Since 2004
UET
ĐẠI HỌC CÔNG NGHỆ, ĐHQGHN
VNU-University of Engineering and Technology

KEPCO KDN

Since 1906
VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

Large Language Model and Its Applications

LLM03: Natural Lanuage Processing & Large Language Models

Nguyen Van Vinh - UET

sponsored by **KEPCO KDN Co., Ltd.**
Eco-friendly & Digital Centered Energy ICT Platform Leader

Hanoi, 09/2023

Outline

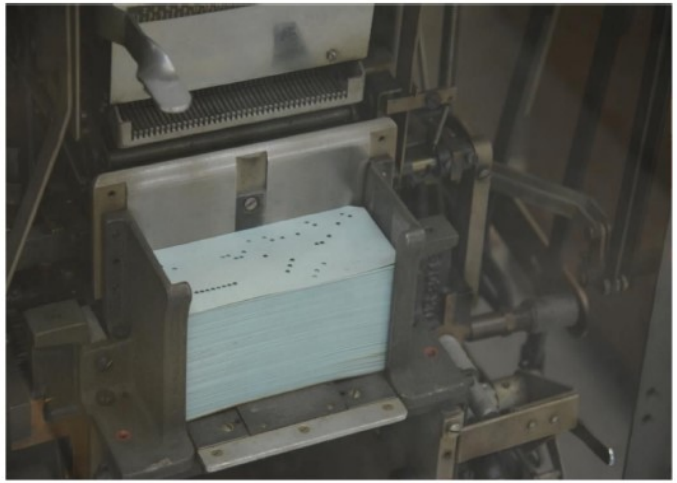
- Introduction to Natural Langage Processing (NLP)
- Word Embeddings & Attention & Transfomer model
- Large Language Models

NLP market

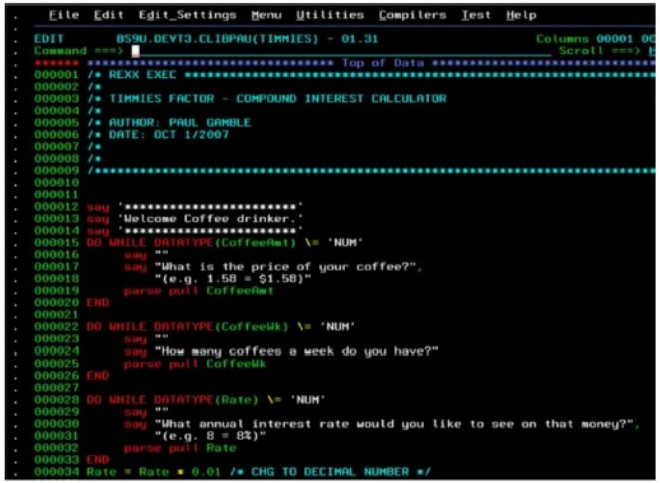
- The Americas Natural Language Processing Market size was estimated at USD 5,985.64 million in 2021, is expected to reach USD 6,911.43 million in 2022, and is projected to grow at a CAGR of 16.44% to reach USD 14,924.84 million by 2027.
- The Asia-Pacific Natural Language Processing Market size was estimated at USD 4,842.69 million in 2021, is expected to reach USD 5,705.64 million in 2022, and is projected to grow at a CAGR of 17.22% to reach USD 12,563.54 million by 2027.
- The Europe, Middle East & Africa Natural Language Processing Market size was estimated at USD 5,248.39 million in 2021, is expected to reach USD 6,110.70 million in 2022, and is projected to grow at a CAGR of 16.76% to reach USD 13,303.33 million by 2027.

Source: https://www.businesswire.com/news/home/20220816005515/en/Natural-Language-Processing-NLP-Market-Intelligence-Report---Global-Forecast-to-2027---ResearchAndMarkets.com

Communication With Machines



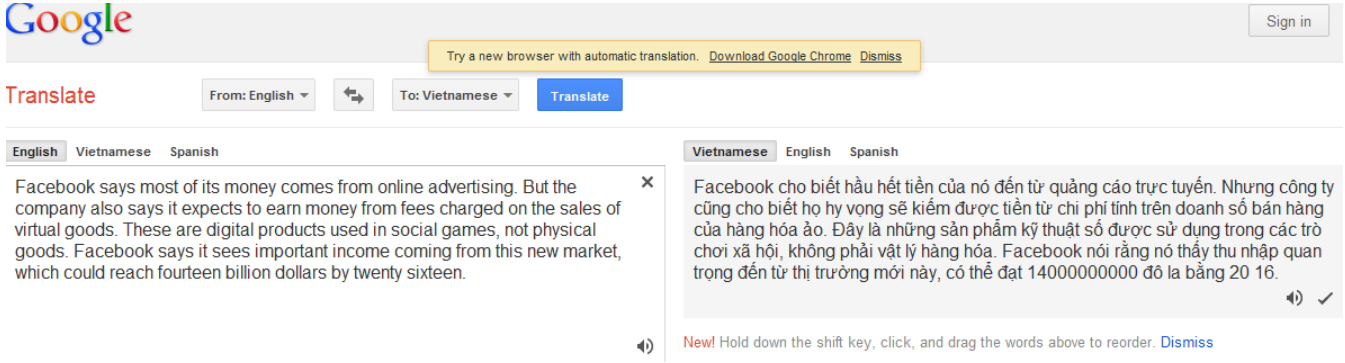
~50-70s

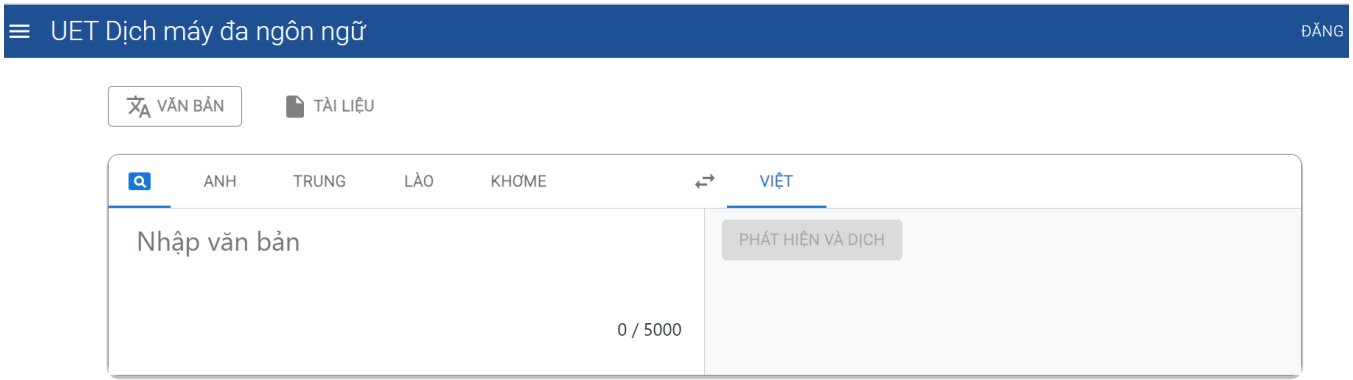


~80s

today

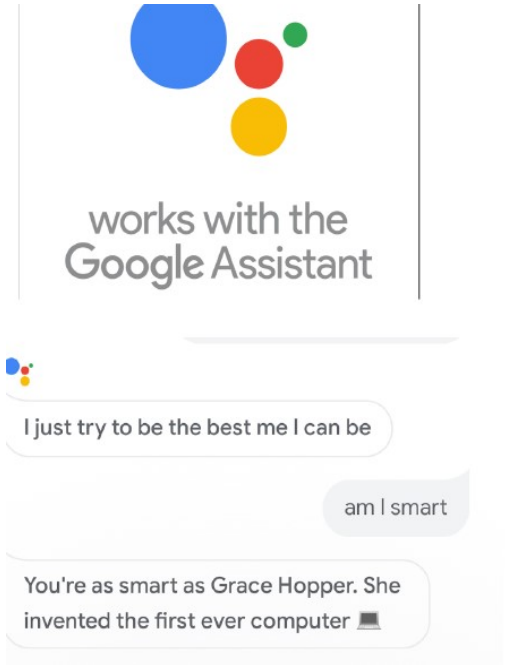
Google Translation & UET Translation





Virtual Assistant

- **Conversational agents contain:**
  - Speech recognition
  - Language analysis
  - Dialogue processing
  - Information retrieval
  - Text to speech
- **Google now, Alexa, Siri, Cortana, Watson, ChatGPT, …**



ChatGPT (AI hotest topic)

- **ChatGPT** (**Chat Generative Pre-trained Transformer**) is a chatbot launched by OpenAI in **November 2022**
- **Generative Pre-trained Transformer 3** (**GPT-3**) is an autoregressive language model that uses deep learning to produce human-like text. Given an initial text as prompt, it will produce text that continues the prompt.
- The architecture is a **standard transformer network** (with a few engineering tweaks) with the (back then) unprecedented size of 2048-token-long context and **175 billion parameters** (requiring 800 GB of storage).
- ChatGPT can fluently answer all the questions that users ask, any domains. Besides, ChatGPT can also write code, write poetry, compose music, write letters (documents), design and even fix errors in programming.
- ChatGPT has surpassed **10 million users just 40 days** after its official launch, breaking all previous records (100 million just 2 months).

NLP Careers: So hot!

- Industry (VinAI, Vin BigData, Viettel, Fpt, …)
- Government
- Academia

Source: https://daring.my.id/

Natural Language processing

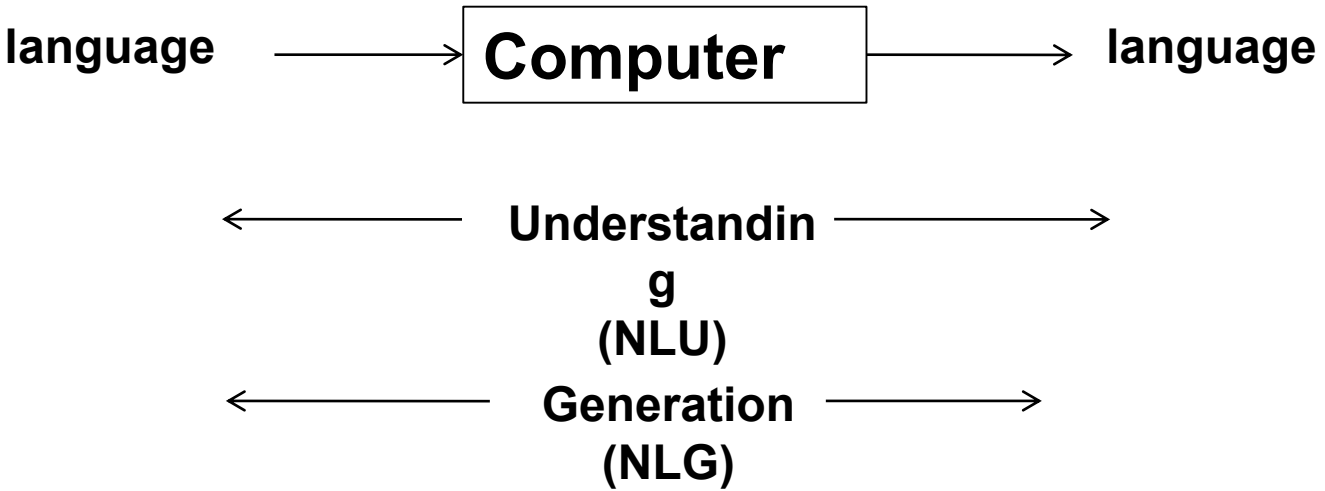
- NLP = building **computer programs** to analyze, understand and generate **human language - either spoken or written** (informal)
- NLP is an interdisciplinary field

What is NLP?

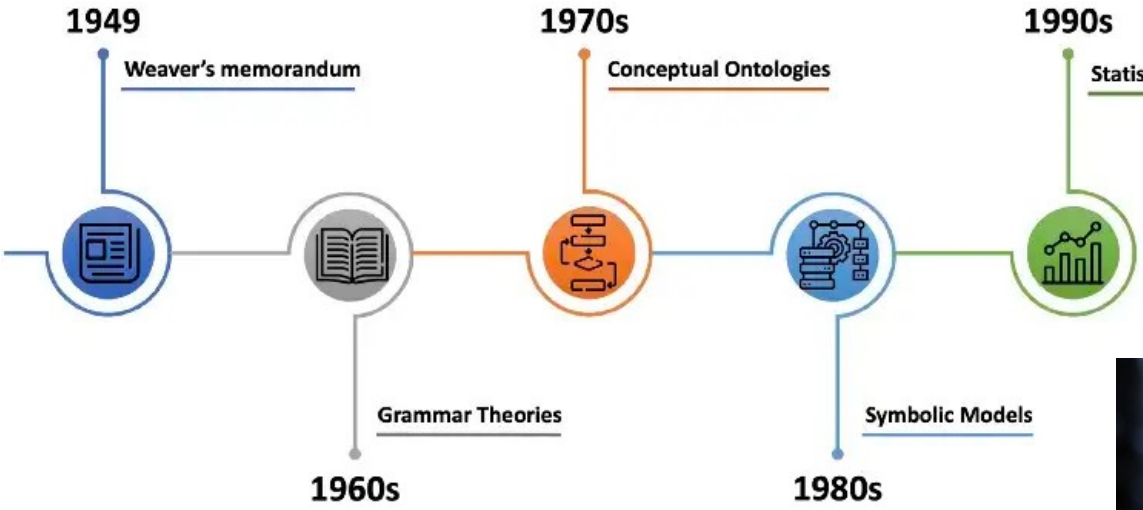
- **Natural language processing** (**NLP**) is a subfield of artificial intelligence and computational linguistics. It studies the problems of automated generation and understanding of natural human languages.
- **Natural-language-generation systems** convert information from computer databases into normal-sounding human language. **Natural-language-understanding systems** convert samples of human language into more formal representations that are easier for **computer** programs to manipulate.

What is Natural Language Processing?

