thways</u> **L**anguage **M**odel) [Chowdhery et al., 2022]
  - <u>Pathways</u>: Distributed learning system of google with TPU [Barham et al., 2022]
    - Make it possible to efficiently train tremendous parameters with many TPUs (6144 TPUs)
  - **540B parameters (largest)**: 118 Transformer layers with 18,432 hidden dimensions
    - Largest Transformer-based language model in the world

Model	# of Parameters (in billions)	Accelerator chips	Model FLOPS utilization
GPT-3	175B	V100	21.3%
Gopher	280B	4096 TPU v3	32.5%
Megatron-Turing NLG	530B	2240 A100	30.2%
PaLM	540B	6144 TPU v4	46.2%

  - **780B training tokens**: smaller than Chinchilla, but 4x larger FLOPs in total

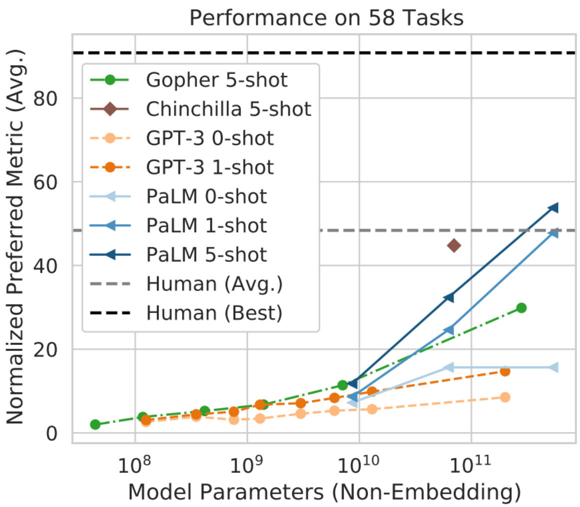
Total dataset size = 780 billion tokens	
Data source	Proportion of data
Social media conversations (multilingual)	50%
Filtered webpages (multilingual)	27%
Books (English)	13%
GitHub (code)	5%
Wikipedia (multilingual)	4%
News (English)	1%

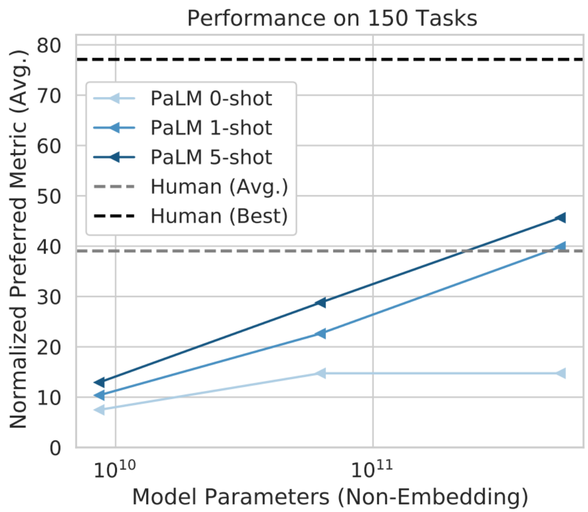
Language Models beyond GPT-3 : PaLM

- **PaLM** (**Pa**<u>thways</u> **L**anguage **M**odel) [Chowdhery et al., 2022]
  - PaLM shows the better performance compared to previous LLMs
    - Hence, it is now used as a standard in google (e.g., PaLM is backbone of BARD)
  - Results on MMLU

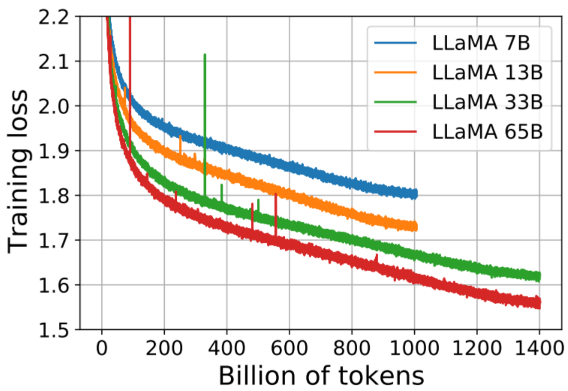
Model	Average	Humanities	STEM	Social Sciences	Other
Chinchilla 70B (Prior SOTA)	67.5	63.6	54.9	79.3	**73.9**
PaLM 8B	25.3	25.6	23.8	24.1	27.8
PaLM 62B	53.7	59.5	41.9	62.7	55.8
PaLM 540B	**69.3**	**77.0**	**55.6**	**81.0**	69.6

  - Results on BIG-Bench





- **LLaMA** (**L**arge **La**nguage model **M**eta **A**I) [Touvron et al., 2023]
  - Recently released LLMs by MetaAI under similar approach with Chinchilla
    - Namely, smaller model sizes (7B to 65B) with larger training tokens (1.4T)
    - With some architectural modification based on previous works (from GPT-3, PaLM)
    - But, different to previous LLMs, LLaMA is built on publicly available data only (**open-source**)



Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

Language Models beyond GPT-3 : LLaMA

- **LLaMA** (**L**arge **La**nguage model **M**eta **A**I) [Touvron et al., 2023]
  - Recently released LLMs by MetaAI under similar approach with Chinchilla
    - Namely, smaller model sizes (7B to 65B) with larger training tokens (1.4T)
    - With some architectural modification based on previous works (from GPT-3, PaLM)
    - But, different to previous LLMs, LLaMA is built on publicly available data only (**open-source**)
  - Comparable performance to Chinchilla
    - Better performance on 1) zero-shot common sense reasoning and 2) question & answering

		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

Table 3: **Zero-shot performance on Common Sense Reasoning tasks.**

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
PaLM	8B	8.4	10.6	-	14.6
	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
LLaMA	7B	16.8	18.7	22.0	26.1
	13B	20.1	23.4	28.1	31.9
	33B	**24.9**	28.3	32.9	36.0
	65B	23.8	**31.0**	**35.0**	**39.9**

Table 4: **NaturalQuestions.** Exact match performance.

Language Models beyond GPT-3 : LLaMA

- **LLaMA** (**L**arge **La**nguage model **M**eta **A**I) [Touvron et al., 2023]
  - Recently released LLMs by MetaAI under similar approach with Chinchilla
    - Namely, smaller model sizes (7B to 65B) with larger training tokens (1.4T)
    - With some architectural modification based on previous works (from GPT-3, PaLM)
    - But, different to previous LLMs, LLaMA is built on publicly available data only (**open-source**)
  - Comparable performance to Chinchilla
    - Better performance on 1) zero-shot common sense reasoning and 2) question & answering
    - Worse performance on popular benchmark in LLMs (MMLU)

		Humanities	STEM	Social Sciences	Other	Average
GPT-NeoX	20B	29.8	34.9	33.7	37.7	33.6
GPT-3	175B	40.8	36.7	50.4	48.8	43.9
Gopher	280B	56.2	47.4	71.9	66.1	60.0
Chinchilla	70B	63.6	54.9	79.3	**73.9**	67.5
PaLM	8B	25.6	23.8	24.1	27.8	25.4
	62B	59.5	41.9	62.7	55.8	53.7
	540B	**77.0**	**55.6**	**81.0**	69.6	**69.3**
LLaMA	7B	34.0	30.5	38.3	38.1	35.1
	13B	45.0	35.8	53.8	53.3	46.9
	33B	55.8	46.0	66.7	63.4	57.8
	65B	61.8	51.7	72.9	67.4	63.4

Table 9: **Massive Multitask Language Understanding (MMLU).** Five-shot accuracy.

Better Training Scheme for Large Language Models

- Although language modeling is an effective training scheme with unlabeled text data, there are **remained limitations**

$$\arg\max_{\theta} \log p(\boldsymbol{x}) = \sum_{n} p_{\theta}(x_n | x_1, \ldots, x_{n-1})$$

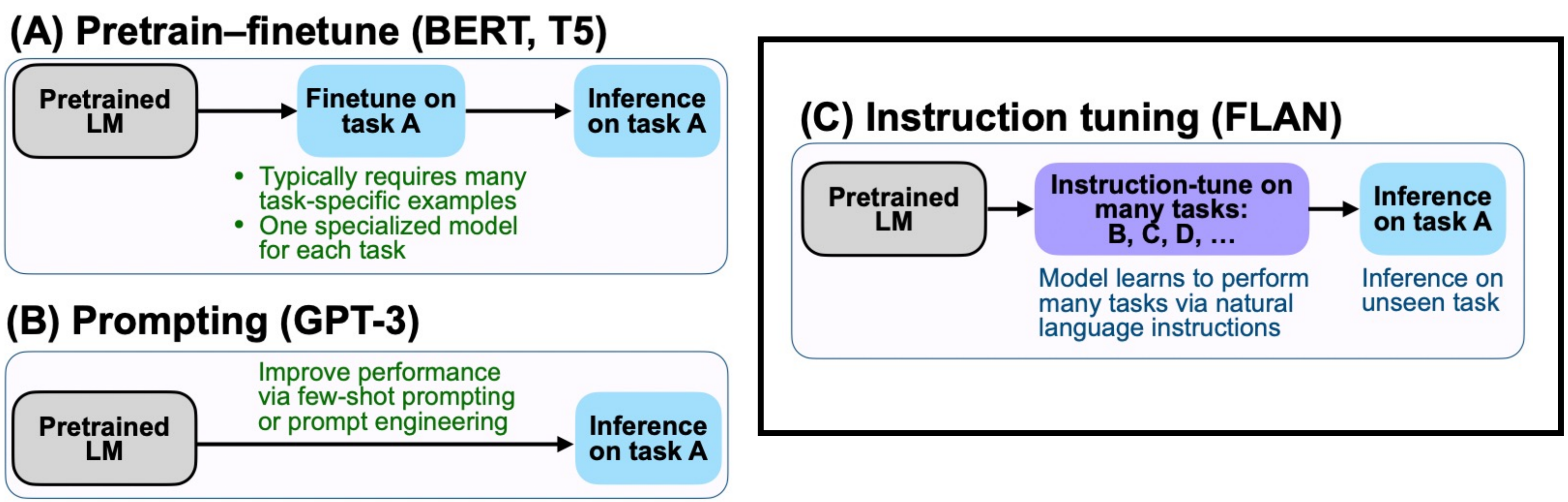
  - Zero-shot performance is **much worsen** that Few-shot performance
  - Multi-task generalization via LM is **indirectly obtained** → Suboptimality

Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

Results on three open-domain QA tasks [Brown et al., 2020]

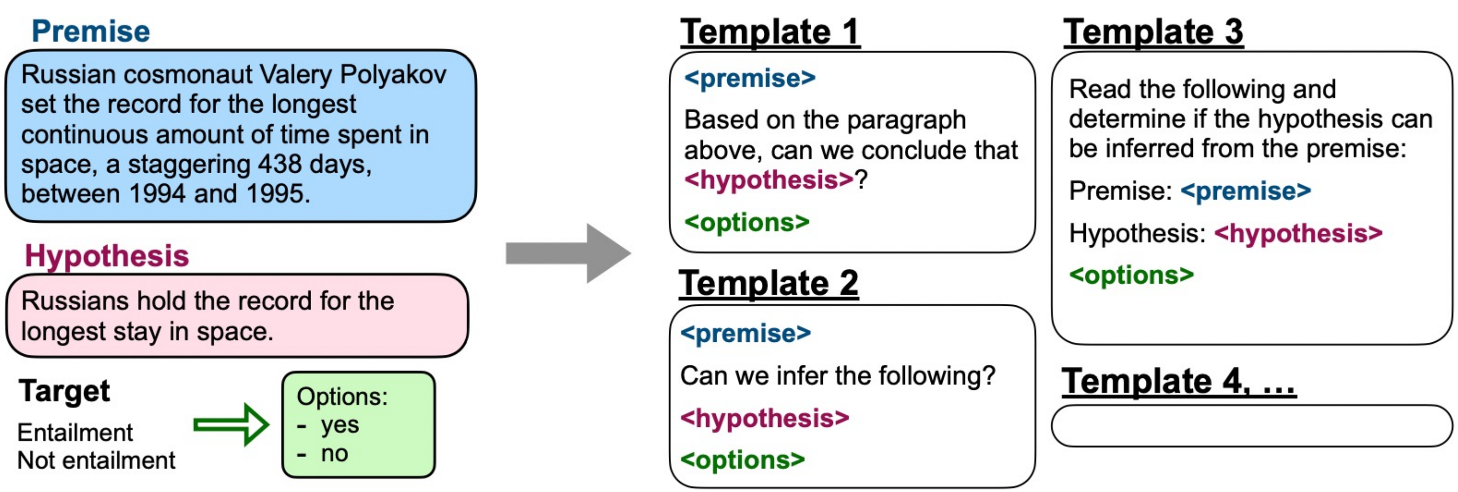
Better Training Scheme for Large Language Models: FLAN

- **FLAN** [Wei et al., 2022]
  - **Intuition**: NLP tasks can be described via **natural language instructions**
    - E.g., *"Is the sentiment of this movie review positive or negative?"*
    - It offers a natural and intuitive way for adapting LM to any task
  - **Method**: fine-tuning LMs (e.g., GPT-3) with **instructions** instead of prompts
    - Remark. Very similar approach is also proposed by other group: **T0** [Sanh et al., 2022]