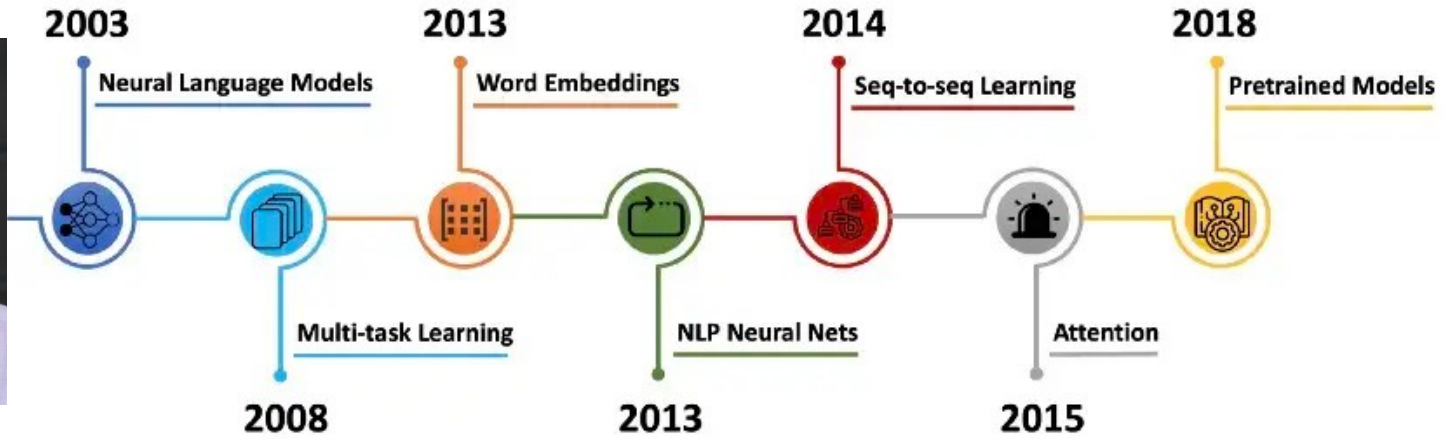
- Computers using natural language as input and/or output



A brief history of NLP



A brief history of NLP



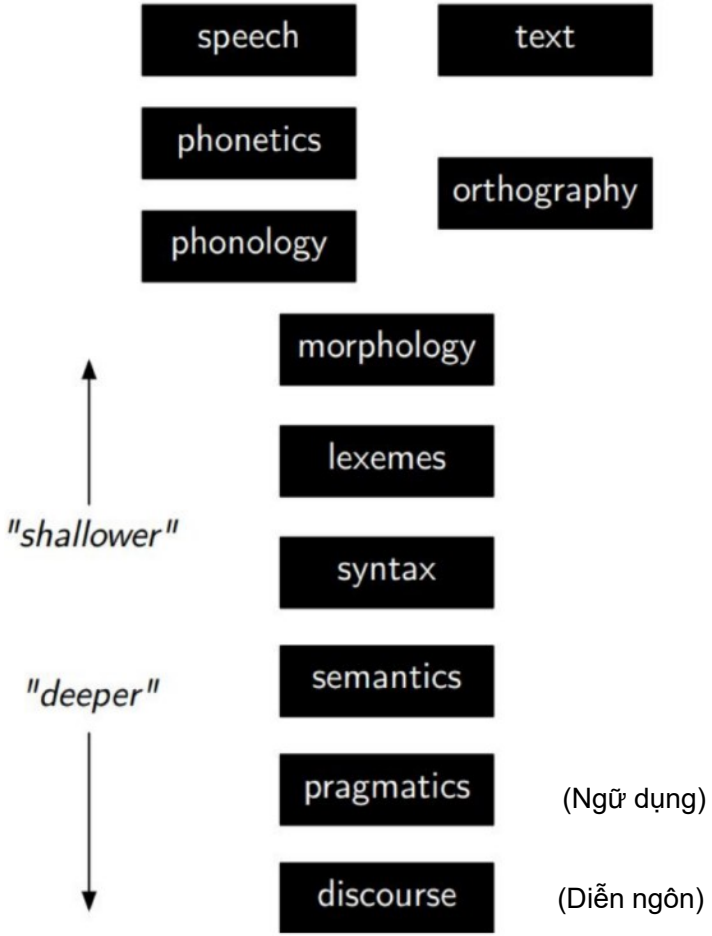
- **2018:** BERT **2019:** T5, RoBERTa **2020:** GPT-3 **2022:** ChatGPT

Natural language processing and Computational linguistics

- **Natural language processing (NLP)** develops methods for solving practical problems involving language:
  - Automatic speech recognition
  - Machine Translation
  - Sentiment Analysis
  - Information extraction from documents

- **Computational linguistics (CL)** focused on using technology to support/implement linguistics:
  - how do we understand language?
  - how do we produce language?
  - how do we learn language?

Level Of Linguistic Knowledge



Factors Changing NLP Landscape

- Increases in computing power
- The rise of the web, then the social web
- Advances in machine learning
- Advances in understanding of language in social context

Natural Language Processing

- **Applications**
  - Machine Translation
  - Information Retrieval
  - Question Answering
  - Dialogue Systems
  - Information Extraction
  - Summarization
  - Sentiment Analysis
  - …

- **Core Technologies (NLP sub-problems)**
  - Language modeling
  - Part-of-speech tagging
  - Syntactic parsing
  - Named-entity recognition
  - Word sense disambiguation
  - Semantic role labeling
  - …

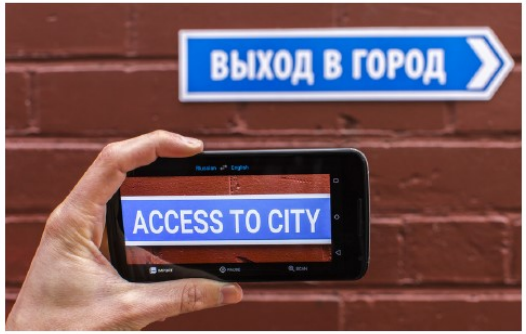
NLP lies at the intersection of computational linguistics and machine learning.

The era of deep learning

- Significant advances in core NLP technologies
- Essential ingredient: large-scale supervision, lots of compute
- Reduced manual effort - less/zero **feature engineering**

GPU

TPU



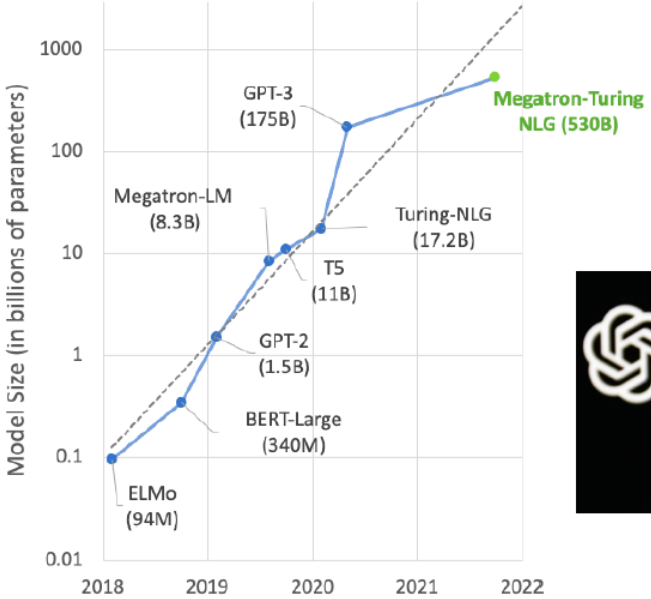
36M sentence pairs

Russian: Машинный перевод - это круто!

English: Machine translation is cool!

The era of pre-training / LLMs

BERT, ELMo, ERNIE…



- Leverages a lot of unlabeled text
- Model size increased by $10^3 - 10^5$x in parameters

ChatGPT examples

DA

What NLP topics should I teach in 2023?

Here are some NLP topics that are popular and relevant in 2023:

1. Transformers (BERT, GPT, etc.)
2. Transfer Learning in NLP
3. Multi-lingual NLP
4. Attention Mechanisms
5. Adversarial Training in NLP
6. NLP for Question Answering
7. Sentiment Analysis
8. Text Generation (summarization, translation, etc.)
9. Named Entity Recognition
10. Dialogue Systems

Note: The field of NLP is constantly evolving, so it's important to stay up-to-date with recent advances and adjust your teaching content accordingly.

Why is NLP difficult?

- Ambiguity
- Sparsity
- Training data for supervised learning is often not available

→ Unsupervised/semi-supervised techniques for training from available data

- Algorithmic challenges
  - vocabulary can be large (e.g., 50K words)
  - data sets are often large (GB or TB)

Ambiguity ???

“At last, a computer that understands you like your mother”

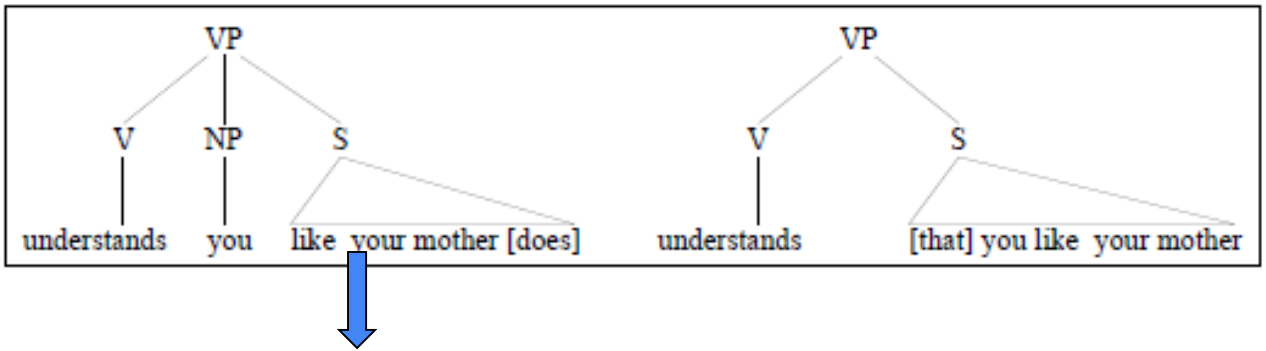
“Ông già đi nhanh quá”

Ambiguity

- “**At last, a computer that understands you like your mother**”
- It understands you as well as your mother understands you
- It understands (that) you like your mother
- It understands you as well as it understands your mother

Ambiguity at Many Levels

- At the **syntactic** level:



Different structures lead to different interpretations

Ambiguity at Many Levels

- At the **semantic** (meaning) level:
  - Two definitions of “bank”
    - an organization where people and businesses can invest or borrow money, change it to foreign money, etc., or a building where these services are offered
    - sloping raised land, especially along the sides of a river
- This is an instance of **word sense ambiguity**

Corpora

- A corpus is a collection of text
  - Often annotated in some way
  - Sometimes just lots of text
- Examples
  - Penn Treebank: 1M words of parsed WSJ
  - Canadian Hansards: 10M+ words of French/English sentences
  - Yelp reviews
  - VLSP Corpus (Vietnamese)

Sparsity

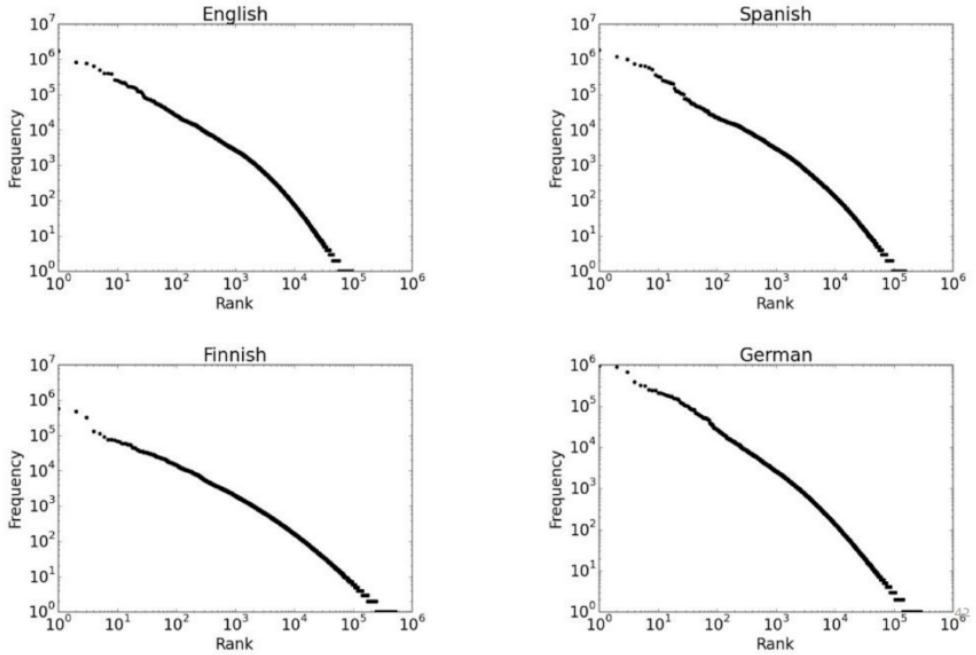
- Sparse data due to Zipf's Law
- Example: the frequency of different words in a large text corpus

any word	
Frequency	Token
1,698,599	the
849,256	of
793,731	to
640,257	and
508,560	in
407,638	that
400,467	is
394,778	a
263,040	I

nouns	
Frequency	Token
124,598	European
104,325	Mr
92,195	Commission
66,781	President
62,867	Parliament
57,804	Union
53,683	report
53,547	Council
45,842	States

Sparsity

- Regardless of how large our corpus is, there will be a lot of infrequent words



Word Embeddings & Attention & Transformer

Word Embeddings

= **Learned** representations from text for representing words

- Input: a large text corpora, *V*, *d*
  - V: a pre-defined vocabulary
  - d: dimension of word vectors (e.g. 300)
  - Text corpora:
    - Wikipedia + Gigaword 5: 6B tokens
    - Twitter: 27B tokens
    - Common Crawl: 840B tokens
- Output: $f : V \rightarrow \mathbb{R}^d$

$$v_{\text{cat}} = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix} \quad v_{\text{dog}} = \begin{pmatrix} -0.124 \\ 0.430 \\ -0.200 \\ 0.329 \end{pmatrix}$$

$$v_{\text{the}} = \begin{pmatrix} 0.234 \\ 0.266 \\ 0.239 \\ -0.199 \end{pmatrix} \quad v_{\text{language}} = \begin{pmatrix} 0.290 \\ -0.441 \\ 0.762 \\ 0.982 \end{pmatrix}$$

Each word is represented by a low-dimensional (e.g., d = 300), real-valued vector

Each coordinate/dimension of the vector doesn't have a particular interpretation

Word Embeddings

- Basic property: similar words have similar vectors

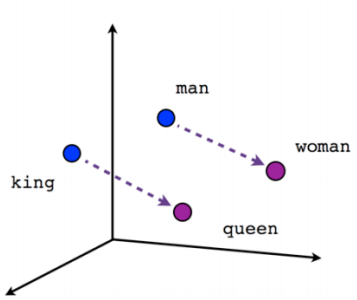
word w^*= “sweden”

$$\arg\max_{w \in V} \cos(e(w), e(w^*))$$

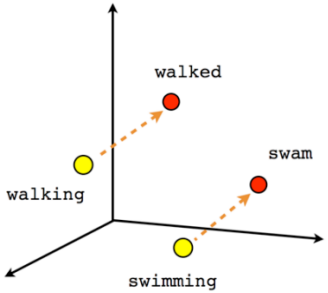
Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Word embeddings

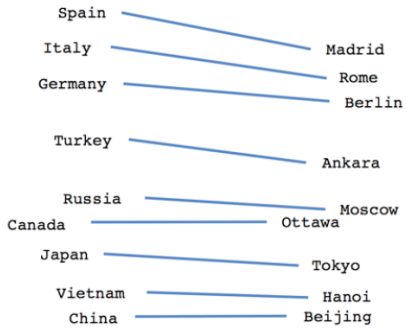
- They have some other nice properties too!



Male-Female



Verb tense



Country-Capital

$$v_{\text{man}} - v_{\text{woman}} \approx v_{\text{king}} - v_{\text{queen}}$$

$$v_{\text{Paris}} - v_{\text{France}} \approx v_{\text{Rome}} - v_{\text{Italy}}$$

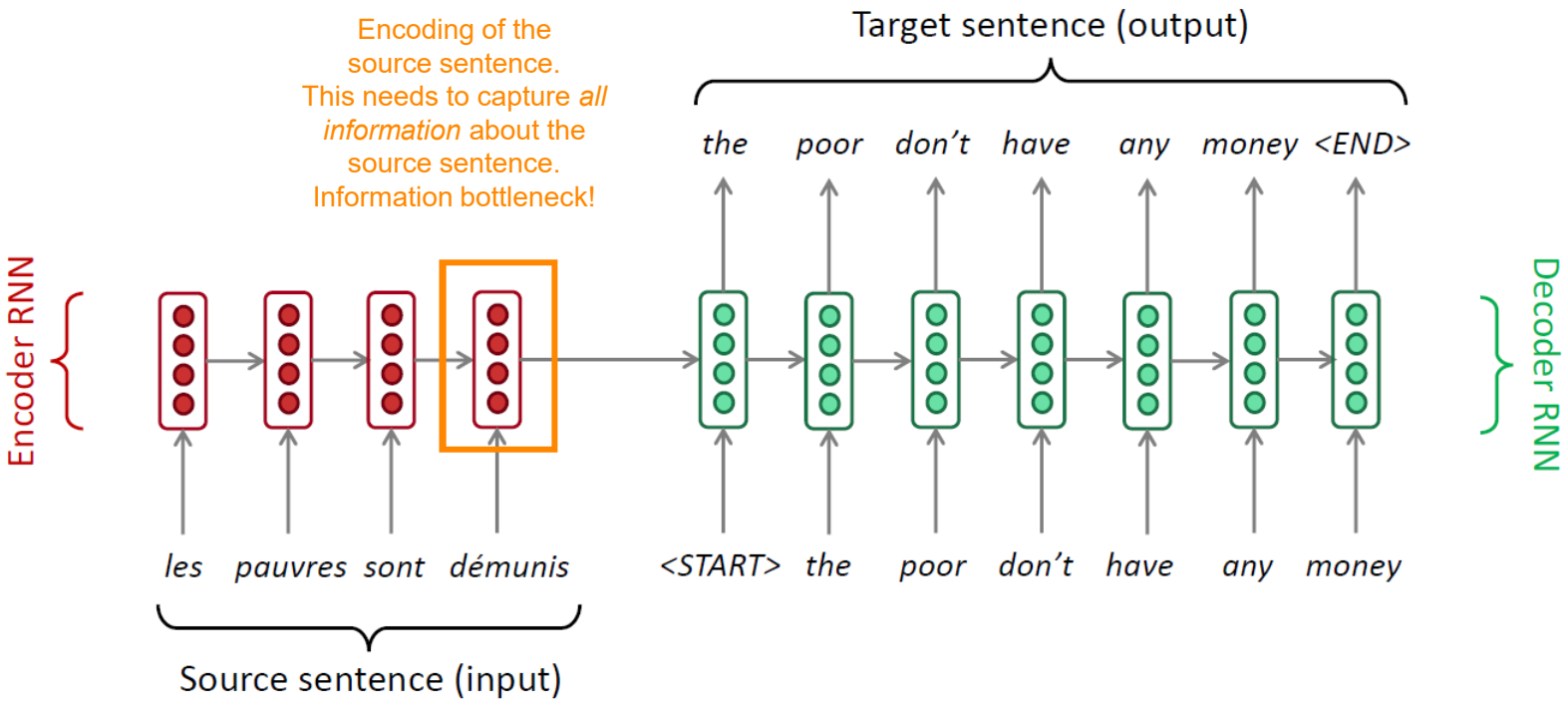
Word analogy test: $a : a^* :: b : b^*$

$$b^* = \arg\max_{w \in V} \cos(e(w), e(a^*) - e(a) + e(b))$$

Trained word embeddings

- word2vec: https://code.google.com/archive/p/word2vec/
- GloVe: https://nlp.stanford.edu/projects/glove/
- FastText: https://fasttext.cc/

Sequence-to-sequence: bottleneck problem



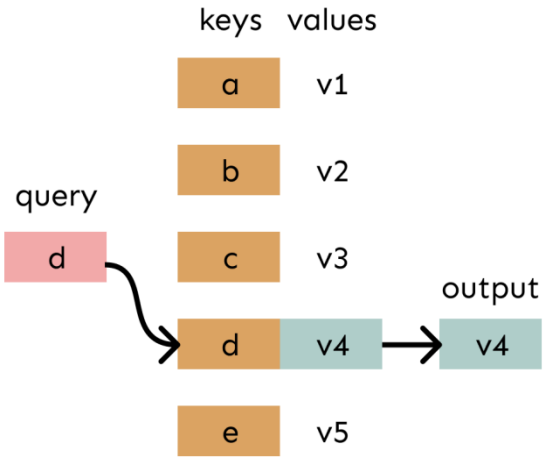
Attention

- **Attention** provides a solution to the bottleneck problem.
- **Core idea:** on each step of the decoder, *focus on a particular part* of the source sequence

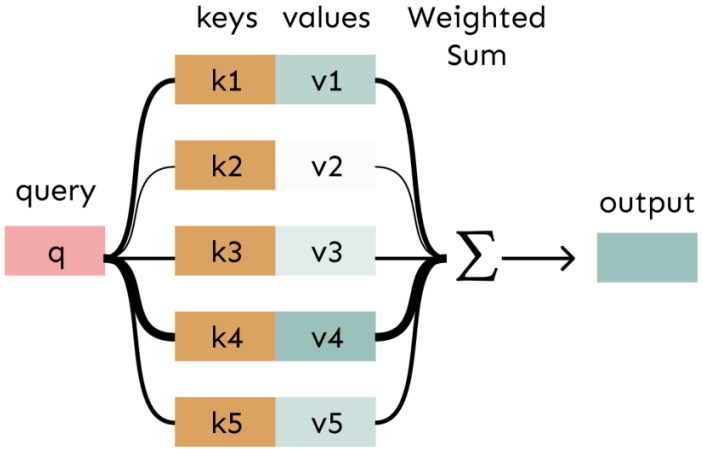
Attention as a soft, averaging lookup table

- We can think of **attention** as performing fuzzy lookup a in **key-value store**

Lookup table: a table of keys that map to values. The query matches one of the keys, returning its value.



Attention: The query matches to all keys softly to a weight between 0 and 1. The keys' values are multipled by the weights and summed.



Problems with RNNs = Motivation for Transformers

- Recurrent models typically factor computation along the symbol positions of the input and output sequences
  - Sequential computation prevents parallelization
  - Critical at longer sequence lengths, as memory constraints limit batching across examples

- Despite advanced RNNs like LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length for codependent computation between states grows with sequence

- **But if attention gives us access to any state… maybe we don't need the RNN???**

Transformers (2017)

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

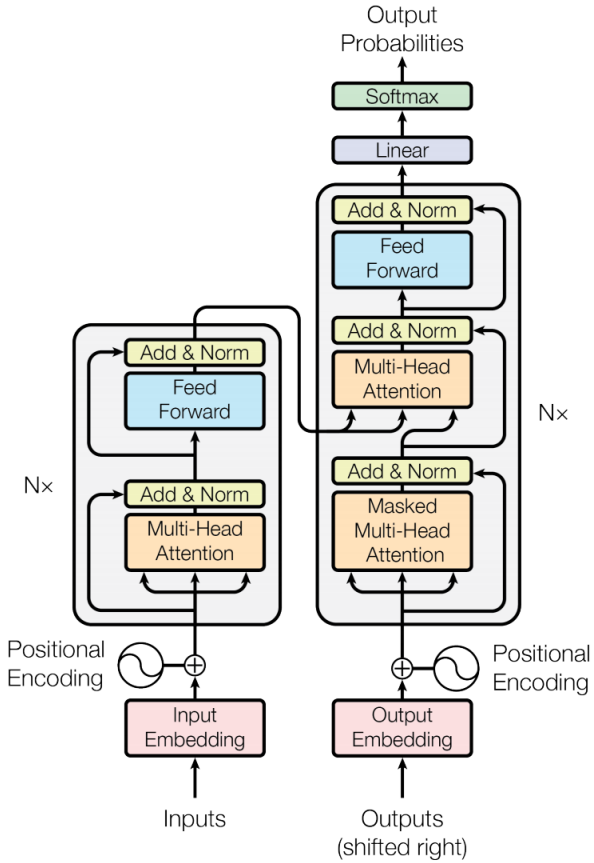
Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

(Vaswani et al., 2017)

Transformer Overview

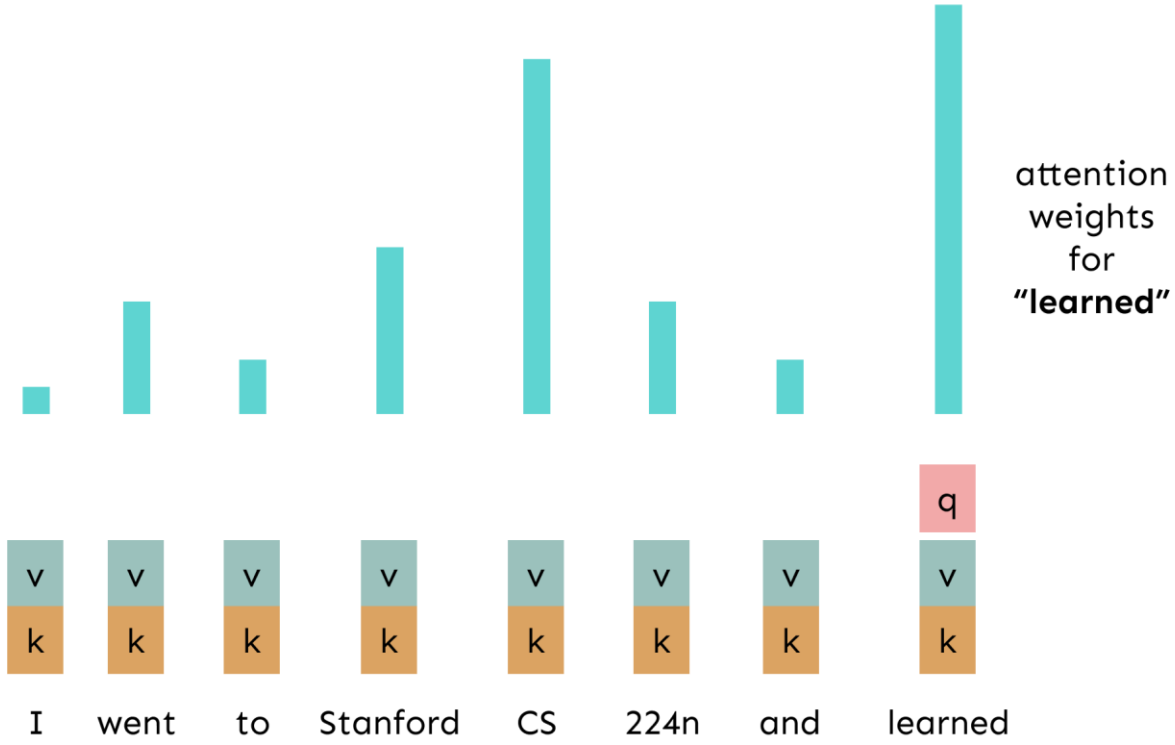
- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier



Transformer Basics

- Let’s define the basic building blocks of transformer networks first: new attention layers!

Seft-attention Example



Self-Attention: keys, queries, values from the same

Let $\boldsymbol{w}_{1:n}$ be a sequence of words in vocabulary V, like *Zuko made his uncle tea*.

For each $\boldsymbol{w}_i$, let $\boldsymbol{x}_i = E\boldsymbol{w}_i$, where $E \in \mathbb{R}^{d \times |V|}$ is an embedding matrix.

1. Transform each word embedding with weight matrices Q, K, V, each in $\mathbb{R}^{d \times d}$

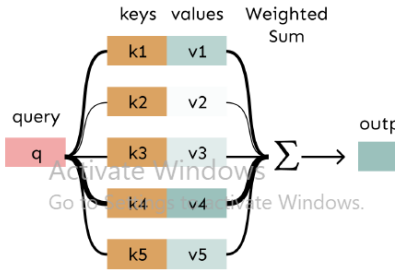
$\boldsymbol{q}_i = Q\boldsymbol{x}_i$ (queries) $\quad \boldsymbol{k}_i = K\boldsymbol{x}_i$ (keys) $\quad \boldsymbol{v}_i = V\boldsymbol{x}_i$ (values)

2. Compute pairwise similarities between keys and queries; normalize with softmax

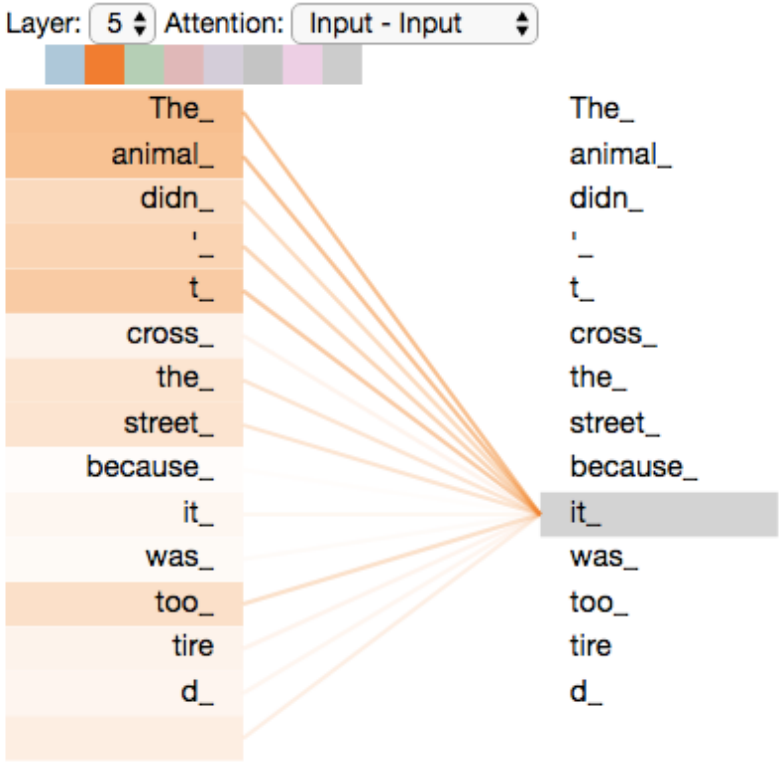
$$\boldsymbol{e}_{ij} = \boldsymbol{q}_i^\top \boldsymbol{k}_j \qquad \boldsymbol{\alpha}_{ij} = \frac{\exp(\boldsymbol{e}_{ij})}{\sum_{j'} \exp(\boldsymbol{e}_{ij'})}$$

3. Compute output for each word as weighted sum of values

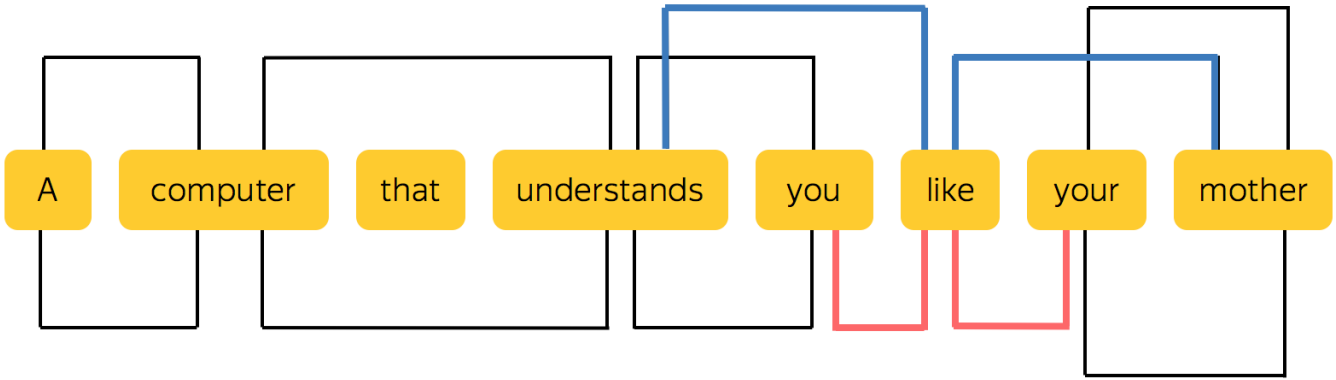
$$\boldsymbol{o}_i = \sum_j \boldsymbol{\alpha}_{ij} \boldsymbol{v}_i$$