Better Training Scheme for Large Language Models: FLAN

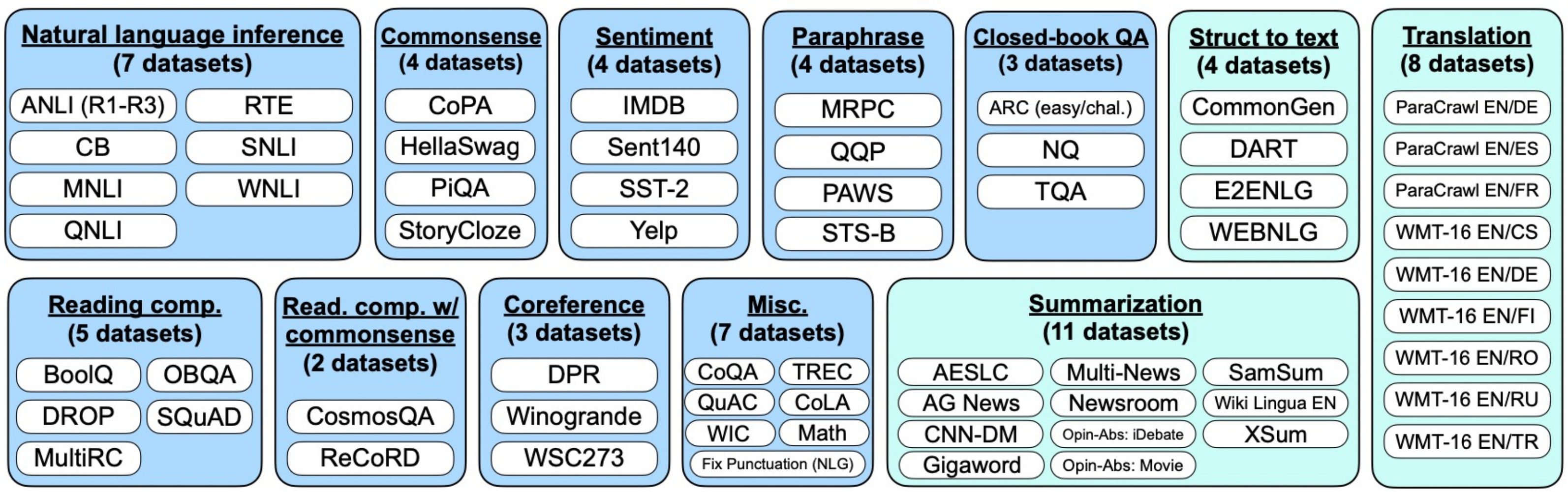
- **FLAN** [Wei et al., 2022]
  - **Intuition**: NLP tasks can be described via **natural language instructions**
    - E.g., *"Is the sentiment of this movie review positive or negative?"*
    - It offers a natural and intuitive way for adapting LM to any task
  - **Method**: fine-tuning LMs (e.g., GPT-3) with **instructions** instead of prompts
    - To increase the diversity, **multiple instructions** are constructed for each task
    - Model output is given as text → each class is mapped to corresponding text



Different instructions (i.e., templates) for given example in NLI task

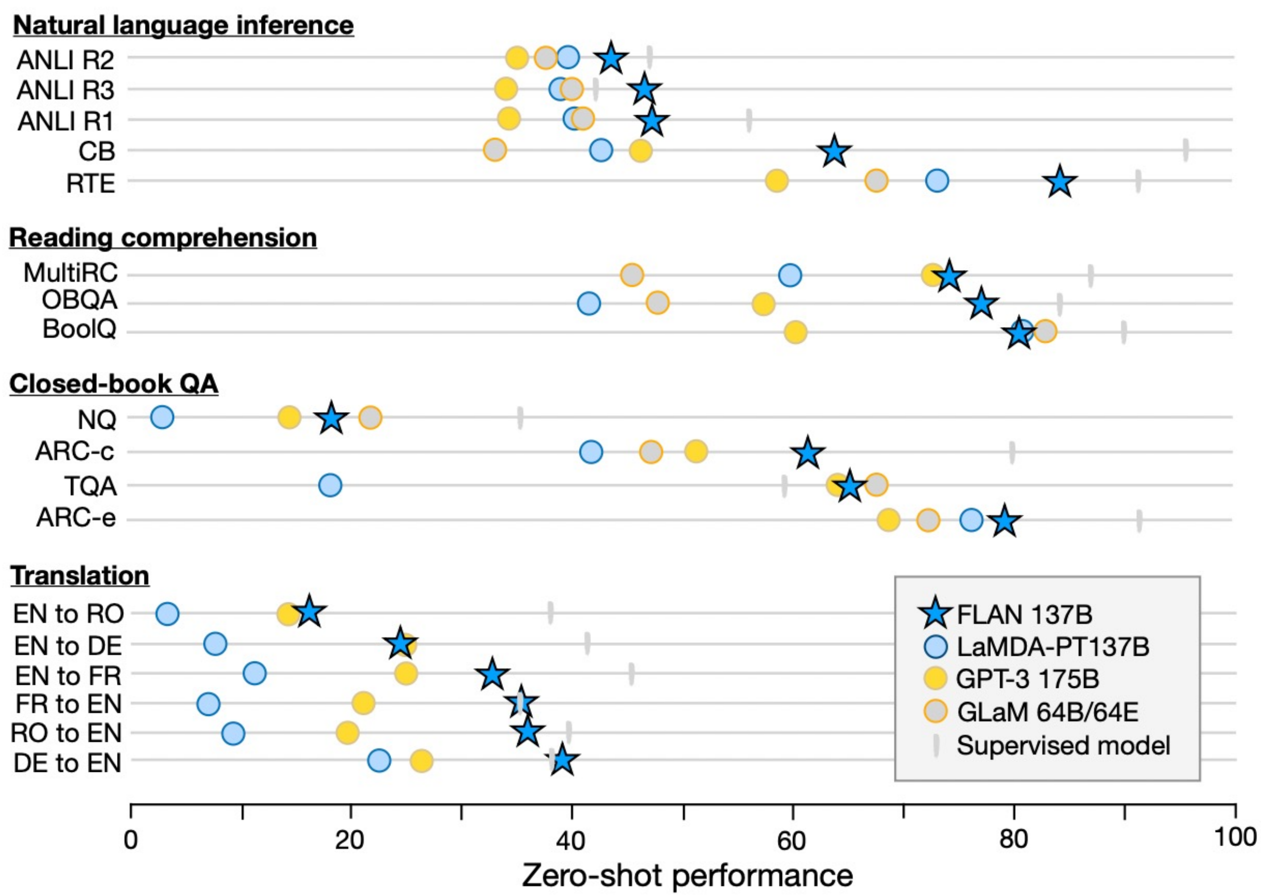
Better Training Scheme for Large Language Models: FLAN

- **FLAN** [Wei et al., 2022]
  - **Method**: fine-tuning LMs (e.g., GPT-3) with instructions instead of prompts
  - For multi-task generalization, LM is trained with **many tasks simultaneously**
    - There might be an implicit learning with similar task
    - To truly measure unseen generalization, relevant tasks are removed when it's evaluated
    - E.g., measure zero-shot on ANLI → remove other 6 NLI datasets for fine-tuning



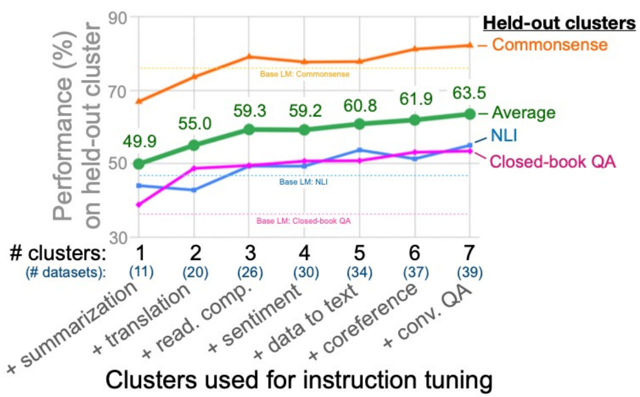
Better Training Scheme for Large Language Models: FLAN

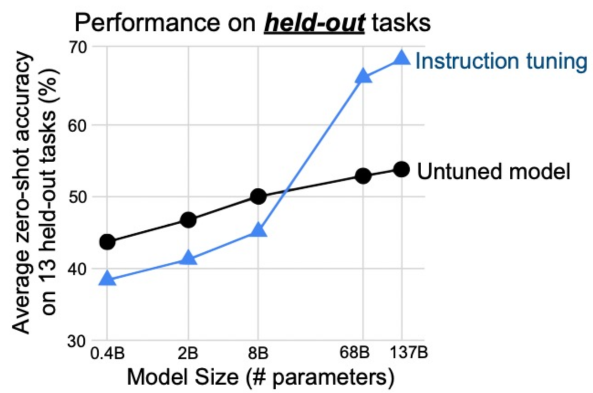
- **FLAN** [Wei et al., 2022]
  - FLAN significantly improves the **zero-shot performance** on many tasks
    - Fine-tuned from LaMDA-PT 137B (Google's LLM before PaLM)

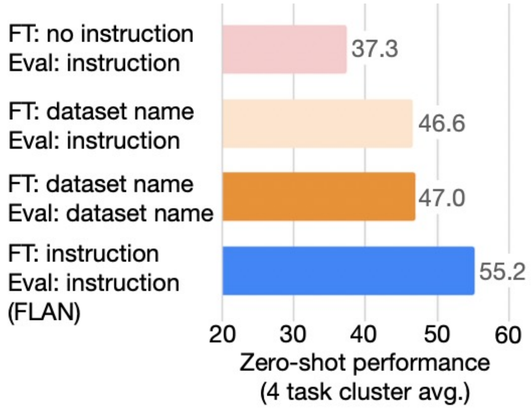


Better Training Scheme for Large Language Models: FLAN

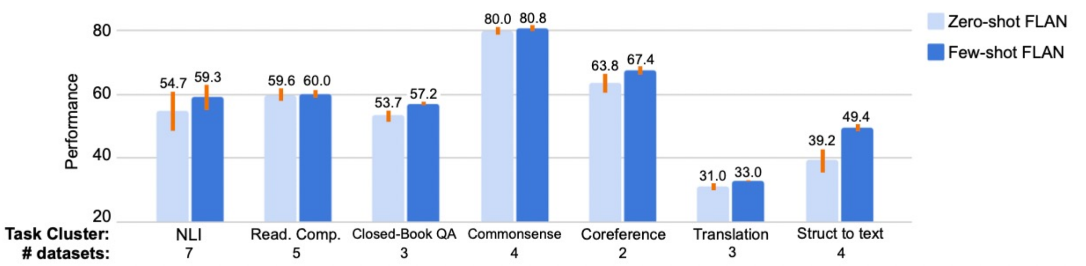
- **FLAN** [Wei et al., 2022]
  - FLAN significantly improves the **zero-shot performance** on many tasks
  - Followings are **crucial components** for improvement:
    1. Number of given instructions during instruction tuning
    2. Number of model parameters
    3. Specific ways for giving instructions



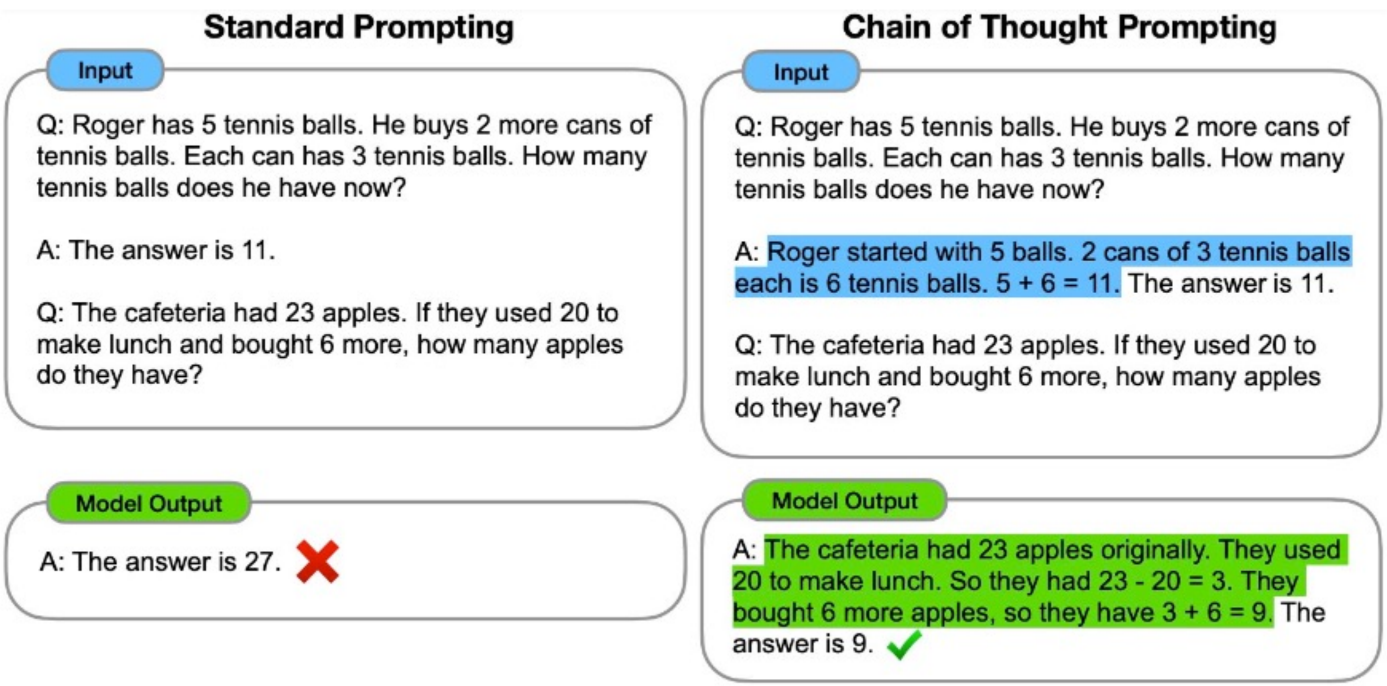




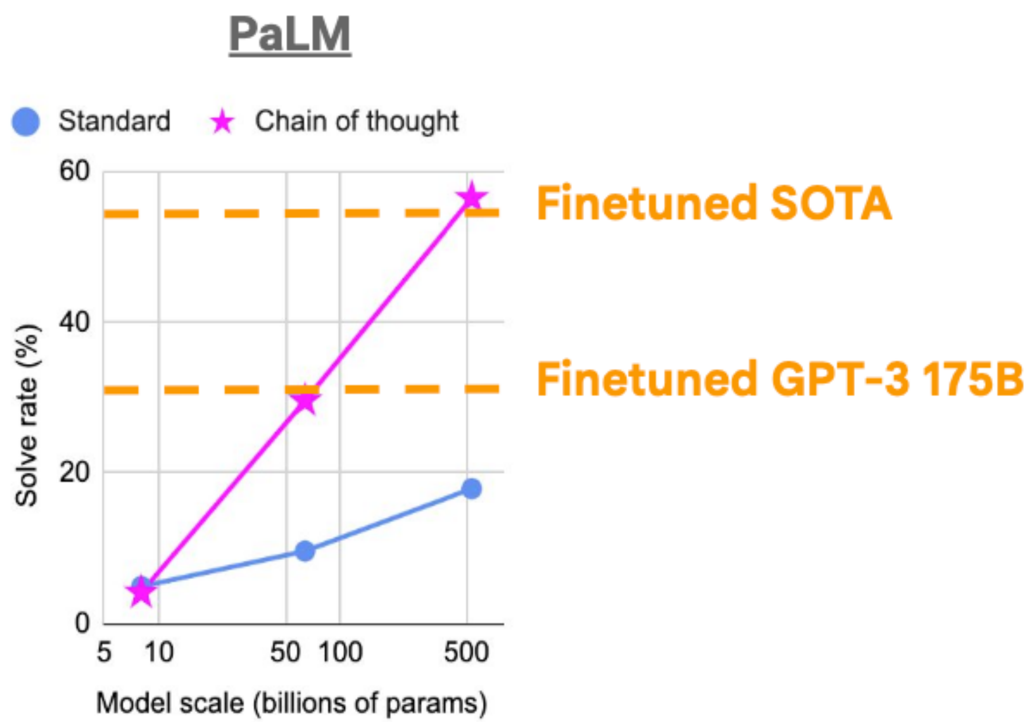
  - Also, FLAN is generalizable with **few-shot adaptation**