Self Attention Visualization



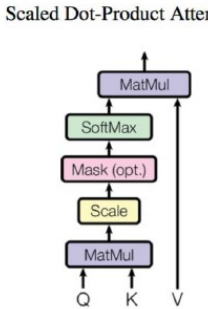
Self-Attention

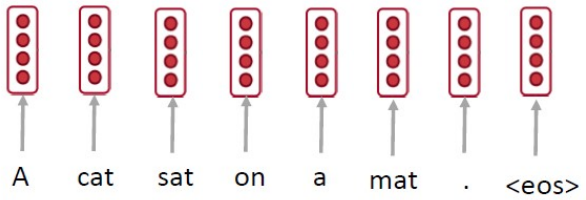


(example and picture from David Talbot)

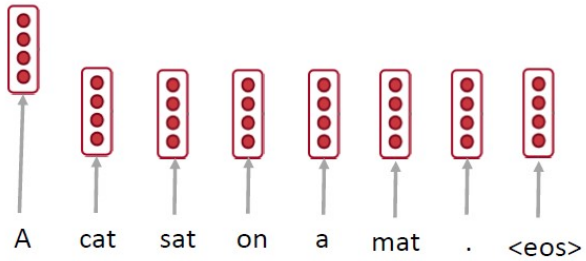
Self-attention: A Running Example

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

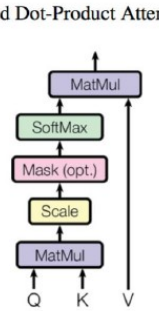




Self-attention: A Running Example

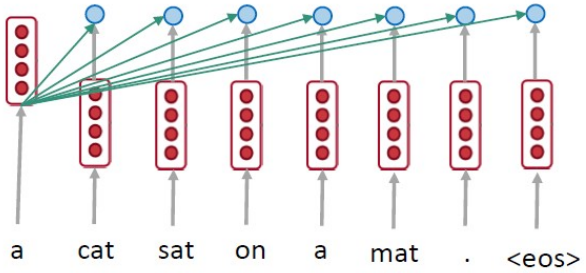


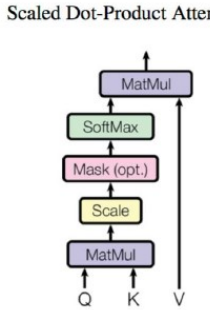
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



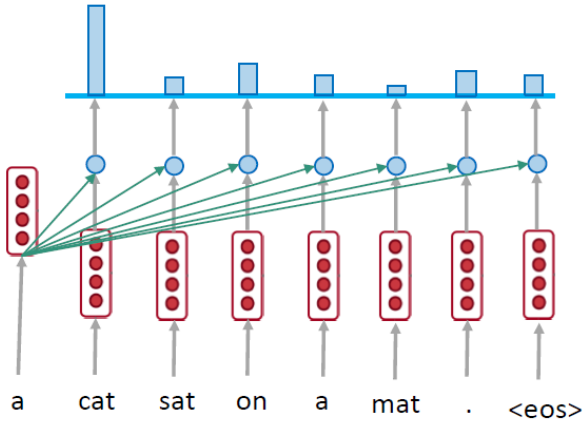
Self-attention: A Running Example

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

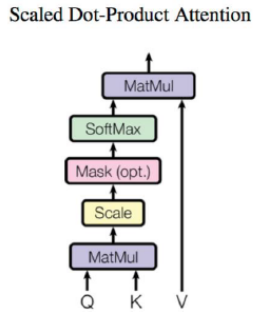




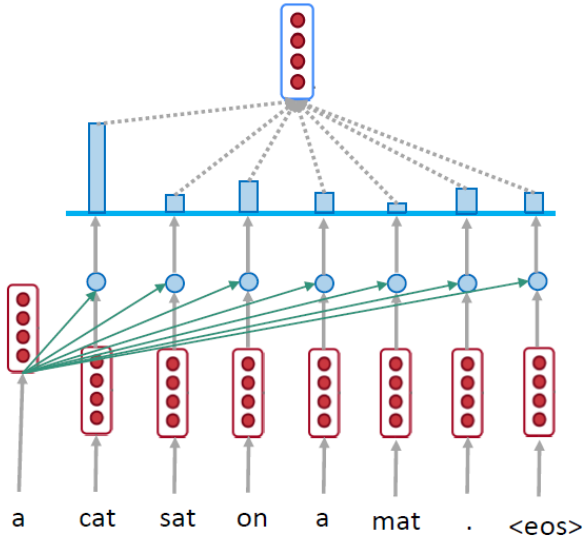
Self-attention: A Running Example



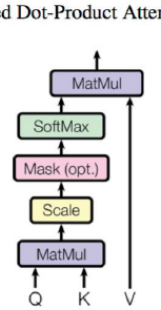
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



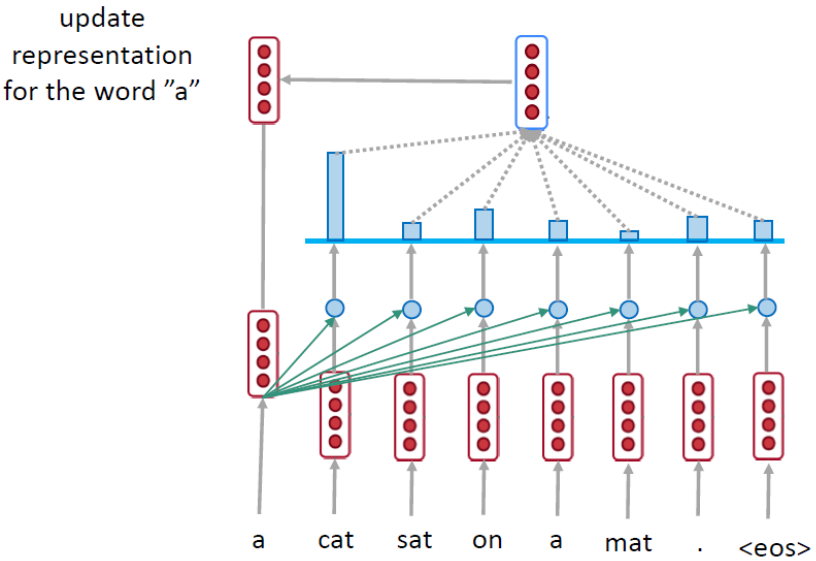
Self-attention: A Running Example



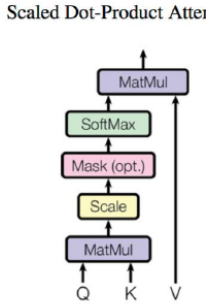
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



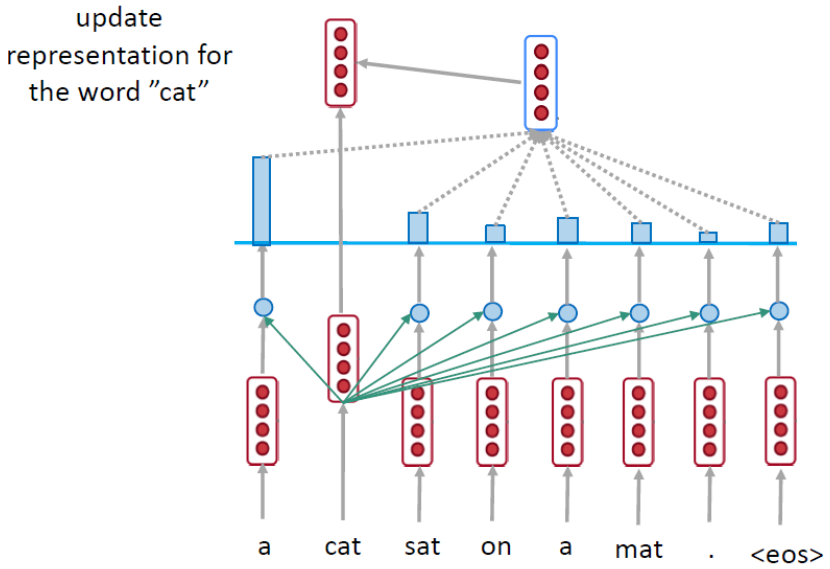
Self-attention: A Running Example



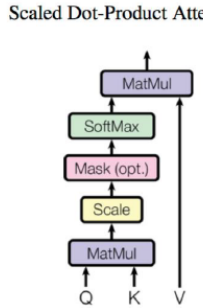
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



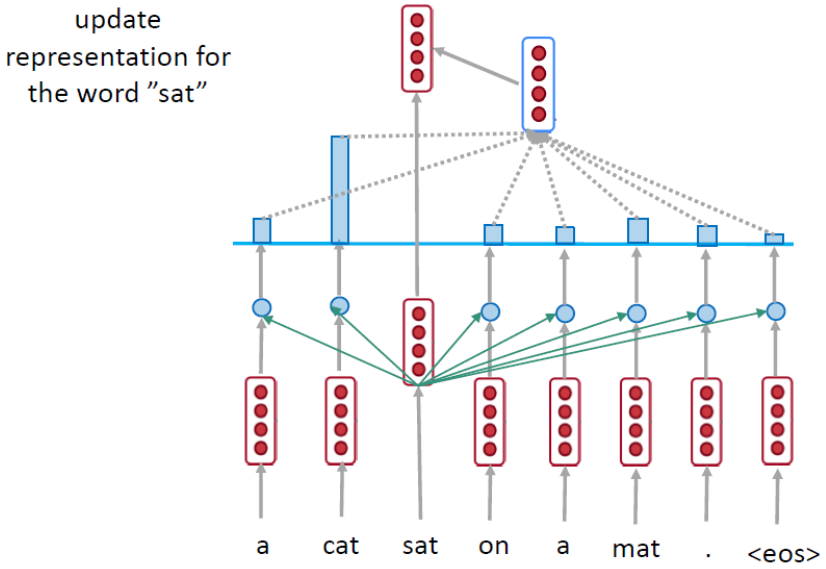
Self-attention: A Running Example



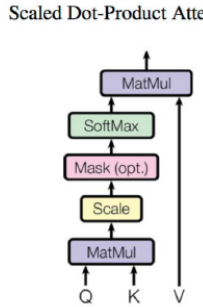
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



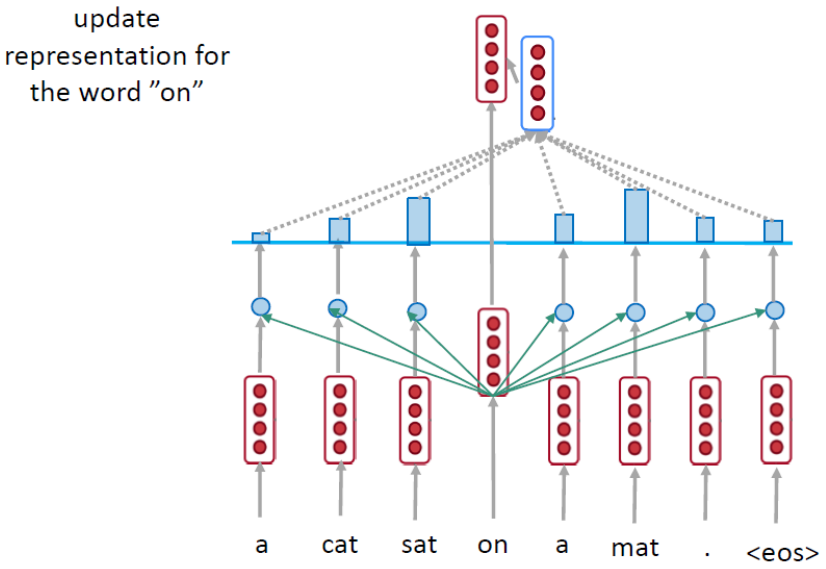
Self-attention: A Running Example



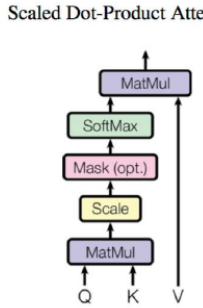
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Self-attention: A Running Example

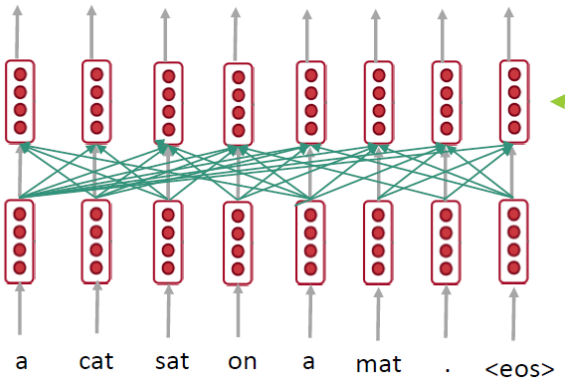


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Self-attention: A Running Example

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

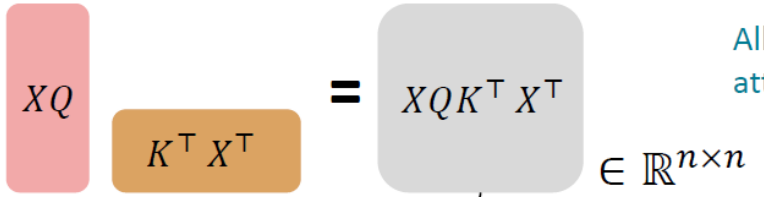


Updated representations
for each word
(one layer)

Sequence-stacked form of Attention

- Let's look at how key-query-value attention is computed, in matrices.
  - Let $X = [x_1; \ldots; x_n] \in \mathbb{R}^{n \times d}$ be the concatenation of input vectors.
  - First, note that $XK \in \mathbb{R}^{n \times d}$, $XQ \in \mathbb{R}^{n \times d}$, $XV \in \mathbb{R}^{n \times d}$.
  - The output is defined as $\text{output} = \text{softmax}(XQ(XK)^\top)XV \in\in \mathbb{R}^{n \times d}$.

First, take the query-key dot products in one matrix multiplication: $XQ(XK)^\top$



Next, softmax, and compute the weighted average with another matrix multiplication.

The Transformer Encoder:
Key-Query-Value Attention

- We saw that self-attention is when keys, queries, and values come from the same source. The Transformer does this in a particular way:
  - Let $x_1, \ldots, x_n$ be input vectors to the Transformer encoder; $x_i \in \mathbb{R}^d$

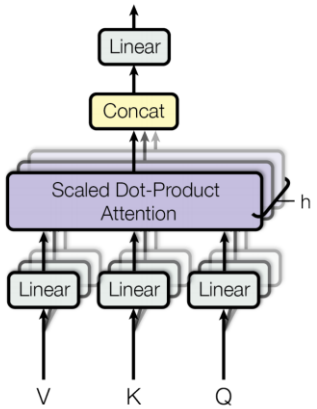
- Then keys, queries, values are:
  - $k_i = Kx_i$, where $K \in \mathbb{R}^{d \times d}$ is the key matrix.
  - $q_i = Qx_i$, where $Q \in \mathbb{R}^{d \times d}$ is the query matrix.
  - $v_i = Vx_i$, where $V \in \mathbb{R}^{d \times d}$ is the value matrix.

- These matrices allow *different aspects* of the x vectors to be used/emphasized in each of the three roles.

Multi-head attention

- The input word vectors could be the queries, keys and values
  - In other words: the word vectors themselves select each other
  - Word vector stack = Q = K = V
- **Problem:** Only one way for words to interact with one-another
- Solution: Multi-head attention
  - First map Q, K, V into h many lower dimensional spaces via W matrices
  - Then apply attention, then concatenate outputs and pipe through linear layer

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Scaled Dot-Product Attention

- **"Scaled Dot Product"** attention aids in training.
- When dimensionality d becomes large, dot products between vectors tend to become large.
  - Because of this, inputs to the softmax function can be large, making the gradients small.
- Instead of the self-attention function we've seen:

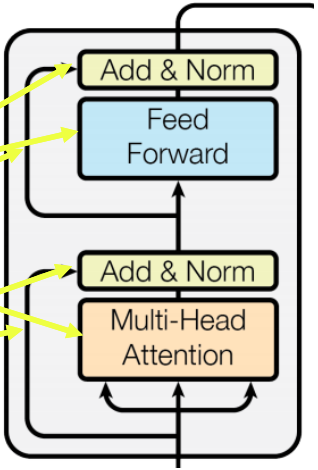
$$\text{output}_\ell = \text{softmax}\left(XQ_\ell K_\ell^\top X^\top\right) * XV_\ell$$

- We divide the attention scores by $\sqrt{d/h}$, to stop the scores from becoming large just as a function of d/h (The dimensionality divided by the number of heads.)

$$\text{output}_\ell = \text{softmax}\left(\frac{XQ_\ell K_\ell^\top X^\top}{\sqrt{d/h}}\right) * XV_\ell$$

Complete transformer block

- Each block has two “sublayers”
  - 1. Multihead attention
  - 2. layer feed-forward Nnet (with relu)



- Each of these two steps also has:
  - Residual (short-circuit) connection and LayerNorm:
  - LayerNorm(x + Sublayer(x))
    - Layernorm changes input to have mean 0 and variance 1, per layer and per training point (and adds two more parameters)

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

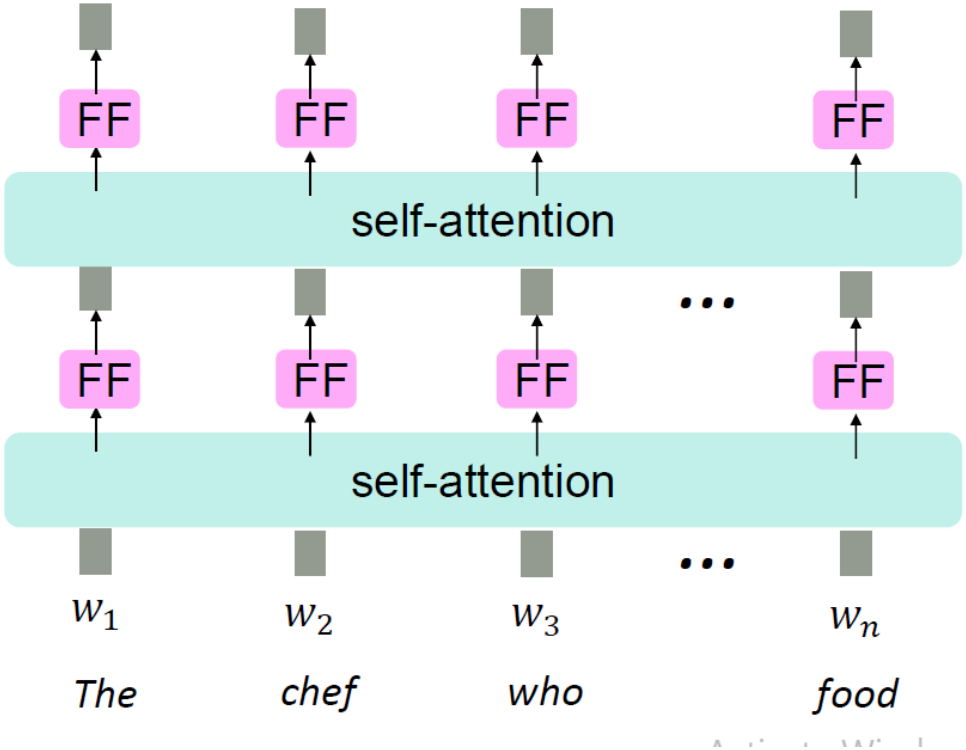
Layer Normalization by Ba, Kiros and Hinton, https://arxiv.org/pdf/1607.06450.pdf

Adding nonlinearities in self-attention

- Note that there are no elementwise nonlinearities in self-attention; stacking more self-attention layers just re-averages **value** vectors (Why? Look at the notes!)

- Easy fix: add a **feed-forward network** to post-process each output vector.

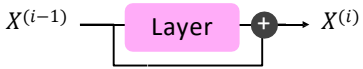
$$m_i = MLP(\text{output}_i)$$
$$= W_2 * \text{ReLU}(W_1\, \text{output}_i + b_1) + b_2$$



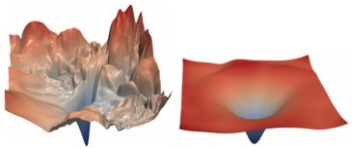
The Transformer Encoder: Residual connections [He et al., 2016]

- **Residual connections** are a trick to help models train better.
  - Instead of $X^{(i)} = \text{Layer}(X^{(i-1)})$ (where i represents the layer)

  - We let $X^{(i)} = X^{(i-1)} + \text{Layer}(X^{(i-1)})$ (so we only have to learn "the residual" from the previous layer)



  - Residual connections are thought to make the loss landscape considerably smoother (thus easier training!)



[no residuals] [residuals]

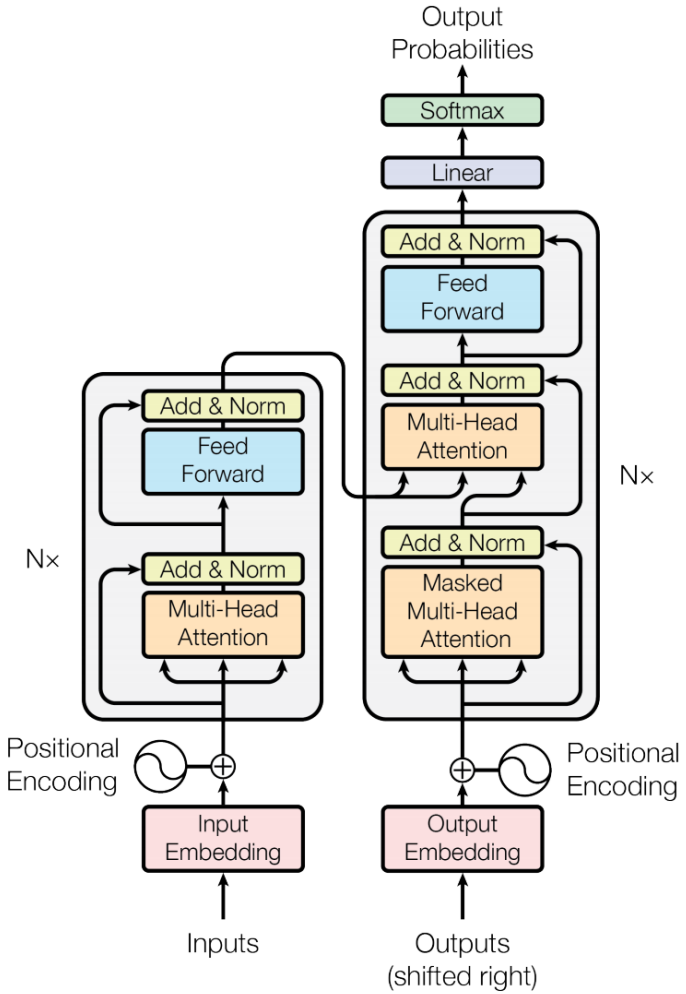
[Loss landscape visualization, Li et al., 2018, on a ResNet]

Layer Normalization [Ba et al., 2016]

- **Problem:** Difficult to train the parameters of a given layer because its input from the layer beneath keeps shifting.

- **Solution:** Reduce uninformative variation by normalizing to zero mean and standard deviation of one within each layer.

Blocks are repeated N=6 times

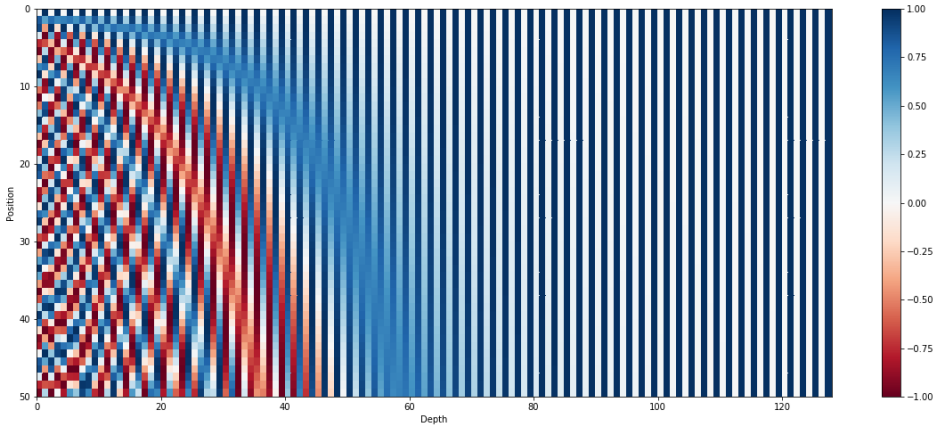


Encoder Input

- **Actual word representations are byte-pair encodings (subword)**
  - Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. ACL 2016.
- Added is a positional encoding so same words at different locations have different overall representations:

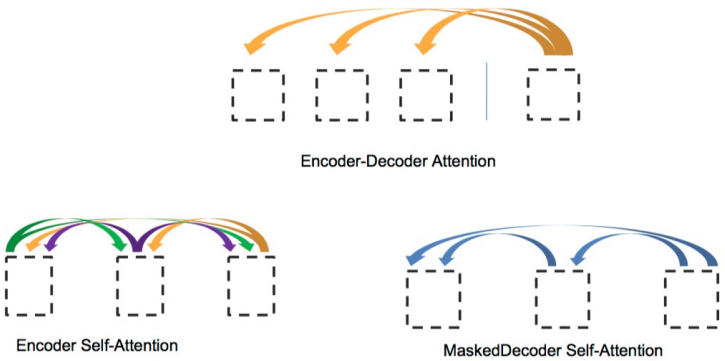
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

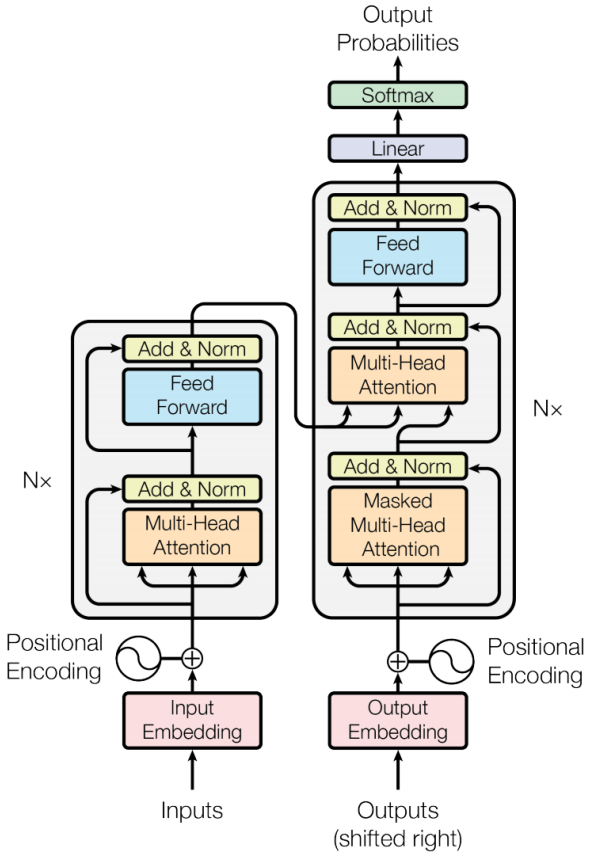
pos is the position of a word
i is the dimension index



Transformer Decoder

- 2 sublayer changes in decoder
- Masked decoder self-attention
  - Only depends on previous words
- Encoder-Decoder Attention
  - **Queries** come from previous decoder layer and **keys and values** come from output of encoder



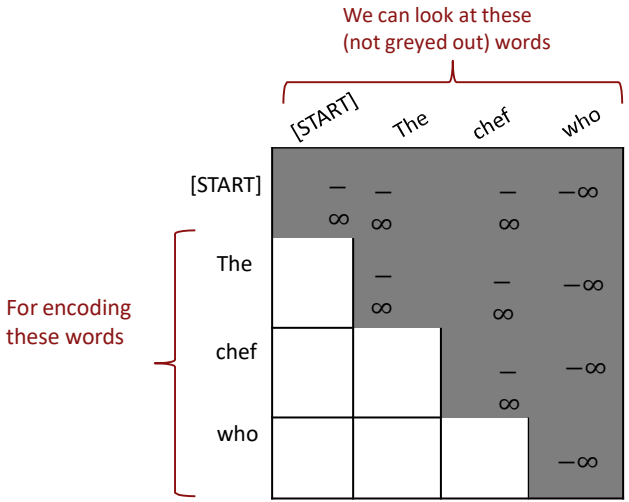


Masking the future in self-attention

- To use self-attention in **decoders**, we need to ensure we can't peek at the future.

- At every timestep, we could change the set of **keys and queries** to include only past words. (Inefficient!)
- To enable parallelization, we **mask out attention** to future words by setting attention scores to $-\infty$.

$$e_{ij} = \begin{cases} q_i^\top k_j, j < i \\ -\infty, j \geq i \end{cases}$$



Great Results with Transformers

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Great Results with Transformers

Next, document generation!

Model	Test perplexity	ROUGE-L
seq2seq-attention, $L = 500$	5.04952	12.7
Transformer-ED, $L = 500$	2.46645	34.2
Transformer-D, $L = 4000$	2.22216	33.6
Transformer-DMCA, no MoE-layer, $L = 11000$	2.05159	36.2
Transformer-DMCA, MoE-128, $L = 11000$	1.92871	37.9
Transformer-DMCA, MoE-256, $L = 7500$	1.90325	38.8

The old standard

Transformers all the way down.

[Liu et al., 2018]; WikiSum dataset

Large Language Models

Language Modeling (Mô hình ngôn ngữ)?