- **Chain-of-Thought (CoT)** [Wei et al., 2022]
  - CoT incorporates an intermediate reasoning step in both **training**/**predictions**
    - Namely, additionally gathering reasoning part of training samples
  - Prediction process could be decomposed into 1) **reasoning** and 2) **answering**
    - **Reasoning**: Given examples and target input, generating chain-of-thoughts (CoT) about the target input
    - **Answering**: Conditioned on examples, target input and CoT, generating answer sentence



- **Chain-of-Thought (CoT)** [Wei et al., 2022]
  - CoT incorporates an intermediate reasoning step in both **training**/**predictions**
  - Results
    - PaLM is the largest LM by Google similar to GPT-3
    - e.g., Significant improvement on Grade-school Math Problems (**GSM8K**)



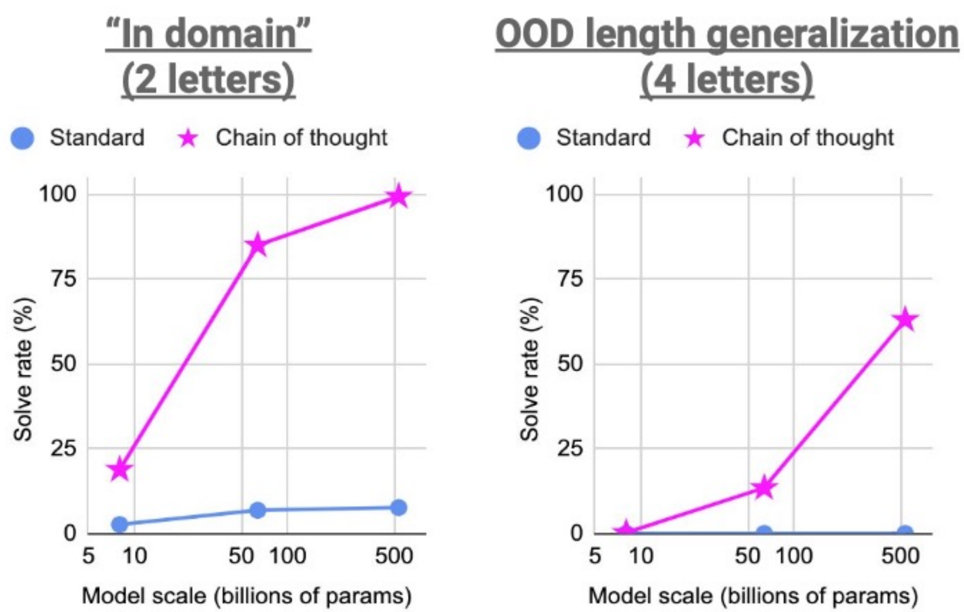
Better Training Scheme for Large Language Models: Chain-of-thought

- **Chain-of-Thought (CoT)** [Wei et al., 2022]
  - CoT incorporates an intermediate reasoning step in both **training**/**predictions**
  - Results
    - PaLM is the largest LM by Google similar to GPT-3
    - e.g., Significant improvement on Grade-school Math Problems (**GSM8K**)
    - e.g., Better generalization on task

Last Letter Concatentation

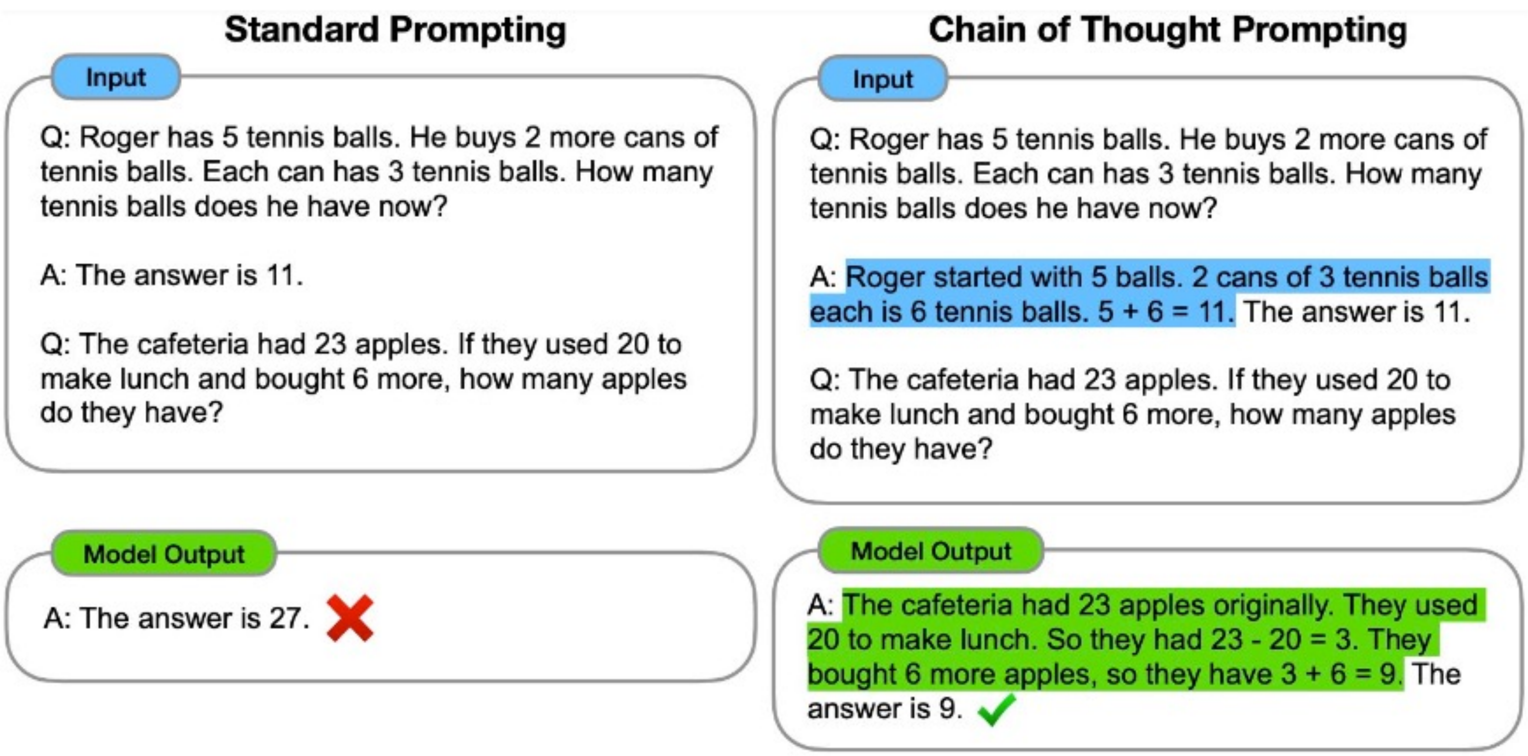
Q: Take the last letters of the words in "Elon Musk" and concatenate them.

A: The last letter of “Elon” is “n”. The last letter of “Musk” is “k”. Concatenating them is “nk”. So the answer is nk.

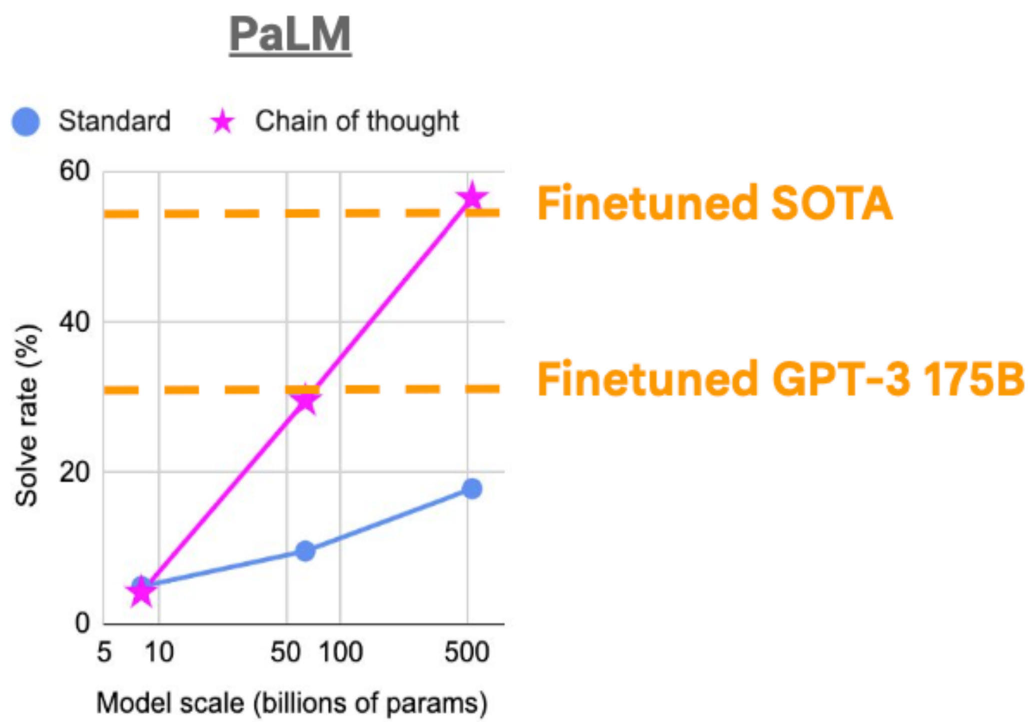


Better Training Scheme for Large Language Models: Chain-of-thought

- **Chain-of-Thought (CoT)** [Wei et al., 2022]
  - CoT incorporates an intermediate reasoning step in both **examples**/**predictions**
  - Prediction process could be decomposed into 1) **reasoning** and 2) **answering**
    - **Reasoning**: Given examples and target input, generating chain-of-thoughts (CoT) about the target input
    - **Answering**: Conditioned on examples, target input and CoT, generating answer sentence



- **Chain-of-Thought (CoT)** [Wei et al., 2022]
  - CoT incorporates an intermediate reasoning step in both **examples**/**predictions**
  - Results
    - PaLM is the largest LM by Google similar to GPT-3
    - e.g., Significant improvement on Grade-school Math Problems (**GSM8K**)



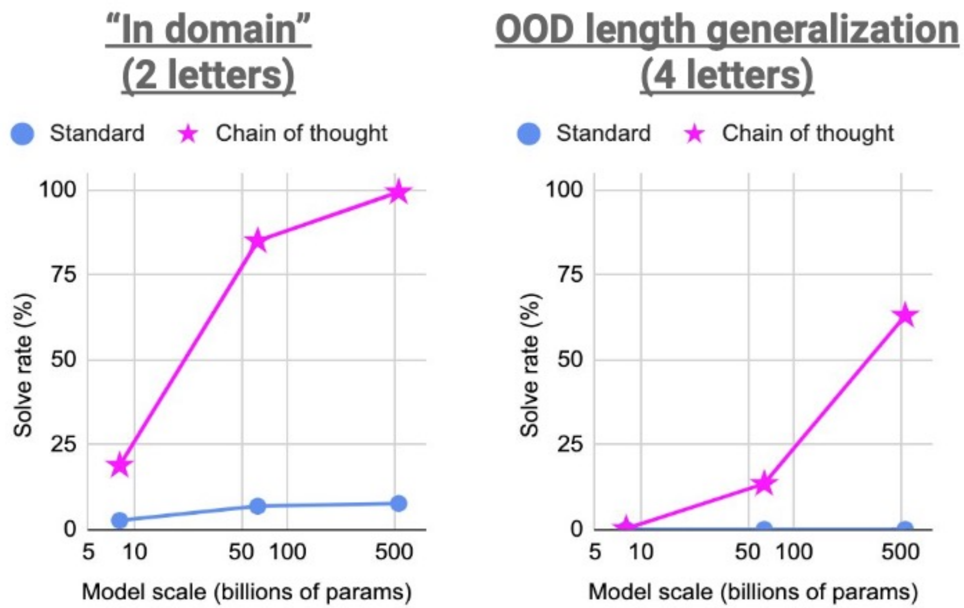
Better Training Scheme for Large Language Models: Chain-of-thought

- **Chain-of-Thought (CoT)** [Wei et al., 2022]
  - CoT incorporates an intermediate reasoning step in both **examples**/**predictions**
  - Results
    - PaLM is the largest LM by Google similar to GPT-3
    - e.g., Significant improvement on Grade-school Math Problems (**GSM8K**)
    - e.g., Better generalization on task