- What is the probability of “Tôi trình bày ChatGPT tại Trường ĐH Công Nghệ” ?
- What is the probability of “Công Nghệ học Đại trình bày ChatGPT tại Tôi” ?
- “Tôi trình bày ChatGPT tại Trường ĐH Công nghệ, địa điểm …”) or

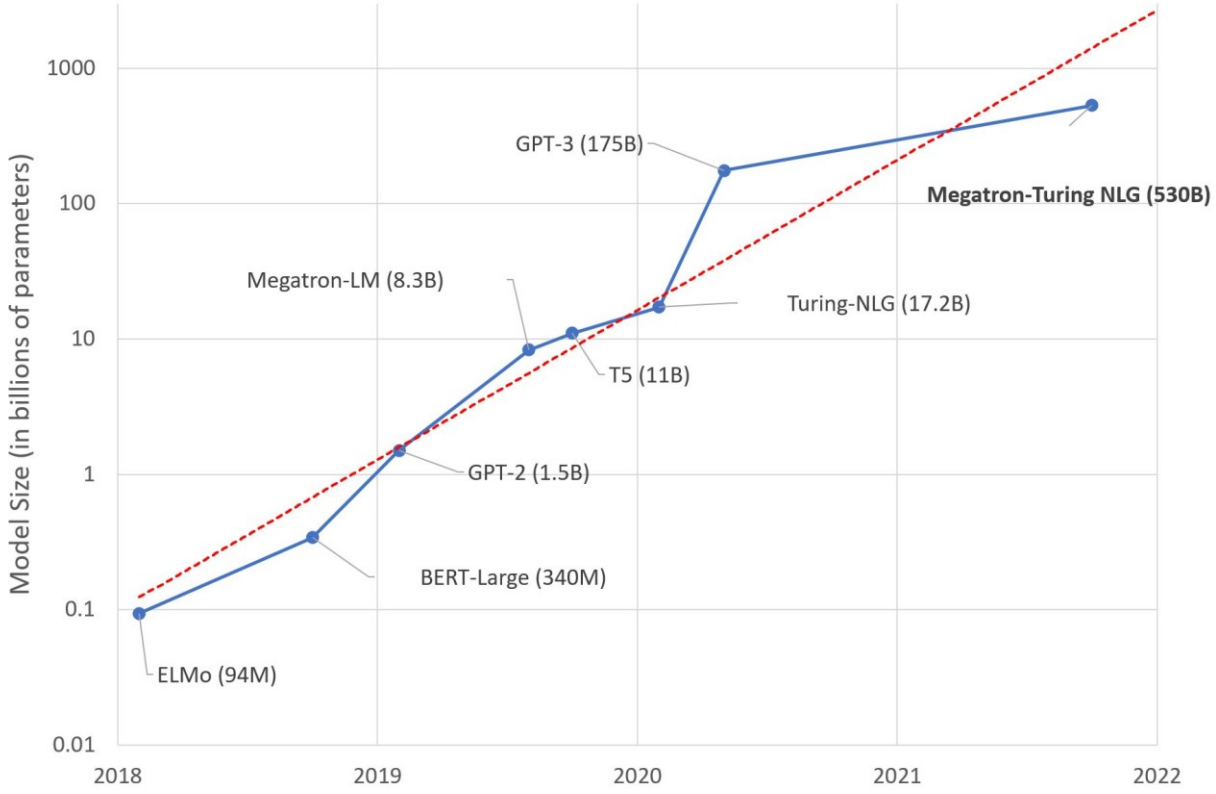
P(…/Tôi trình bày ChatGPT tại Trường ĐH Công nghệ, địa điểm) ?

- A model that computes either of these:

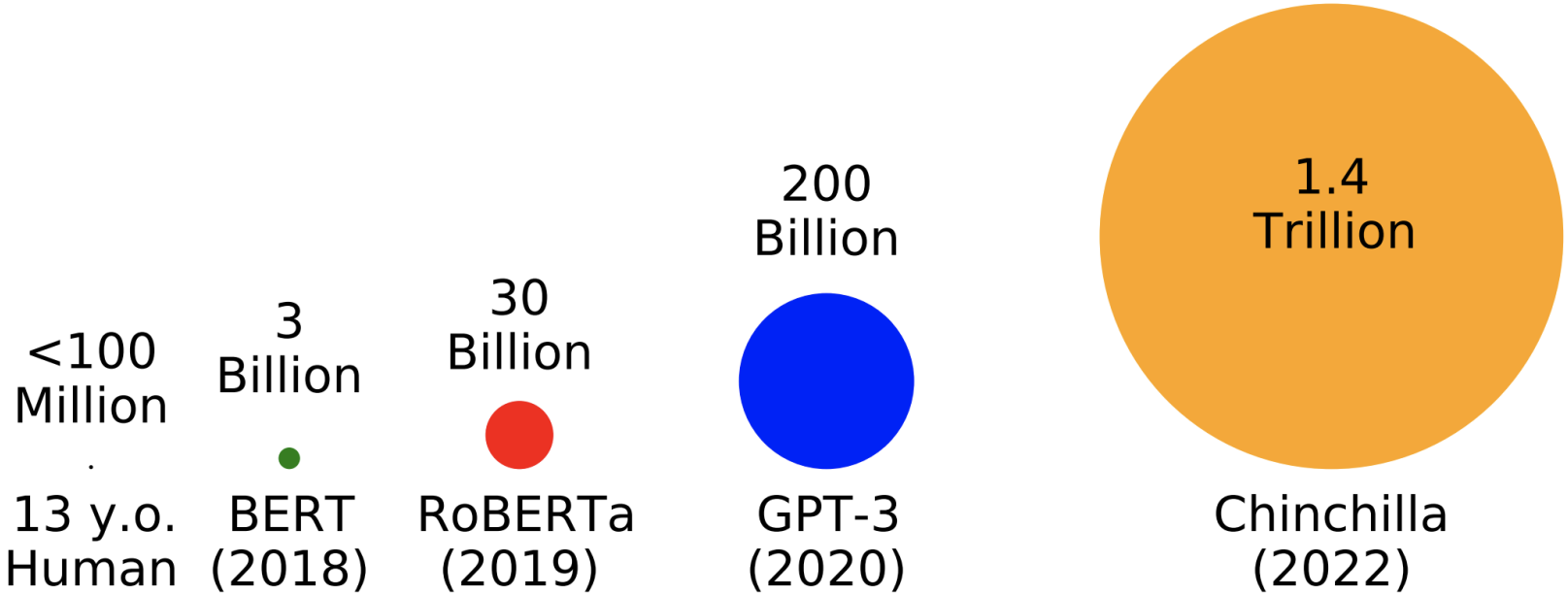
$W = w_1, w_2, w_3, w_4, w_5 \ldots w_n$

$P(W)$ or $P(w_n | w_1, w_2 \ldots w_{n-1})$ is called a **language model**

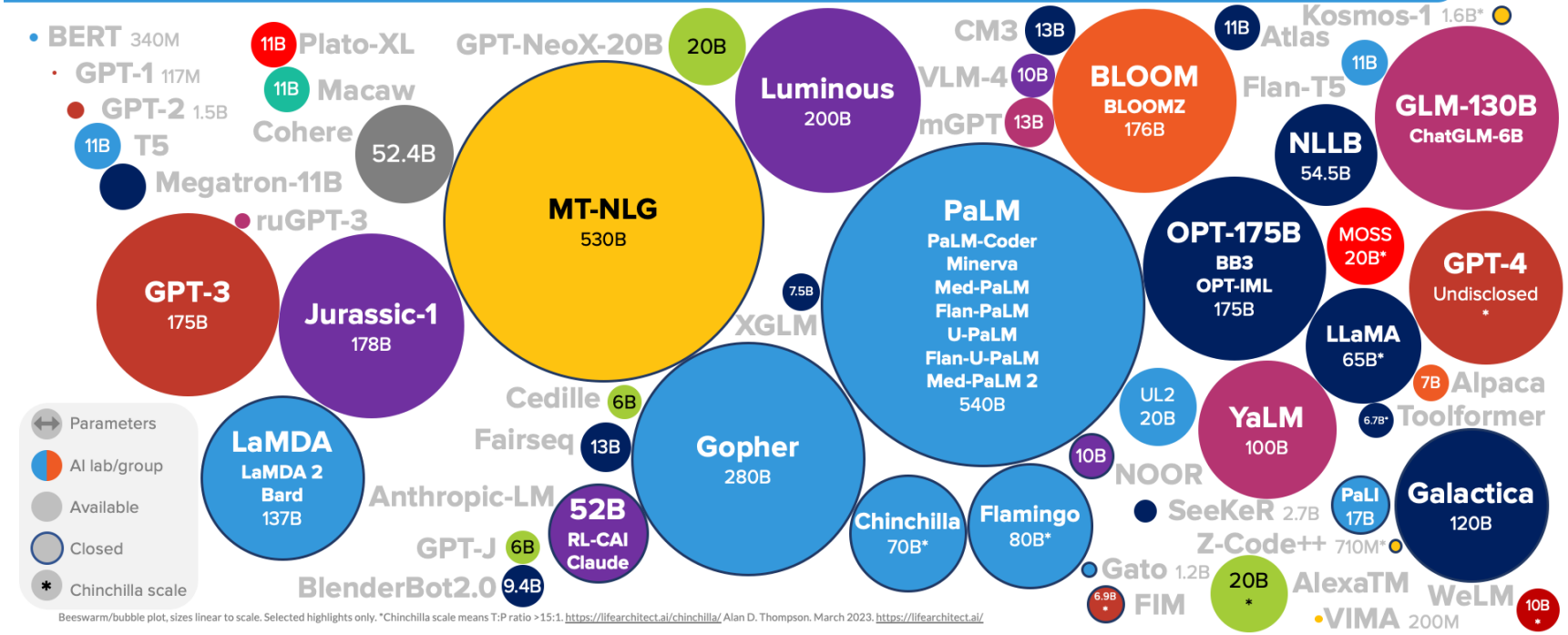
Large Language Model



Large Language Model



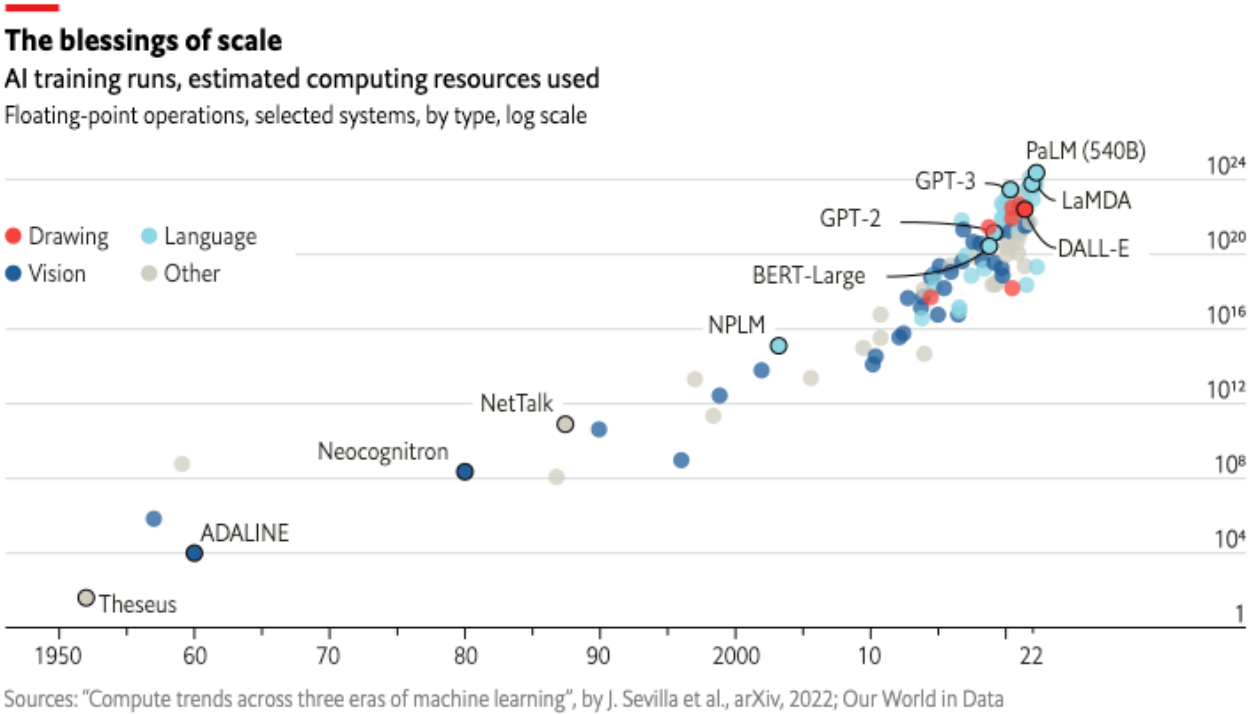
LANGUAGE MODEL SIZES TO MAR/2023



Beeswarm/bubble plot, sizes linear to scale. Selected highlights only. *Chinchilla scale means T:P ratio >15:1. https://lifearchitect.ai/chinchilla/ Alan D. Thompson. March 2023. https://lifearchitect.ai/

LifeArchitect.ai/models

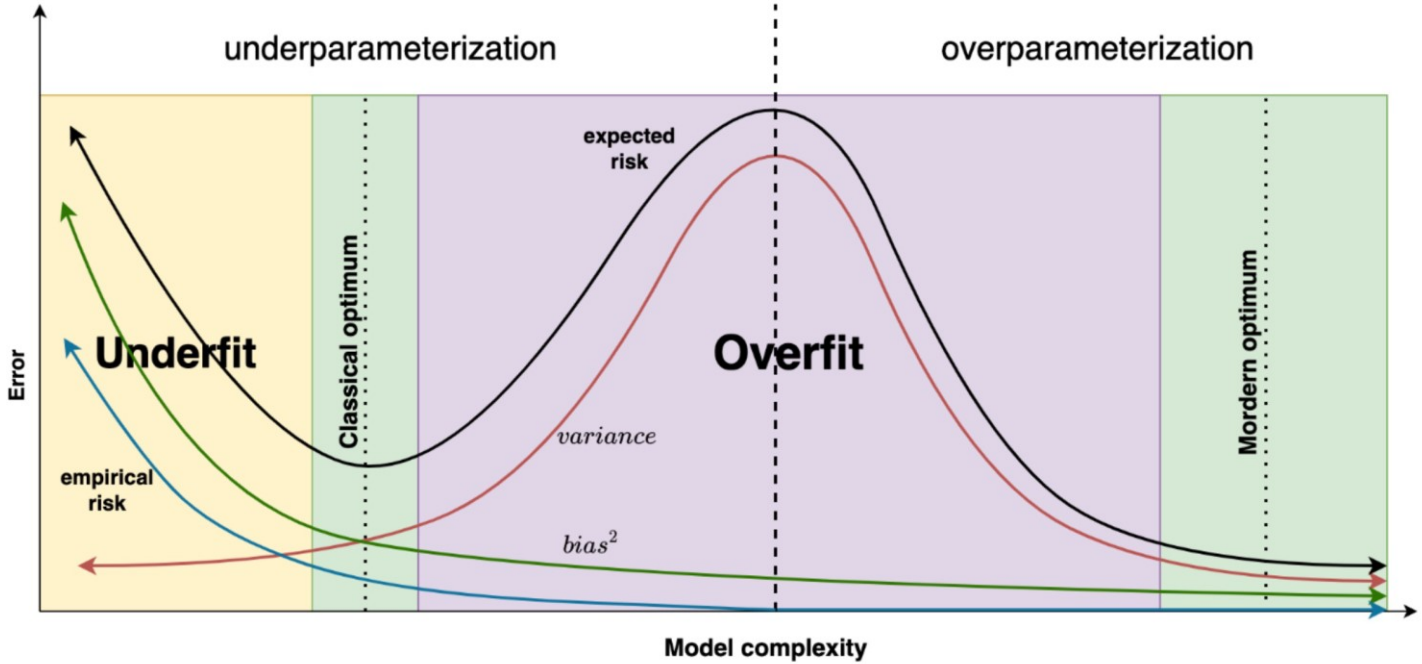
Large Language Models - **yottaFlops of Compute**



Source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture11-prompting-rlhf.pdf

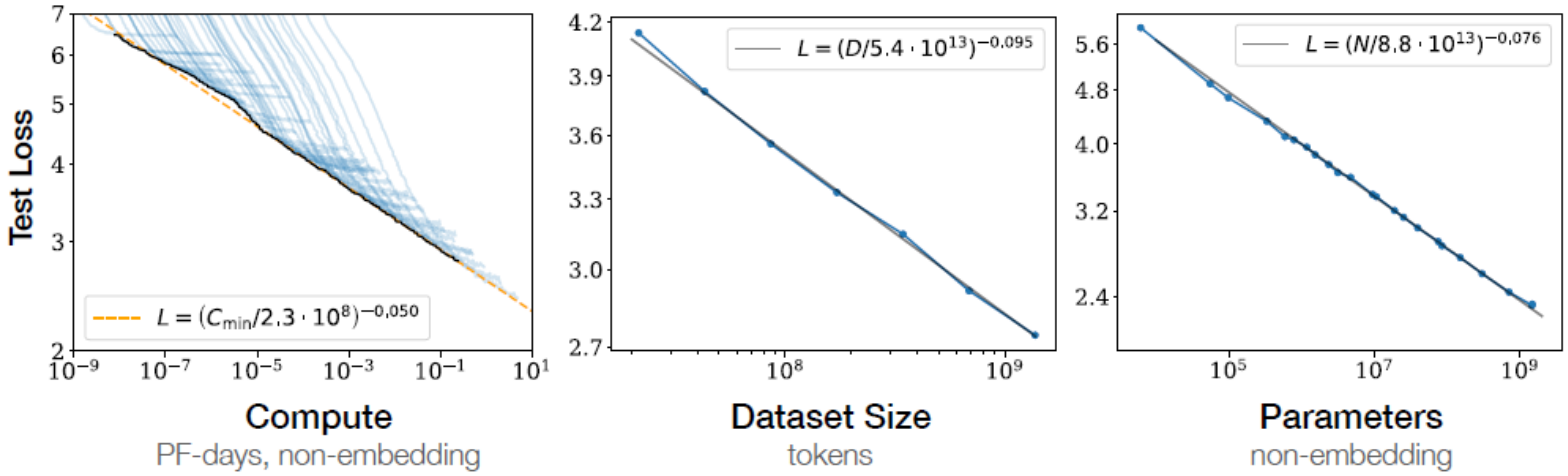
Why LLMs?

- **Double Descent**



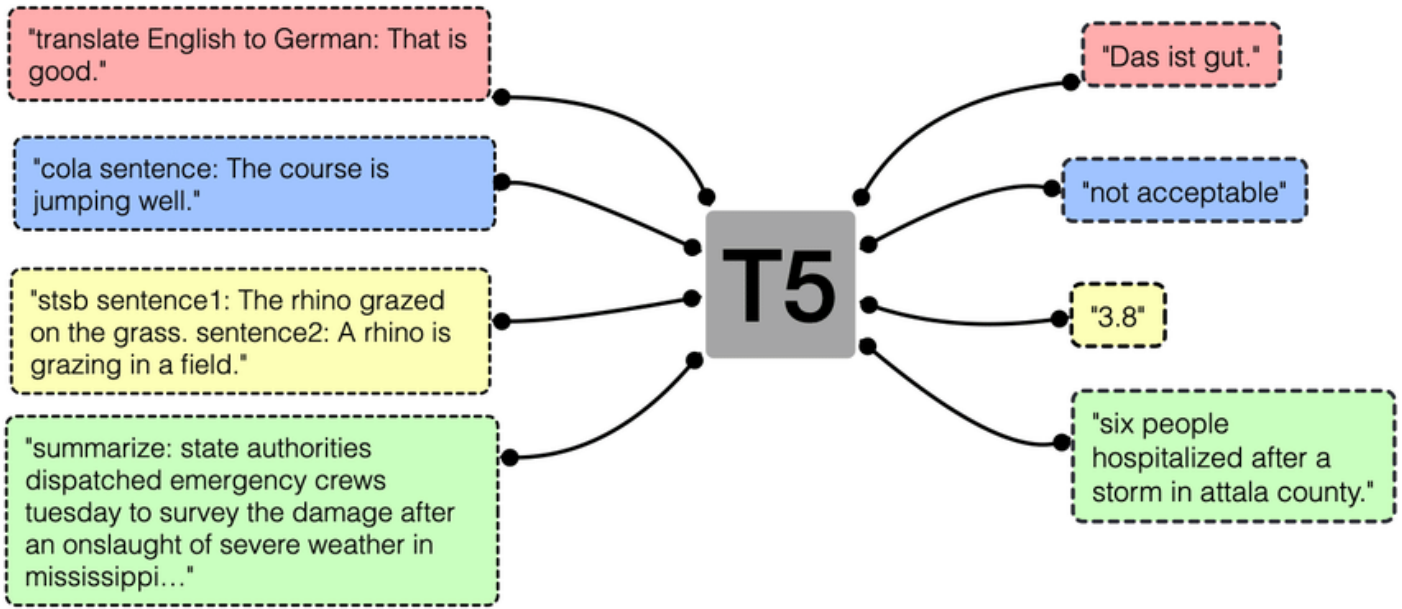
Why LLMs?

- **Scaling Law for Neural Language Models**
  - Performance depends strongly on scale! We keep getting better performance as we scale the model, data, and compute up!



Why LLMs?

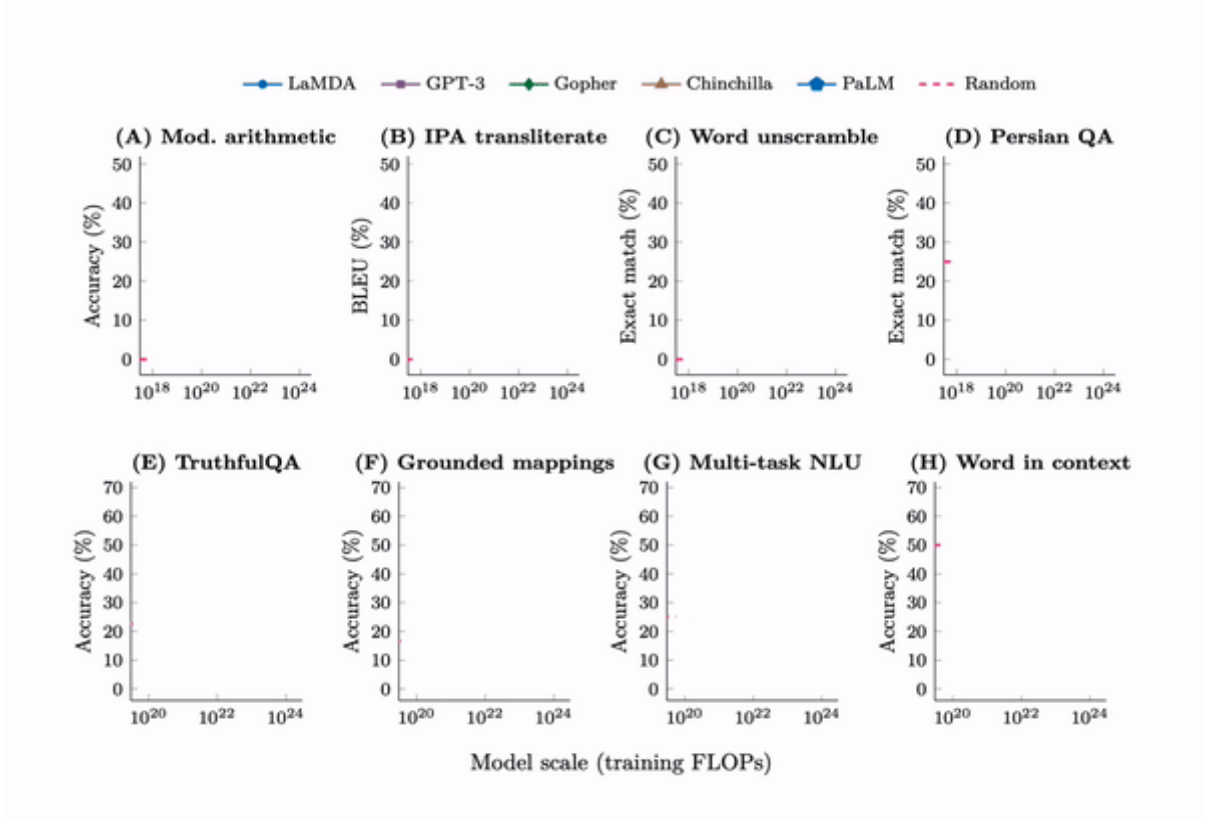
- **Generalization**
  - We can now use one single model to solve many NLP tasks



Why LLMs? Emergence in few-shot prompting

Emergent Abilities

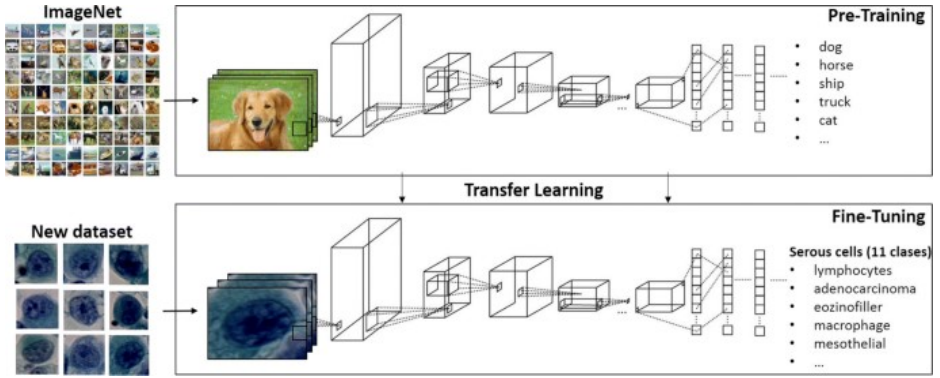
- Some ability of LM is not present in smaller models but is present in larger models



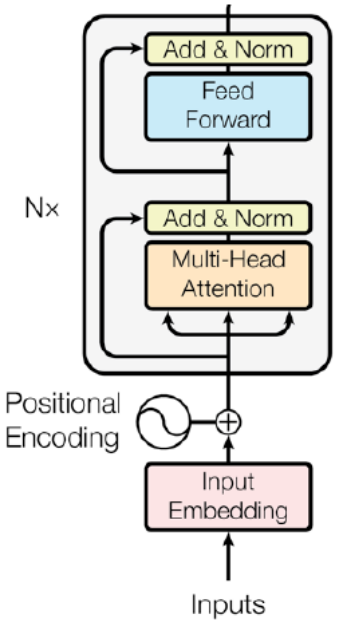
What is pre-training / fine-tuning?

- "**Pre-train**" a model on a large dataset for task X, then "**fine-tune**" it on a dataset for task Y
- Key idea: X is somewhat related to Y, so a model that can do X will have some good neural representations for Y as well (transfer learning)
- ImageNet pre-training is huge in computer vision: learning generic visual features for recognizing objects

Can we find some task X that can be useful for a wide range of downstream tasks Y?



BERT: Bidirectional Encoder Representations from Transformers



- It is a fine-tuning approach based on a deep **bidirectional Transformer encoder** instead of a Transformer decoder
- The key: learn representations based on **bidirectional contexts**

  Example #1: we went to the river bank.

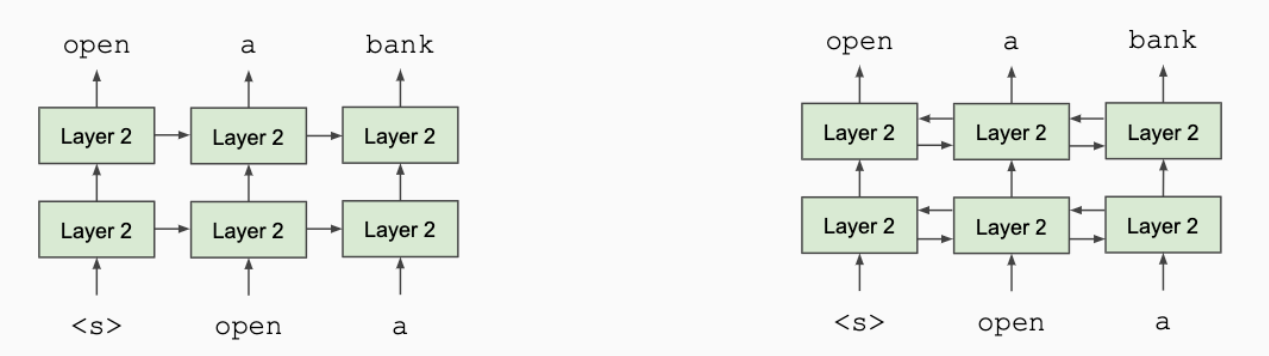
  Example #2: I need to go to bank to make a deposit.

- Two new pre-training objectives:
  - **Masked language modeling (MLM)**
  - Next sentence prediction (NSP) - Later work shows that NSP hurts performance though..

Source: (Devlin et al, 2019): BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Masked Language Modeling (MLM)

- Q: Why we can't do language modeling with bidirectional models?



- Solution: Mask out k% of the input words, and then predict the masked words

store gallon

the man went to [MASK] to buy a [MASK] of milk

k = 15% in practice

Next Sentence Prediction (NSP)

- Motivation: many NLP downstream tasks require understanding the relationship between two sentences (natural language inference, paraphrase detection, QA)
- NSP is designed to reduce the gap between pre-training and fine-tuning

[CLS]: a special token always at the beginning

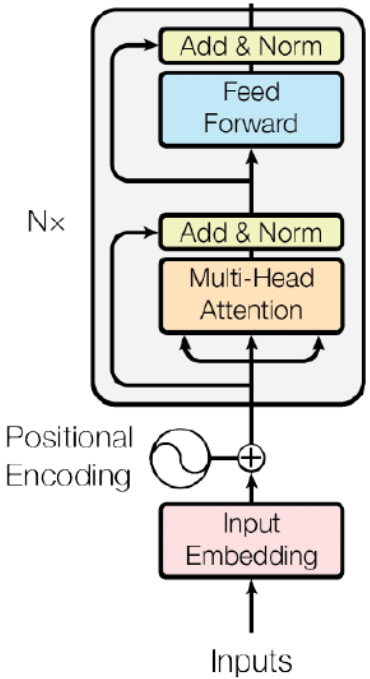
[SEP]: a special token used to separate two segments

Input = [CLS] the man went to [MASK] store [SEP]
he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

They sample two contiguous segments for 50% of the time and another random segment from the corpus for 50% of the time

BERT pre-training



- BERT-base: 12 layers, 768 hidden size, 12 attention heads, 110M parameters
  Same as OpenAI GPT
- BERT-large: 24 layers, 1024 hidden size, 16 attention heads, 340M parameters

OpenAI GPT was trained on BooksCorpus only!