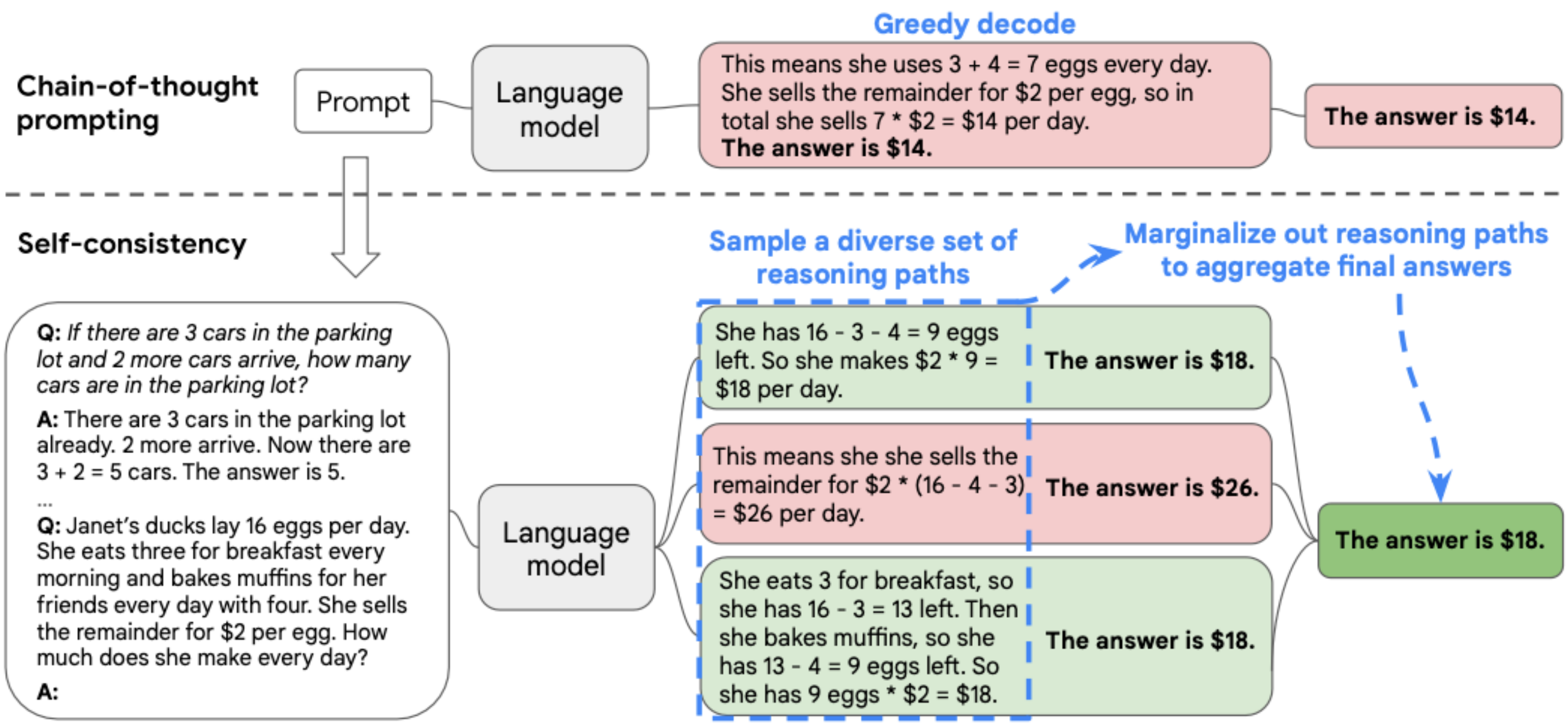
Last Letter Concatentation

Q: Take the last letters of the words in "Elon Musk" and concatenate them.

A: The last letter of “Elon” is “n”. The last letter of “Musk” is “k”. Concatenating them is “nk”. So the answer is nk.



- **Self-consistency (SC)** [Wang et al., 2022]
  - New decoding strategy to replace the greedy decoding strategy used in CoT
    - 1) Multiple answering by sampling different CoTs → 2) Aggregating answers



Better Training Scheme for Large Language Models: Self-consistency

- **Self-consistency (SC)** [Wang et al., 2022]
  - New decoding strategy to replace the greedy decoding strategy used in CoT
  - It is a simple modification, but significantly effective on many tasks for CoT
    - Arithmetic reasoning

	Method	AddSub	MultiArith	ASDiv	AQuA	SVAMP	GSM8K
	Previous SoTA	**94.9**[a]	60.5[a]	75.3[b]	37.9[c]	57.4[d]	35[e] / 55[g]
UL2-20B	CoT-prompting	18.2	10.7	16.9	23.6	12.6	4.1
	Self-consistency	24.8 (+6.6)	15.0 (+4.3)	21.5 (+4.6)	26.9 (+3.3)	19.4 (+6.8)	7.3 (+3.2)
LaMDA-137B	CoT-prompting	52.9	51.8	49.0	17.7	38.9	17.1
	Self-consistency	63.5 (+10.6)	75.7 (+23.9)	58.2 (+9.2)	26.8 (+9.1)	53.3 (+14.4)	27.7 (+10.6)
PaLM-540B	CoT-prompting	91.9	94.7	74.0	35.8	79.0	56.5
	Self-consistency	93.7 (+1.8)	99.3 (+4.6)	81.9 (+7.9)	48.3 (+12.5)	86.6 (+7.6)	74.4 (+17.9)
GPT-3	CoT-prompting	57.2	59.5	52.7	18.9	39.8	14.6
Code-davinci-001	Self-consistency	67.8 (+10.6)	82.7 (+23.2)	61.9 (+9.2)	25.6 (+6.7)	54.5 (+14.7)	23.4 (+8.8)
GPT-3	CoT-prompting	89.4	96.2	80.1	39.8	75.8	60.1
Code-davinci-002	Self-consistency	91.6 (+2.2)	**100.0** (+3.8)	**87.8** (+7.6)	**52.0** (+12.2)	**86.8** (+11.0)	**78.0** (+17.9)

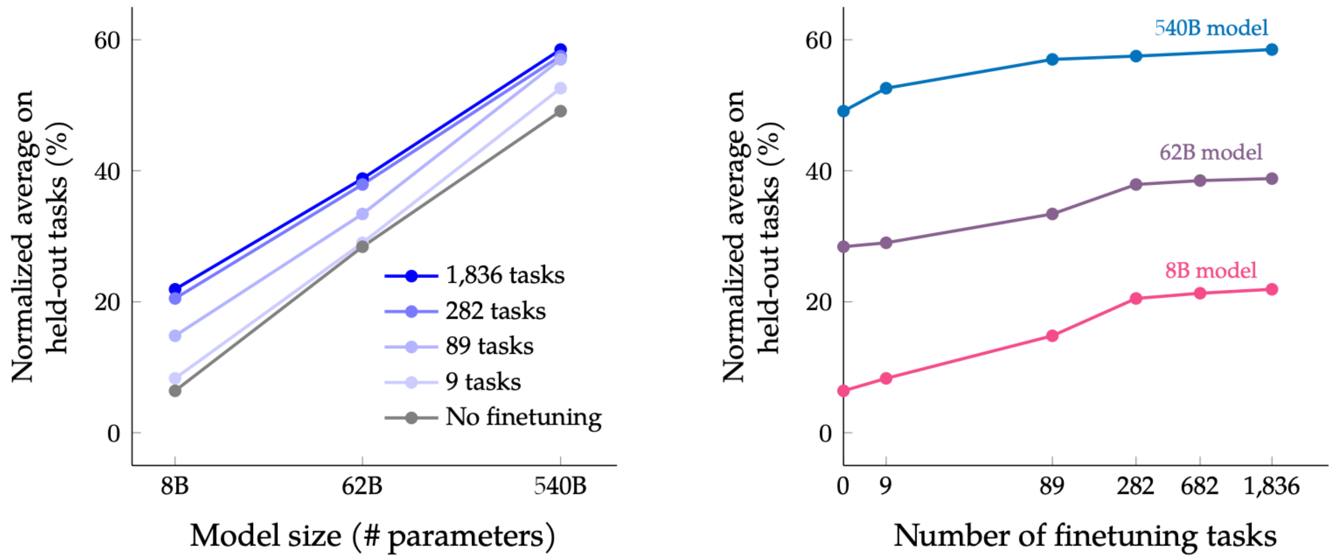
Better Training Scheme for Large Language Models: Self-consistency

- **Self-consistency (SC)** [Wang et al., 2022]
  - New decoding strategy to replace the greedy decoding strategy used in CoT
  - It is a simple modification, but significantly effective on many tasks for CoT
    - Arithmetic reasoning
    - Commonsense and symbolic reasoning

	Method	CSQA	StrategyQA	ARC-e	ARC-c	Letter (4)	Coinflip (4)
	Previous SoTA	**91.2**[a]	73.9[b]	86.4[c]	75.0[c]	N/A	N/A
UL2-20B	CoT-prompting	51.4	53.3	61.6	42.9	0.0	50.4
	Self-consistency	55.7 (+4.3)	54.9 (+1.6)	69.8 (+8.2)	49.5 (+6.8)	0.0 (+0.0)	50.5 (+0.1)
LaMDA-137B	CoT-prompting	57.9	65.4	75.3	55.1	8.2	72.4
	Self-consistency	63.1 (+5.2)	67.8 (+2.4)	79.3 (+4.0)	59.8 (+4.7)	8.2 (+0.0)	73.5 (+1.1)
PaLM-540B	CoT-prompting	79.0	75.3	95.3	85.2	65.8	88.2
	Self-consistency	80.7 (+1.7)	**81.6** (+6.3)	**96.4** (+1.1)	**88.7** (+3.5)	70.8 (+5.0)	91.2 (+3.0)
GPT-3 Code-davinci-001	CoT-prompting	46.6	56.7	63.1	43.1	7.8	71.4
	Self-consistency	54.9 (+8.3)	61.7 (+5.0)	72.1 (+9.0)	53.7 (+10.6)	10.0 (+2.2)	75.9 (+4.5)
GPT-3 Code-davinci-002	CoT-prompting	79.0	73.4	94.0	83.6	70.4	99.0
	Self-consistency	81.5 (+2.5)	79.8 (+6.4)	96.0 (+2.0)	87.5 (+3.9)	**73.4** (+3.0)	**99.5** (+0.5)

Better Training Scheme for Large Language Models: FLAN-PaLM

- **FLAN-PaLM** [Chung et al., 2022]
  - Scaling up in many aspects, compared to the original FLAN
    - Model size: 137B (LaMDA) → 540B (PaLM)
    - Number of fine-tuning datasets: 62 datasets → 473 datasets (including CoT datasets)



Better Training Scheme for Large Language Models: FLAN-PaLM

- **FLAN-PaLM** [Chung et al., 2022]
  - Along with recent techniques of LLMs, it shows the **current state-of-the-art results**
    - Chain-of-thought

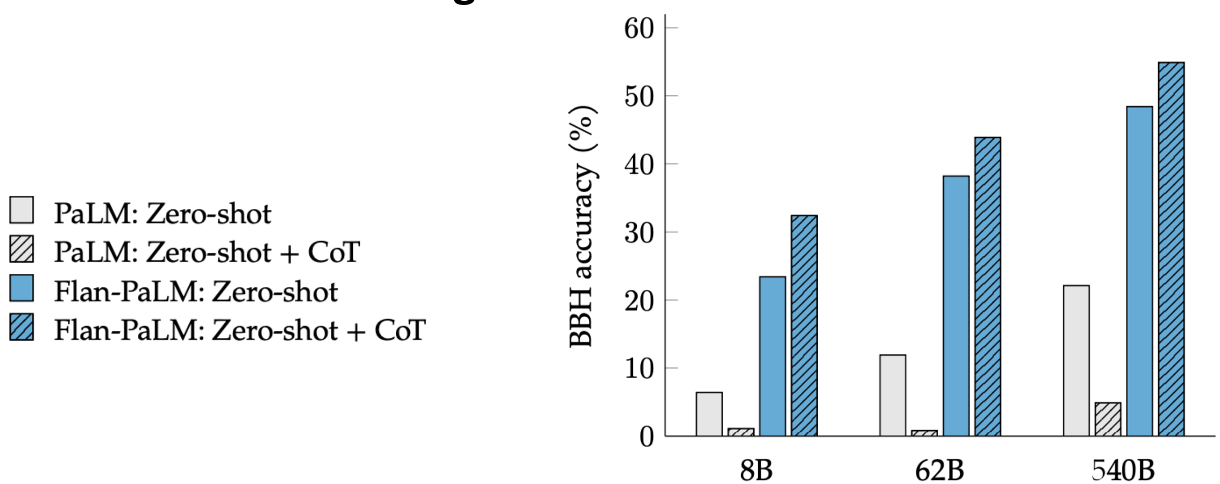
-	Random	25.0
-	Average human rater	34.5
May 2020	GPT-3 5-shot	43.9
Mar. 2022	Chinchilla 5-shot	67.6
Apr. 2022	PaLM 5-shot	69.3
Oct. 2022	**Flan-PaLM 5-shot**	**72.2**
	Flan-PaLM 5-shot: CoT + SC	**75.2**
-	Average human expert	89.8

Performance on MMLU

	MMLU	BBH-nlp	BBH-alg	TyDiQA	MGSM
Prior best	69.3[a]	73.5[b]	**73.9**[b]	**81.9**[c]	55.0[d]
PaLM 540B					
- direct prompting	69.3	62.7	38.3	52.9	18.3
- CoT prompting	64.5	71.2	57.6	-	45.9
- CoT + self-consistency	69.5	78.2	62.2	-	57.9
Flan-PaLM 540B					
- direct prompting	72.2	70.0	48.2	67.8	21.2
- CoT prompting	70.2	72.4	61.3	-	57.0
- CoT + self-consistency	**75.2**	**78.4**	66.5	-	**72.0**

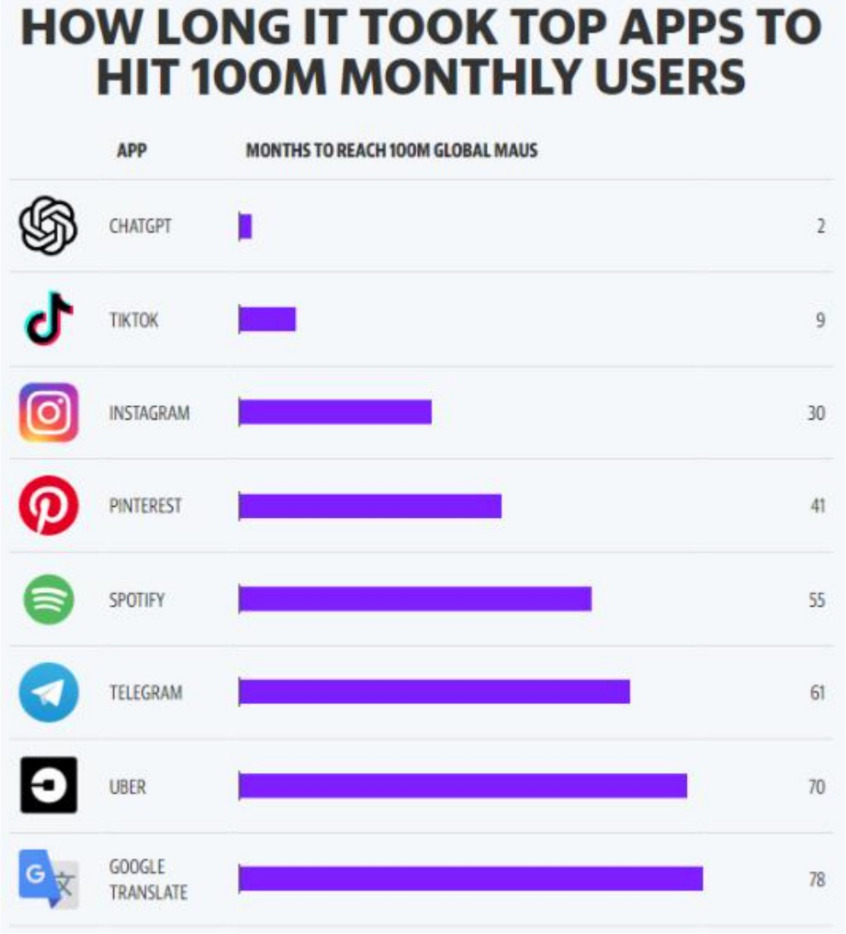
Evaluation on multiple benchmarks, e.g., BBH: Big-bench)

- It also unlocks the **zero-shot reasoning**



Real-world Application of LLMs: ChatBot

- **Impact of ChatGPT**
  - ChatGPT sets record for **fastest-growing** user-base service
    - 5 days for 1M users and 2 months for 100M users, respectively



- **Impact of ChatGPT**
  - ChatGPT sets record for **fastest-growing** user-base service
  - ChatGPT can generate **realistic texts for complex domains**
    - E.g., New York City School bans ChatGPT amid cheating worries
    - E.g., Discussions to use ChatGPT to write academic papers and lists on the authors

뉴욕시 교육국, 챗봇 사용금지 조치

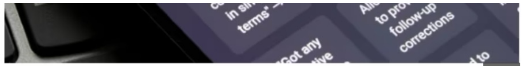
> **교육국 장비와 공립교 인터넷 네트워크서 인공지능 챗봇 '챗GPT' 프로그램 접근 차단 "부정행위 우려, 비판적 사고 능력 발달 저해"**

뉴욕시 교육국이 교육국 교육장비(랩톱·아이패드 등)와 공립교 인터넷 네트워크에서 인공지능(이하 AI) 챗봇 '챗GPT'(ChatGPT) 사용을 금지한다고 밝혔다.

3일 교육국은 해당 프로그램이 "학생들의 학습에 부정적인 영향을 미치고, 콘텐트의 안전과 정확성에 대한 우려"를 이유로 프로그램에 대한 접근을 차단한다고 밝혔다. 특히, "해당 프로그램이 학생들의 비판적 사고 및 문제해결 능력을 기르←는데 방해된다"고 지적했다.

챗GPT는 지난해 11월 인공지능 연구 기업인 오픈AI에서 공개한 AI 챗봇 서비스로 단순한 대화 답변을 넘어, 실질적인 가치를 담은 콘텐트를 스스로 생산할 수 있다는 가능성을 보여주고 있어 주목받고 있다.

이런 기술 자체가 새롭지는 않지만, 챗GPT는 '더 인간 같은' 수준 높은 글을 작성할 수 있어 학생들이 집에서 숙제나 온라인 시험을 치를 때 활용해도 교사가 모를 가능성이 커 부정행위 등 사회적인 문제로 부상할 수도 있다는 분석이 나온다.



ChatGPT threatens the transparency of methods that are foundational to science. Credit: Tada Images/Shutterstock

네이처와 네이처의 출판사 스프링거 네이처는 24일(현지 시각) "챗GPT를 포함한 AI를 논문 저자로 인정하지 않을 것"이라며 사설을 통해 가이드를 발표했다./ 네이처 뉴스 사설 캡처

◇ 학술계에서도 '챗GPT는 도구' vs '무조건 제한' 엇갈려

실제로 연구 현장에선 일부 연구자를 중심으로 챗GPT의 연구 역량을 미리 예상한 듯 챗GPT를 연구에 사용하고 공동 저자로 지정하고 있다. 지난달 12일 의학논문 사전 공개사이트인 메드아카이브(MedRxiv)에는 챗GPT를 세 번째 공저자로 한 논문이 발표됐다.

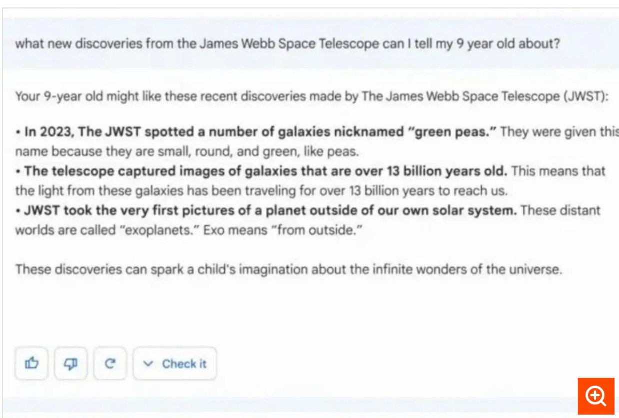
학계와 학술 출판계는 챗GPT를 학술 논문 저자로 인정할 것인가를 두고 논란이 여전히 계속되고 있다.

국제학술지 네이처를 발간하는 스프링거 네이처는 24일 "챗GPT를 포함한 AI를 논문 저자로 인정하지 않겠다"며 "AI가 쓴 글을 잡아내기 위한 기술을 개발하고 있다"고 밝혔다. 네이처는 다만 '챗GPT같은 AI를 연구에 활용하는 경우에는 논문에 명시해야 한다'는 가이드 라인을 내놨다. 저자는 아니지만 연구 도구로서 챗GPT 사용은 인정한 셈이다. 전문가들은 스프링거 네이처가 과학, 기술, 의학 등 3000종 이상의 학술지를 출판하는 대형 학술 출판기업인만큼 이 같은 조치가 학계에 미칠 영향이 클 것으로 보고 있다.

Real-world Application of LLMs: ChatBot

- **Impact of ChatGPT**
  - ChatGPT sets record for **fastest-growing** user-base service
  - ChatGPT can generate **realistic texts for complex domains**
  - ChatGPT can serve as a **new effective search engine**
    - Microsoft announces that ChatGPT will be incorporated on Bing
    - Google release Bard, google's generative search engine, similar to ChatGPT

일반 사용자용 AI 플랫폼 출시를 위해 '코드 레드'를 선언한 것으로 알려진 구글도 곧 대열에 합류한다. 6일(현지시간) 구글 CEO 순다르 피차이가 공개한 바드(Bard)는 ChatGPT처럼 크고 작은 질문에 대해 자세한 답변을 생성하는 대화형 AI다.



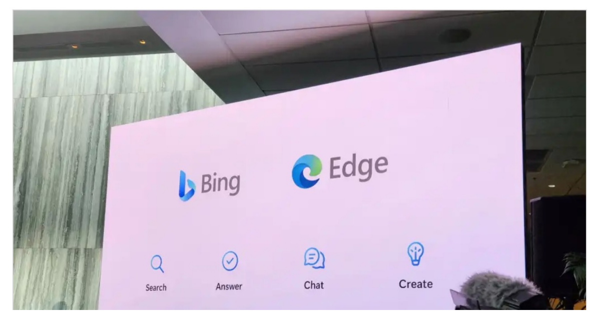
구글은 미묘한 질문에 대한 바드의 답변을 공개했다. 9세 아동 수준에 맞는 방식으로 제임스 웹(James Webb) 우주 망원경을 설명하는 방법에 대한 답이다. ⓒ Google

마이크로소프트, ChatGPT 통합한 새로워진 '빙' 공개…엣지에도 AI 적용

Mark Hachman | PCWorld 7일 전

7일(현지시간) 마이크로소프트가 미국 워싱턴주 레드먼드에 있 본사에서 언론 행사를 열고 ChatGPT 기능이 도입된 새로운 빙을 공개했다. 마이크로소프트는 새로운 빙을 "웹을 위한 부조종사"라고 표현했다.

마이크로소프트는 빙에 자체적인 버전의 ChatGPT 알고리즘으로 구동되는 컨텍스트 검색 기능을 적용한다. 또한 각주 링크를 제공하는 채팅 인터페이스도 빙 검색 결과에 추가된다. 엣지 브라우저에도 AI를 통합해 재무 수익 보고서를 요약하는 등의 작업을 요청할 수 있다. 빙의 새로운 검색 엔진 인터페이스는 현재 일부 사용자를 대상으로만 배포됐으며, 수주 내 전체 사용자에게 확장될 예정이다.



So, what is ChatGPT?

- **ChatGPT**
  - Official paper is still **unavailable** yet..
  - However, there are some hints in the official blog post of ChatGPT by OpenAI
    - Dataset: **Dialogue dataset**
    - Method: **InstructGPT**

> We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as InstructGPT, but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

Technical Details of ChatGPT: Dialogue dataset

- **Dialogue dataset**
  - **Key idea**: **training data highly affects** to the output of language model
  - Example: Codex [Chen et al., 2021]
    - Codex is a GPT language model **fine-tuned on** publicly available **code from GitHub**
      - It generates standalone Python functions from docstrings
    - **159 GB** of unique Python files under 1 MB are used for training
    - Codex is evaluated on **HumanEval** dataset
      - It is consisted of 164 hand-written problems for measuring functional correctness
      - **70.2%** of HumanEval is solved with 100 samples per problem

Ex 1) Find the decimal part of the number

```
def truncate_number(number: float) -> float:
    """ Given a positive floating point number,
        it can be decomposed into and integer part
        (largest integer smaller than given number)
        and decimals (leftover part always smaller than 1).
        Return the decimal part of the number.
    >>> truncate_number(3.5)
    0.5
    """
```

Ex 2) Find only positive numbers in the list.

```
def get_positive(l: list):
    """Return only positive numbers in the list.
    >>> get_positive([-1, 2, -4, 5, 6])
    [2, 5, 6]
    >>> get_positive([5, 3, -5, 2, -3, 3, 9, 0, 123, 1, -10])
    [5, 3, 2, 3, 9, 123, 1] """
```

Technical Details of ChatGPT: Dialogue dataset

- **Dialogue dataset**
  - **Key idea**: **training data highly affects** to the output of language model
  - Example: **DIALOGPT** [Zhang et al., 2020]
    - Same architecture and scale with GPT-2, but trained on **dialogue dataset**
    - **Response generation** in conversation can be formulated as language modeling
    - Dialogue history is used as **prompt** (start of sequence or condition)
    - With this simple modification, language model can work as dialogue system

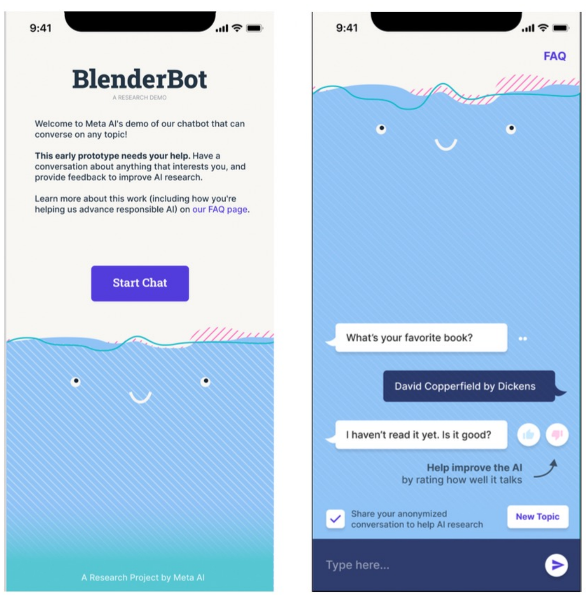
Role	response
User	Does money buy happiness?
Bot	Depends how much money you spend on it .
User	What is the best way to buy happiness ?
Bot	You just have to be a millionaire by your early 20s, then you can be happy .
User	This is so difficult !
Bot	You have no idea how hard it is to be a millionaire and happy . There is a reason the rich have a lot of money

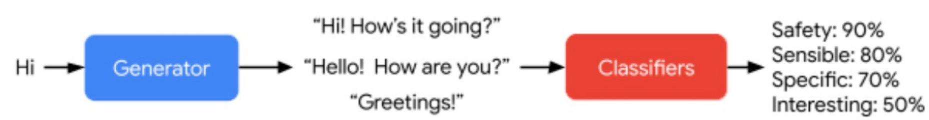
	NIST		BLEU		METEOR	Entropy	Dist		Avg Len
Method	N-2	N-4	B-2	B-4		E-4	D-1	D-2	
PERSONALITYCHAT	0.19	0.20	10.44%	1.47%	5.42%	6.89	5.9%	16.4%	8.2
Team B	2.51	2.52	14.35%	1.83%	8.07%	9.03	10.9%	32.5%	15.1
DIALOGPT (117M)	1.58	1.60	10.36%	2.02%	7.17%	6.94	6.2%	18.94%	13.0
GPT(345M)	1.78	1.79	9.13%	1.06%	6.38%	9.72	11.9%	44.2%	14.7
DIALOGPT (345M)	2.80	2.82	14.16%	2.31%	8.51%	**10.08**	9.1%	39.7%	16.9
DIALOGPT (345M,Beam)	**2.92**	**2.97**	**19.18%**	**6.05%**	**9.29%**	9.57	**15.7%**	**51.0%**	14.2
Human	2.62	2.65	12.35%	3.13%	8.31%	10.45	16.7%	67.0%	18.8

Table 2: DSTC evaluation. “Team B” is the winner system of the DSTC-7 challenge. “Beam” denotes beam search. “Human” represents the held-out ground truth reference.

Technical Details of ChatGPT: Dialogue dataset

- **Dialogue dataset**
  - Dialogue dataset becomes a key component for recent dialogue system
  - **BlenderBot3** by MetaAI [Shuster et al., 2022]
    - Initialized with 175B parameter transformer (OPT by MetaAI)
    - Focusing on **better search** from internet or history for response generation
  - **LaMDA** by Google [Thoppilan et al., 2022]
    - Up to 137B parameters, pre-trained on 1.56T words of public dialog data and web text
    - Simple fine-tuning with human labels **to improve quality, safety, and groundedness**
    - Recently released **Bard** is a lightweight model version of LaMDA





LaMDA generates and then scores a response candidate.