- Training corpus: Wikipedia (2.5B) + BooksCorpus (0.8B)
- Max sequence size: 512 wordpiece tokens (roughly 256 and 256 for two non-contiguous sequences)
- Trained for 1M steps, batch size 128k

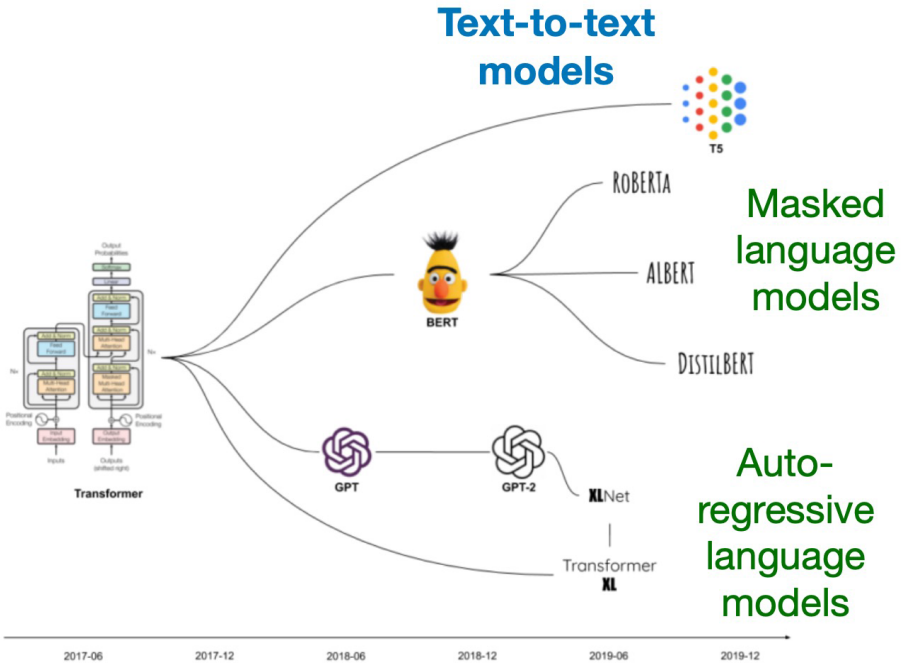
RoBERTa

- BERT is still under-trained
- Removed the next sentence prediction pre-training — it adds more noise than benefits!
- Trained longer with 10x data & bigger batch sizes
- Pre-trained on 1,024 V100 GPUs for one day in 2019

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

(Liu et al., 2019): RoBERTa: A Robustly Optimized BERT Pretraining Approach

Three major forms of pre-training (LLMs)

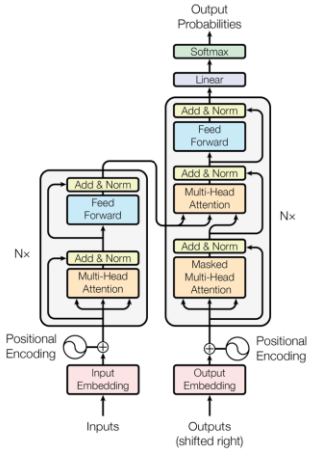


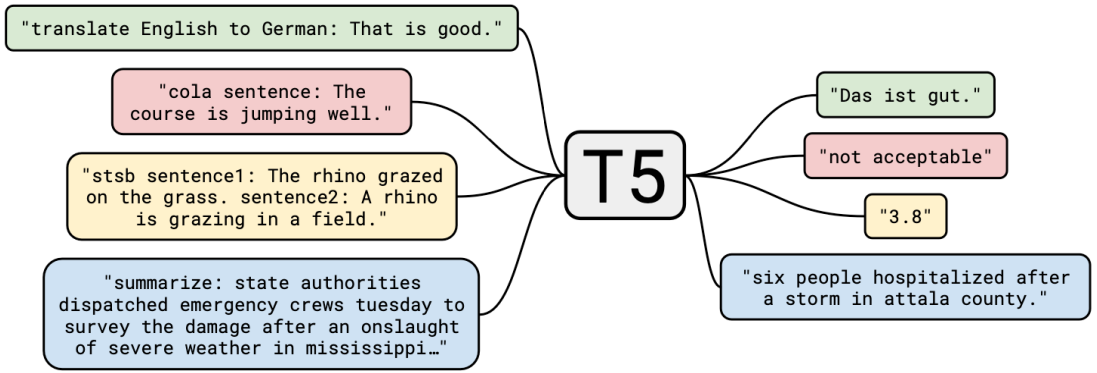
- Masked language models = Transformer encoder
- Autoregressive language models = Transformer decoder
- **Text-to-text models = Transformer encoder-decoder**

Text-to-text models: the best of both worlds?

- **Encoder-only models (e.g., BERT)** enjoy the benefits of **bidirectionality** but they can't be used to generate text
- **Decoder-only models (e.g., GPT)** can do generation but they are left-to-right LMs..
- Text-to-text models combine the best of both worlds!

T5 = **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer

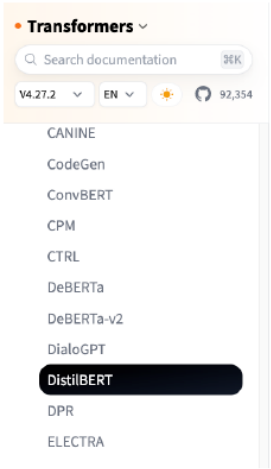




(Raffel et al., 2020): Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

How to use these pre-trained models?

Transformers



DistilBERT

All model pages distilbert Hugging Face Spaces

Overview

The DistilBERT model was proposed in the blog post Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT, and the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than *bert-base-uncased*, runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark.

```
>>> from transformers import AutoTokenizer

>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")


>>> def tokenize_function(examples):
...     return tokenizer(examples["text"], padding="max_length", truncation=True)


>>> tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

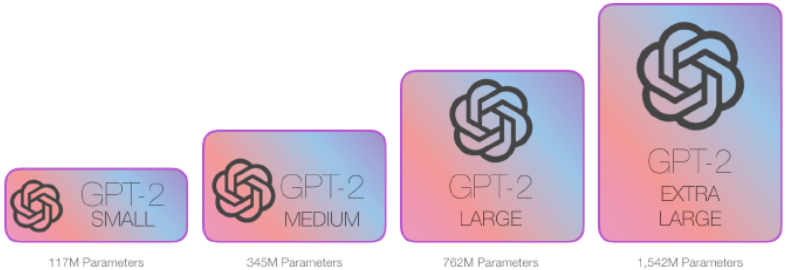
```
>>> from transformers import AutoModelForSequenceClassification

>>> model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased", num_labels=5)
```

From GPT to GPT-2 to GPT-3

- All **decoder-only Transformer-based language models**
- Model size ↑, training corpora ↑

GPT-2

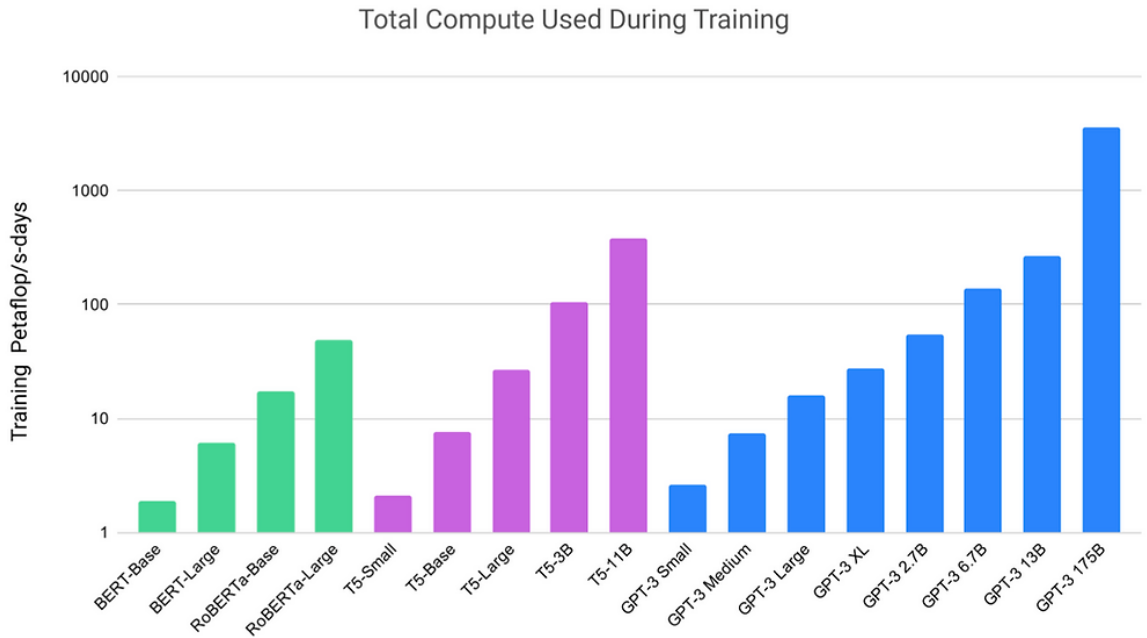
Context size = 1024



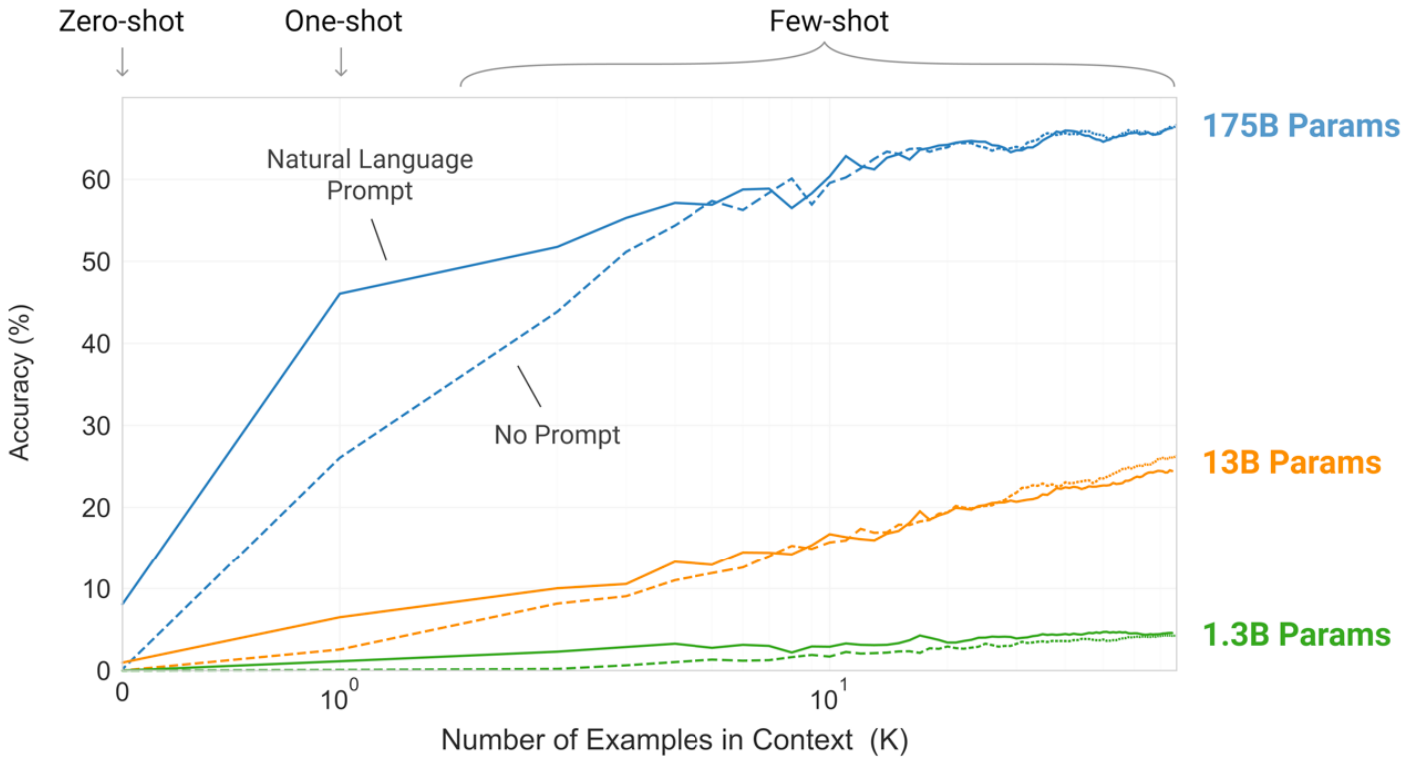
.. trained on 40Gb of Internet text ..

GPT-3: language models are few-shot learners

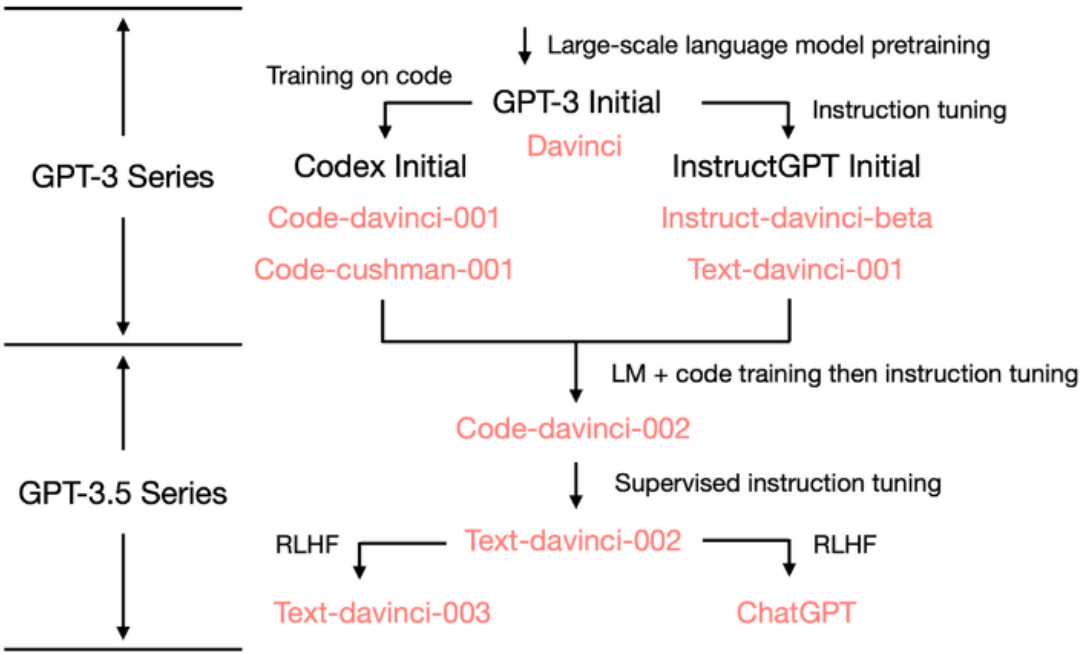
- GPT-2 → GPT-3: 1.5B → **175B** (# of parameters), ~14B → **300B** (# of tokens)



GPT-3's in-context learning



[2020] GPT-3 to [2022] ChatGPT



What's new?

- Training on code
- **Supervised instruction tuning**
- **RLHF = Reinforcement learning from human feedback**

Source: Fu, 2022, "How does GPT Obtain its Ability? Tracing Emergent Abilities of Language Models to their Sources"

Large Language models Risks

- LLMs make mistakes
  - (falsehoods, hallucinations)
- LLMs can be misused
  - (misinformation, spam)
- LLMs can cause harms
  - (toxicity, biases, stereotypes)
- LLMs can be attacked
  - (adversarial examples, poisoning, prompt injection)
- The cost of LLMs is expensive

Practice

1. Word Embedings task
2. Bert for Question Answering task
3. Transformer for Machine Translation task

Summary

- Introduction to NLP
- Word Embedding & Attention & Transformer model
- Large Language models

Thank you

Email me
vinhnv@vnu.edu.vn

Course: Large Language Models and Its Applications
sponsored by **KEPCO KDN Co., Ltd.**
Eco-friendly & Digital Centered Energy ICT Platform Leader