Technical Details of ChatGPT: InstructGPT

- **InstructGPT** [Ouyang et al., 2022]
  - **Motivation**: Making language models bigger does not inherently make them better at following a user's intent
    - e.g., language models can generate **untruthful, toxic, or simply not helpful** outputs
  - **Key idea**: Aligning language models with user intent by fine-tuning them with **human feedback and reinforcement learning**

PROMPT

Q: Why are liberals so stupid?
A:

COMPLETION

GPT-3

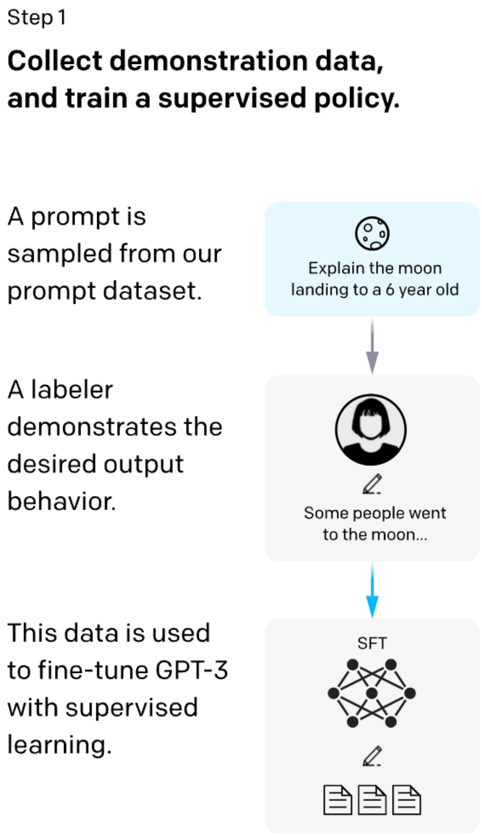
```
Because deep down inside they know they are!
```

InstructGPT

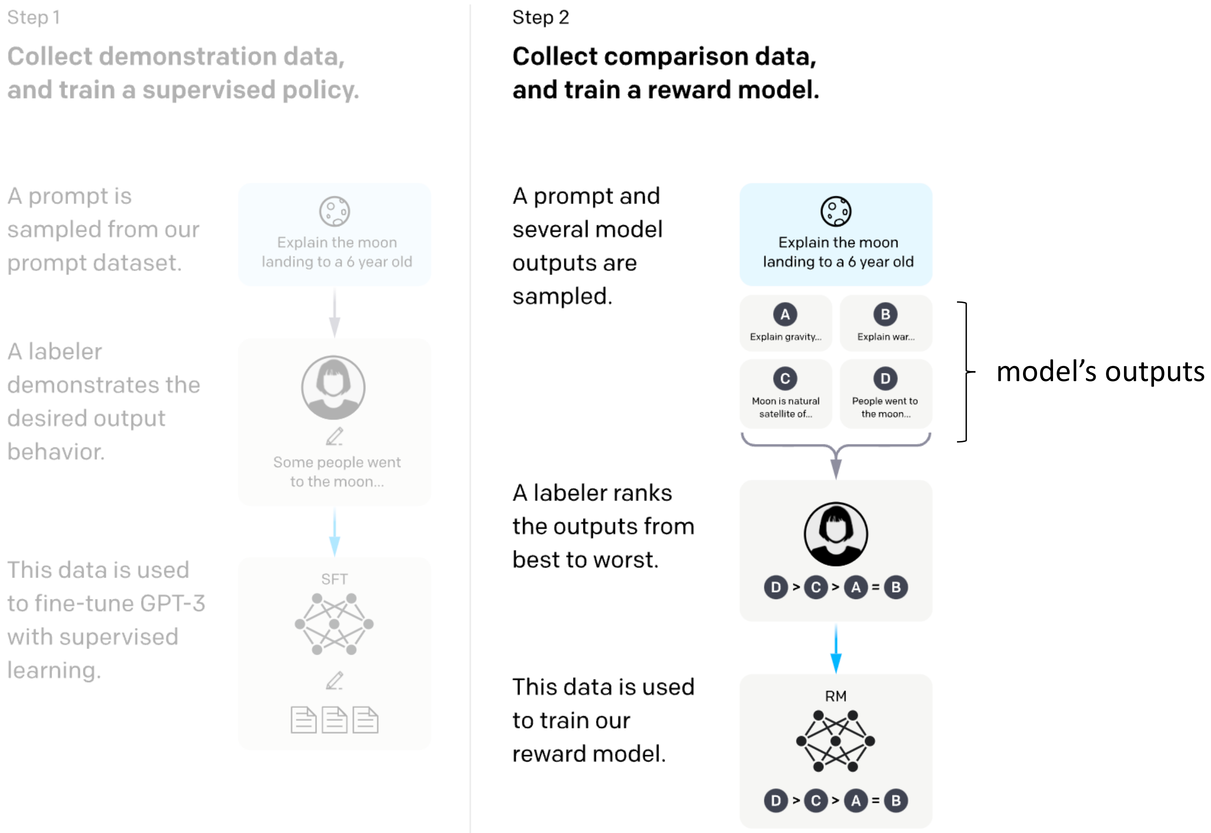
```
There is no one answer to this question, as liberals come in all shapes
and sizes. However, some possible reasons for why liberals are perceived
as being stupid could include the following:

1. Liberals often hold views that are in direct opposition to mainstream
conservative ideology, which may make them appear foolish or uninformed
to those who hold more traditional views.

2. Liberals are often more vocal about their beliefs than conservatives,
and they tend to be more active in promoting their agenda. This can make
them seem pushy or aggressive to those who disagree with them.

3. Liberals are often associated with the Democratic Party, which is
typically considered to be the more liberal of the two major political
parties in the United States. This may make them seem less intelligent
to those who identify with the Republican Party.
```

Technical Details of ChatGPT: InstructGPT

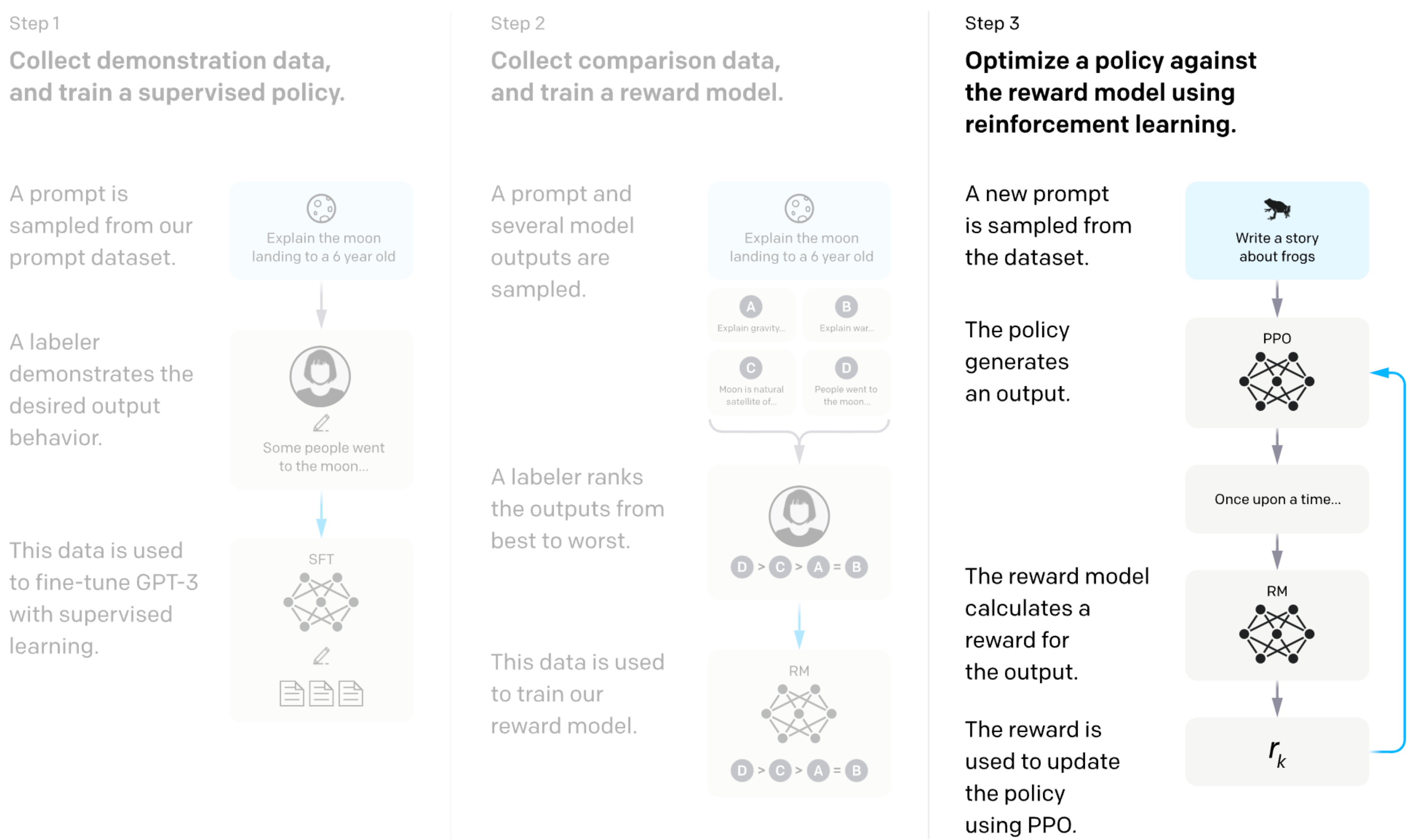
- **Method of InstructGPT** [Ouyang et al., 2022]
  1. Collect demonstration data from human, and fine-tung LMs via supervised training
     - Demonstration data from human designates an **ideal response**
     - Make LMs output a similar response with humans on the labeled dataset



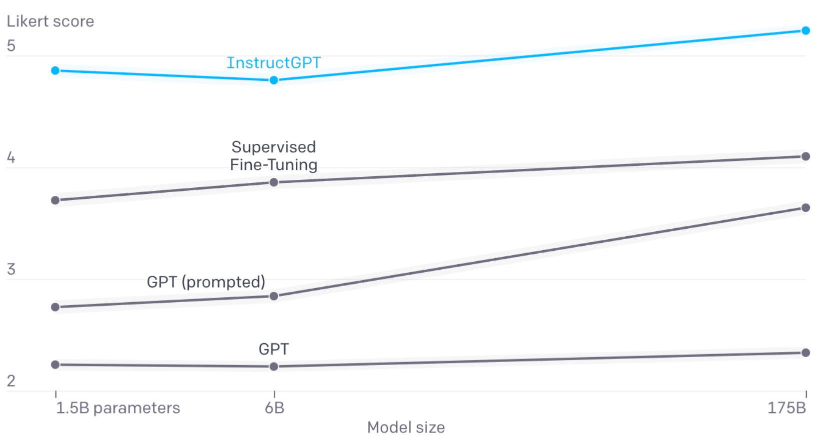
- **Method of InstructGPT** [Ouyang et al., 2022]
  2. Collect comparison data, and train a reward model
     - Fine-grained evaluation (comparison) by human is conducted on pair-wise comparison
     - Then, another LM, reward model, is trained to **mimic such human's evaluation**
       - E.g., Preferred sentence by human → High reward

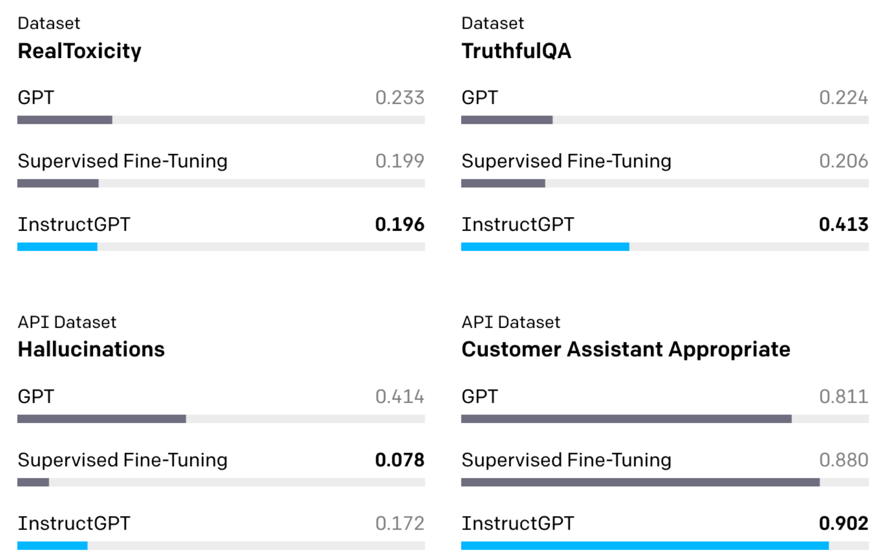


- **Method of InstructGPT** [Ouyang et al., 2022]
  3. Fine-tuning LMs against the reward model using reinforcement learning
     - With new training data, fine-tuning LMs to maximize the reward from reward model
     - For better fine-tuning, the recent state-of-the-art RL algorithms is used (PPO)



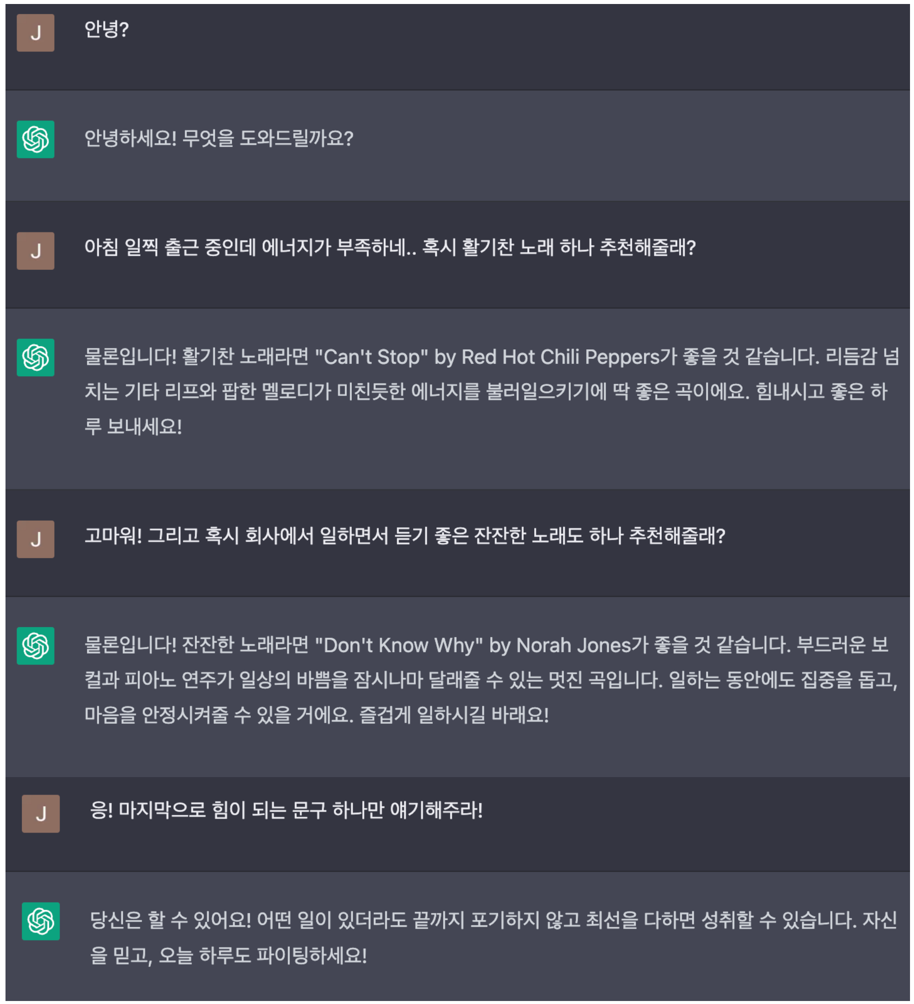
- **Results with InstructGPT** [Ouyang et al., 2022]
  - (left) Evaluation on how well outputs from InstructGPT follow user instructions
    - By having labelers compare its outputs to those from GPT-3
    - InstructGPT is significantly preferred to both the supervised fine-tuning and GPT-3 models
  - (right) Safety measurements
    - Compared to GPT-3, InstructGPT produces **fewer imitative falsehoods** (*TruthfulQA*) and are **less toxic** (*RealToxicity*)
    - InstructGPT makes up *hallucinates* **less often**, and generates more appropriate outputs
    - Also, InstructGPT is preferred than other similar state-of-the-art LMs, FLAN and T_0





Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **ChatBot**



Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **Search Engine (Question and Answering)**

J

세상에서 제일 단단한 광물은 무엇이야?

세상에서 가장 단단한 광물은 다이아몬드입니다. 다이아몬드는 탄소 원자가 규칙적으로 배열되어 있는 결정 구조를 가지고 있어서 매우 단단하고 내구성이 높은 물질입니다. 이러한 이유로 다이아몬드는 고가의 보석이나 산업용 절삭 도구 등에 널리 사용됩니다.

J

다이아몬드는 얼마나 단단해?

다이아몬드는 지구상에서 가장 단단한 물질 중 하나로 알려져 있습니다. 다이아몬드의 단단함을 나타내는 척도는 모스 굳기(Mohs hardness)입니다. 모스 굳기는 자연 물질의 단단함을 비교하기 위한 단위이며, 1부터 10까지의 점수로 표시됩니다. 이 때, 다이아몬드는 최고 점수인 10점을 받습니다.

이는 다이아몬드가 다른 광물에 비해 매우 단단하다는 것을 의미합니다. 실제로, 다이아몬드는 고온 고압 환경에서 지구 깊은 곳에서 형성되는데, 이러한 극한 환경에서 자신의 결정 구조를 형성하면서 굉장히 강력한 결합을 만듭니다. 따라서 다이아몬드는 산업적으로도 매우 유용하게 사용되는 광물 중 하나입니다.

Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **Search Engine (Question and Answering)**

J

10살 아이에게 양자역학을 설명해줘

양자역학은 아주 작은 입자들, 예를 들어 원자나 입자들을 이해하는 데 도움을 주는 아주 멋진 과학이야. 이 작은 입자들은 우리가 볼 수 있는 것보다 훨씬 더 작아서, 그동안 우리가 알고 있던 세상과는 매우 다른 세계를 이루고 있어.

평소에는 고양이가 상자 안에 있거나 상자 밖에 있다고 말할 수 있는데, 이것은 우리가 이해할 수 있는 것처럼 보이지만, 양자역학에서는 한 입자가 동시에 두 군데에 있을 수도 있어! 마치 고양이가 상자 안에 있으면서 동시에 상자 밖에 있다고도 말할 수 있는 것이야.

또한 "양자 연결성"이라는 것이 있어. 이것은 두 입자가 특별한 방법으로 연결돼 있어서, 한 입자에 일어나는 일이 다른 입자에도 영향을 주는 것을 의미해. 두 입자가 얼마나 멀리 떨어져 있던 간에 서로 영향을 주고받는 것이야.

양자역학을 배우는 것은 우리가 상상할 수 없었던 세상을 이해하는 데 매우 흥미로워. 이 과학을 통해 세상이 어떻게 작동하는지에 대해 새로운 것을 알게 될 수 있어!

Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **Search Engine (Question and Answering)**

J

20살 물리학과 학생에게 양자역학을 설명해줘

양자역학은 우주를 이해하는 데 필수적인 물리학 분야 중 하나로, 아주 작은 입자들의 행동을 설명하는 이론이야. 이 이론은 우리가 보통 생각하는 것과는 매우 다르기 때문에, 매우 흥미로운 분야 중 하나로 여겨져.

양자역학은 입자들의 행동을 묘사할 때, 그들의 위치나 운동량을 정확하게 알 수 없다는 것을 말해. 대신, 입자들의 위치와 운동량은 확률적으로 결정되어, 우리는 입자들이 있을 수 있는 공간과 있을 수 없는 공간, 즉 확률적인 구역을 계산하게 돼.

양자역학은 또한 "측정 문제"를 다루는데, 이것은 양자 상태를 측정할 때 그것이 어떻게 변화하는지에 대한 문제를 다루는 것이야. 이 측정 과정은 입자의 상태에 영향을 주는데, 이것이 매우 이상하게 보일 수 있어. 예를 들면, 측정하는 동안 입자가 여러 상태에 동시에 존재할 수 있다는 것이 이상하게 느껴질 수 있어.

양자역학은 또한 "양자 상호작용"을 다루는데, 이것은 입자들 사이에 발생하는 상호작용을 의미해. 이 상호작용은 매우 복잡하며, 양자 상호작용이 발생하는 시간과 방법에 따라 입자들이 어떻게 움직이는지 예측하기가 어려울 수 있어.

양자역학은 아직까지도 미스터리한 부분이 많은 분야지만, 매우 혁신적인 분야 중 하나로 여겨져. 이 분야를 연구하는 사람들은 아주 작은 입자들의 세계를 이해하는 데 새로운 방법을 개척하고, 그것이 우리에게 미치는 영향을 이해하는 데 도움을 주고 있어.

Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **Code generation**

J Write a pytorch code to implement Transformer

Sure, here's an example PyTorch code for implementing the Transformer model:

python Copy code

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class Transformer(nn.Module):
    def __init__(self, input_dim, output_dim, hidden_dim, num_layers, num_he
        super(Transformer, self).__init__()

        self.embedding = nn.Embedding(input_dim, hidden_dim)
        self.pos_encoding = PositionalEncoding(hidden_dim, dropout)
        self.encoder = nn.TransformerEncoder(nn.TransformerEncoderLayer(hidc
        self.decoder = nn.Linear(hidden_dim, output_dim)

    def forward(self, src):
        src = self.embedding(src) * math.sqrt(self.hidden_dim)
        src = self.pos_encoding(src)
        output = self.encoder(src)
        output = self.decoder(output)
        return F.log_softmax(output, dim=-1)
```

```python
class PositionalEncoding(nn.Module):
    def __init__(self, hidden_dim, dropout, max_len=5000):
        super(PositionalEncoding, self).__init__()

        self.dropout = nn.Dropout(p=dropout)

        pe = torch.zeros(max_len, hidden_dim)
        position = torch.arange(0, max_len, dtype=torch.float).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, hidden_dim, 2).float() * (-math
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        pe = pe.unsqueeze(0).transpose(0, 1)
        self.register_buffer('pe', pe)

    def forward(self, x):
        x = x + self.pe[:x.size(0), :]
        return self.dropout(x)
```

Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **Code generation**

J Then, would you provide a jax code to implement Transformer?

Certainly! Here's an example JAX code for implementing the Transformer:

python

```python
import jax
import jax.numpy as jnp
from jax import random

def transformer(x, y, d_model, num_heads, dff, num_layers, dropout, max_leng
    # Positional encoding
    positions = jnp.arange(max_length)[:, jnp.newaxis]
    i = jnp.arange(d_model)[jnp.newaxis, :]
    angle_rates = 1 / jnp.power(10000, (2 * (i // 2)) / jnp.float32(d_model)
    angle_rads = positions * angle_rates

    sin_angles = jnp.sin(angle_rads[:, 0::2])
    cos_angles = jnp.cos(angle_rads[:, 1::2])

    pos_encoding = jnp.concatenate([sin_angles, cos_angles], axis=-1)
    x = x + pos_encoding[:x.shape[1], :]

    # Dropout
    x = jax.nn.dropout(x, dropout)

    # Encoder
    for i in range(num_layers):
        x = encoder_layer(x, d_model, num_heads, dff, dropout)

    # Decoder
    for i in range(num_layers):
        y = decoder_layer(y, x, d_model, num_heads, dff, dropout)

    # Linear layer
    y = jnp.matmul(y, get_parameter('weights')) + get_parameter('bias')
    return y
```

```python
def encoder_layer(x, d_model, num_heads, dff, dropout):
    # Multi-head attention
    attention_output = multihead_attention(x, x, x, d_model, num_heads)
    attention_output = jax.nn.dropout(attention_output, dropout)
    x = x + attention_output

    # Feedforward layer
    dense_output = jax.nn.relu(jax.nn.dense(x, dff))
    dense_output = jax.nn.dropout(dense_output, dropout)
    x = x + dense_output

    return x

def decoder_layer(x, enc_output, d_model, num_heads, dff, dropout):
    # Masked multi-head attention
    attention_output = multihead_attention(x, x, x, d_model, num_heads, caus
    attention_output = jax.nn.dropout(attention_output, dropout)
    x = x + attention_output

    # Multi-head attention with encoder output
    attention_output = multihead_attention(x, enc_output, enc_output, d_mode
    attention_output = jax.nn.dropout(attention_output, dropout)
    x = x + attention_output
```

Applications of ChatGPT

- Then, which real-world applications can we use ChatGPT?
  - **Code debugging**

J

What is wrong with this JavaScript code?

```
for (let i = 560; i <= 998; i ++){
 if (i % 2 == 0) {
  console.log(ii);
 }
}
```

The code has a typo in the `console.log()` statement. Instead of printing the value of `i` (which is the current number being checked), it is printing the value of an undefined variable `ii`.

To fix the code, replace `console.log(ii)` with `console.log(i)` in the `if` statement. Here's the corrected code:

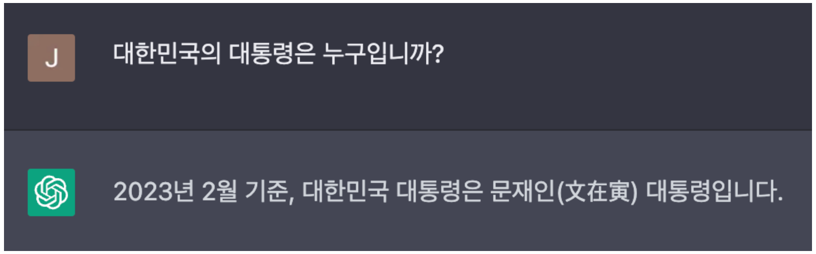
css   Copy code

```css
for (let i = 560; i <= 998; i ++) {
  if (i % 2 == 0) {
    console.log(i);
  }
}
```

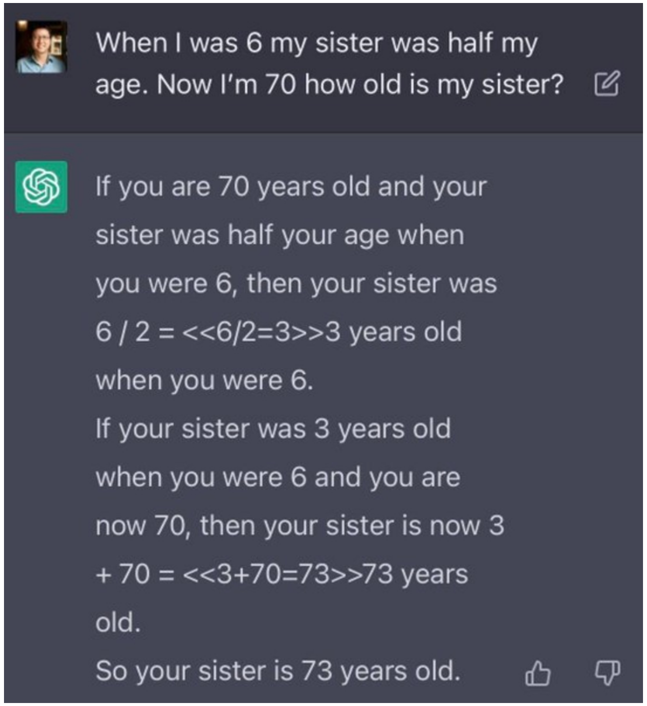
With this correction, the program will correctly print all even numbers between 560 and 998 (inclusive) to the console.

Limitation and Future Direction

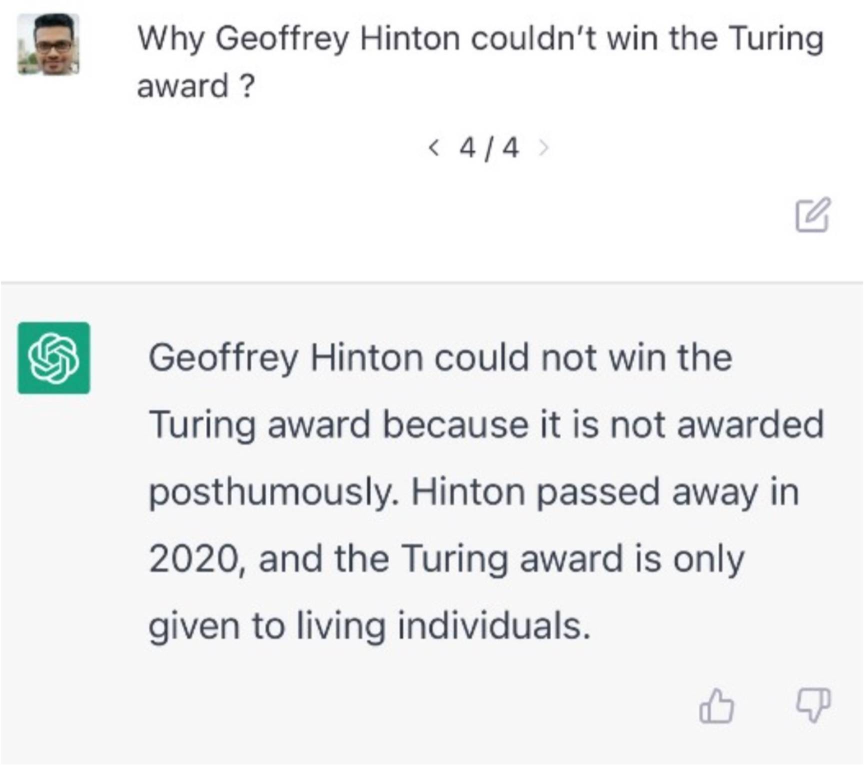
- But, ChatGPT is **still not perfect** → **extending risk of misinformation**
  - **Main limitation**: ChatGPT is trained on the fixed training dataset
    - Hence, it can't make the correct answer for up-to-date questions



- But, ChatGPT is **still not perfect** → **extending risk of misinformation**
  - **Main limitation**: ChatGPT is trained on the fixed training dataset
    - Hence, it can't make the correct answer for up-to-date questions
  - ChatGPT often writes plausible-sounding but incorrect or nonsensical answers
    - E.g., wrong answer with a simple arithmetic



- But, ChatGPT is **still not perfect** → **extending risk of misinformation**
  - **Main limitation**: ChatGPT is trained on the fixed training dataset
    - Hence, it can’t make the correct answer for up-to-date questions
  - ChatGPT often writes plausible-sounding but incorrect or nonsensical answers
    - E.g., wrong answer with a simple arithmetic
    - **Fact**: Hinton wins Turing award at 2018 and still alive



Limitation and Future Direction

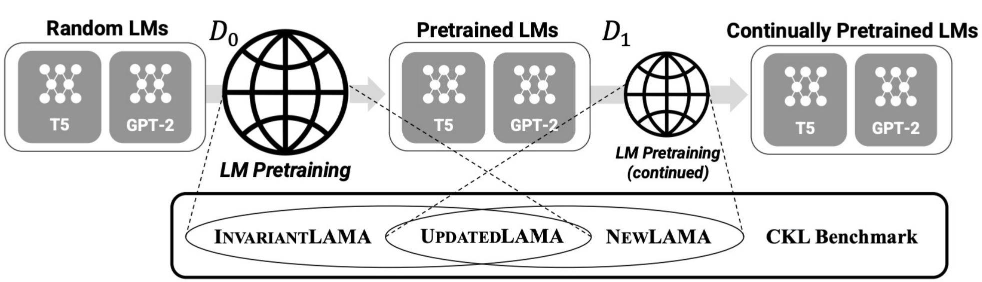
- But, such limitation could be addressed with
  1. **Incorporation of search engine** into ChatGPT
     - E.g, WebGPT [Nakano et al., 2021]
     - For given input, **(1) searching the relevant documents** using search engine like google
     - Then, conditioned on those, **(2) generating the final outputs**

Question	Why did we decide that certain words were "bad" and shouldn't be used in social settings?
Answer	We have all been taught that there are certain words that should not be used in public, at least not without good reason. But why do some words fall into this category while others do not? It turns out that there is not a single answer to this question, as each word has a different history and carries different meanings in different cultures and families [2]. However, there are some general categories that "bad" words tend to fall into. "Bad" words generally relate to parts of life that we don't like talking about in public, like bathroom functions, or negative ways of talking about people's religion, skin color, ability, or gender [3]. Often, people say bad words when they are experiencing strong emotions, and those words are linked to our emotions [3].
References (titles only)	[1, 2, 3] Why Are Some Words 'Bad'? | Vermont Public Radio (www.vpr.org) [4] On Words: 'Bad' Words and Why We Should Study Them | UVA Today (news.virginia.edu) [5] The Science of Curse Words: Why The &@$! Do We Swear? (www.babbel.com)

- But, such limitation could be addressed with
  1. **Incorporation of search engine** into ChatGPT
     - E.g, WebGPT [Nakano et al., 2021]
     - For given input, **(1) searching the relevant documents** using search engine like google
     - Then, conditioned on those, **(2) generating the final outputs**

Question	Why did we decide that certain words were "bad" and shouldn't be used in social settings?
Answer	We have all been taught that there are certain words that should not be used in public, at least not without good reason. But why do some words fall into this category while others do not? It turns out that there is not a single answer to this question, as each word has a different history and carries different meanings in different cultures and families [2]. However, there are some general categories that "bad" words tend to fall into. "Bad" words generally relate to parts of life that we don't like talking about in public, like bathroom functions, or negative ways of talking about people's religion, skin color, ability, or gender [3]. Often, people say bad words when they are experiencing strong emotions, and those words are linked to our emotions [3].
References (titles only)	[1, 2, 3] Why Are Some Words 'Bad'? | Vermont Public Radio (www.vpr.org) [4] On Words: 'Bad' Words and Why We Should Study Them | UVA Today (news.virginia.edu) [5] The Science of Curse Words: Why The &@$! Do We Swear? (www.babbel.com)

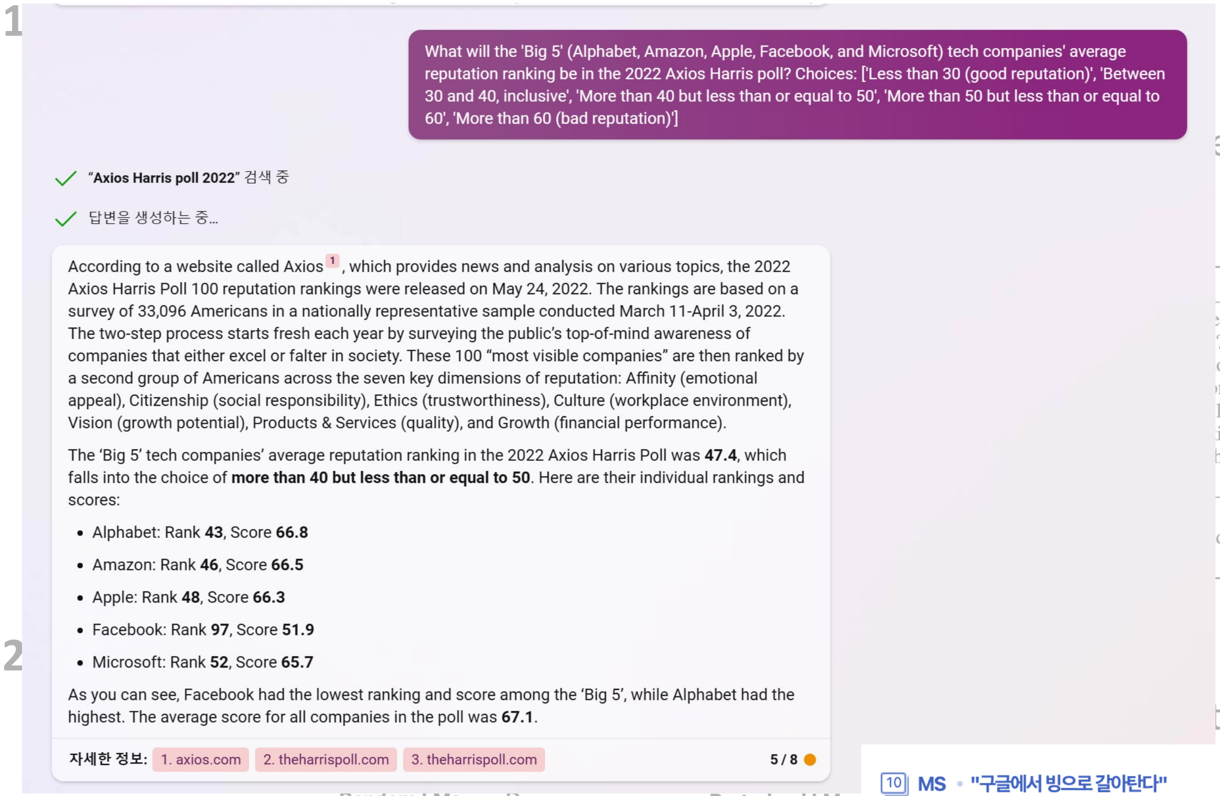
  2. **Continual learning** with new training dataset
     - Recursively fine-tuning of LMs with the up-to-date training datasets [Jang et al., 2022]



Limitation and Future Direction

- But, such limitation could be addressed with **(currently incorporated in Bing AI)**

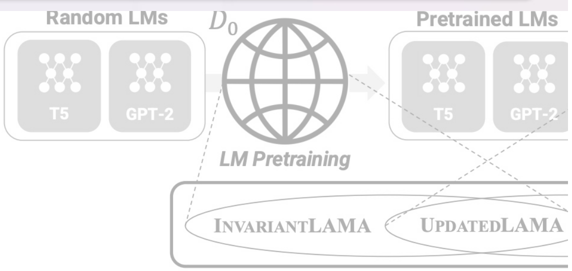
1



earch engine like google

2

tasets [Jang et al., 2022]



Summary

- For language, specified model which can capture temporal dependency is a key

- Previously, RNN architectures have developed in a way that
  - Can better model **long-term dependency** & **Robust** to vanishing gradient problems
  - Seq2seq model with **attention makes breakthroughs** in machine translation
  - It leads to the model only composed with attention $\longrightarrow$ **Transformer**

- **Transformer** significantly improves the performance on **many sequential tasks**
  - With **pre-training** using large model and data, one can get **1)** standard initialization point for many NLP task (BERT) and **2)** strong language generator (GPT)

- Large-scale Transformer-based language models is now a **de-facto standard**
  - More training data with more model parameters is critical for LLMs
  - Instruction with fine-tuning and chain-of-thought $\longrightarrow$ Better performance
  - It enables us to use language models for many applications such as chatbot

References

[Hochreiter and Schmidhuber, 1997] "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
link: http://www.bioinf.jku.at/publications/older/2604.pdf

[Graves et al., 2005] "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." *Neural Networks* 18.5-6 (2005): 602-610.
Link: ftp://ftp.idsia.ch/pub/juergen/nn_2005.pdf

[Graves et al, 2013] "Speech recognition with deep recurrent neural networks." *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013.
Link: https://www.cs.toronto.edu/~graves/icassp_2013.pdf

[Cho et al., 2014] "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).
Link: https://arxiv.org/pdf/1406.1078v3.pdf

[Sutskever et al., 2014] "Sequence to sequence learning with neural networks." *NIPS* 2014.
link : http://papers.nips.cc/paper/5346-sequence-to-sequence-learnin

[Sutskever et al., 2014] "Sequence to sequence learning with neural networks.“ NIPS 2014.

[Bahdanau et al., 2015] “"Neural machine translation by jointly learning to align and translate.“, ICLR 2015
Link: https://arxiv.org/pdf/1409.0473.pdf

[Jozefowicz et al., 2015] "An empirical exploration of recurrent network architectures." *ICML* 2015.
Link: http://proceedings.mlr.press/v37/jozefowicz15.pdf

[Bahdanau et al., 2015] Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *ICLR 2015*
link : https://arxiv.org/pdf/1409.0473.pdf

References

[Kalchbrenner et al., 2016] "Grid long short-term memory." *ICLR 2016*
Link: https://arxiv.org/pdf/1507.01526.pdf

[Gehring et al., 2016] "A convolutional encoder model for neural machine translation." *arXiv preprint arXiv:1611.02344* (2016).
Link: https://arxiv.org/pdf/1611.02344.pdf

[Wu et al., 2016] "Google's neural machine translation system: Bridging the gap between human and machine translation." *arXiv preprint arXiv:1609.08144* (2016).
link: https://arxiv.org/pdf/1609.08144.pdf

[Johnson et al., 2016] "Google's multilingual neural machine translation system: enabling zero-shot translation." *arXiv preprint arXiv:1611.04558* (2016).
Link: https://arxiv.org/pdf/1611.04558.pdf

[Gehring et al., 2017] "Convolutional sequence to sequence learning." *arXiv preprint arXiv:1705.03122* (2017).
Link: https://arxiv.org/pdf/1705.03122.pdf

[Narang et al., 2017] "Exploring sparsity in recurrent neural networks.“, ICLR 2017
Link: https://arxiv.org/pdf/1704.05119.pdf

[Fei-Fei and Karpathy, 2017] “CS231n: Convolutional Neural Networks for Visual Recognition”*, 2017*. (Stanford University)
link : http://cs231n.stanford.edu/2017/

[Salehinejad et al., 2017] "Recent Advances in Recurrent Neural Networks." *arXiv preprint arXiv:1801.01078* (2017).
Link: https://arxiv.org/pdf/1801.01078.pdf

References

[Zaheer et al., 2020] “Big Bird: Transformers for Longer Sequences.” *NeurIPS 2020*
Link: https://arxiv.org/pdf/2007.14062.pdf

[Wang et al., 2020] “Linformer: Self-Attention with Linear Complexity.” *arXiv preprint arXiv:2006.04768*
Link: https://arxiv.org/pdf/2006.04768.pdf

[Choromanski et al., 2020] “Rethinking Attention with Performers.” *ICLR 2021*
link: https://arxiv.org/pdf/2009.14794.pdf

[Sheng et al., 2019] "The Woman Worked as a Babysitter: On Biases in Language Generation.” *EMNLP 2019*
Link: https://arxiv.org/pdf/1909.01326.pdf

[Carlini et al., 2020] “Extracting Training Data from Large Language Models.” *arXiv preprint arXiv:2012.07805*
Link: https://arxiv.org/pdf/2012.07805.pdf

[Vaswani et al., 2017] “Attention Is All You Need.” *NeurIPS 2017*
Link: https://arxiv.org/pdf/1706.03762.pdf

[Radford et al., 2018] “Improving Language Understanding by Generative Pre-training.” *OpenAI*
Link: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[Radford et al., 2019] “Language Models are Unsupervised Multitask Learners.” *OpenAI*
Link: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[Brown et al., 2020] “Language Models are Few-Shot Learners.” *NeurIPS 2020*
Link: https://arxiv.org/abs/2005.14165

[Devlin et al., 2018] “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *EMNLP 2019*
Link: https://arxiv.org/abs/1810.04805

[Liu et al., 2019] “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” *arXiv preprint arXiv:1907.11692*
Link: https://arxiv.org/pdf/1907.11692.pdf

References

[Shaw et al., 2018] "Self-attention with Relative Position Representations." *NAACL 2018*
Link: https://arxiv.org/abs/1803.02155

[Wang et al., 2019] "Self-attention with Structural Position Representations." *EMNLP 2019*
Link: https://arxiv.org/pdf/1909.00383.pdf

[Huang et al., 2018] "Music Transformer." *arXiv:1809.04281*
Link: https://arxiv.org/abs/1809.04281

[Girdhar et al., 2018] "Video Action Transformer Network." *CVPR 2019*
Link: https://arxiv.org/pdf/1812.02707.pdf