Transformers

Sweta Agrawal
swetaagrawal@google.com

My Research

Google Translate

Gemini

Quick Poll

- ❑ How familiar are you with transformers?
- ❑ What is the key building block of transformers?
- ❑ Have you used it in your projects?
- ❑ Have you implemented it yourself?

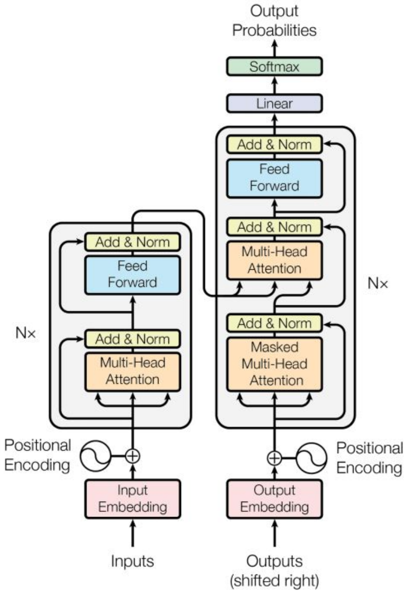
Plan

Session I: Transformers and its nuts and bolts

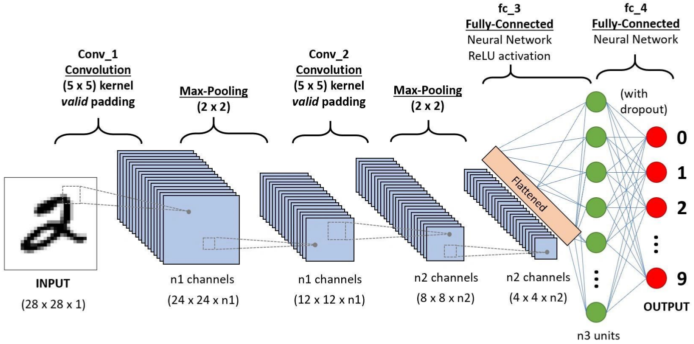
Attention Mechanism, Architecture Overview

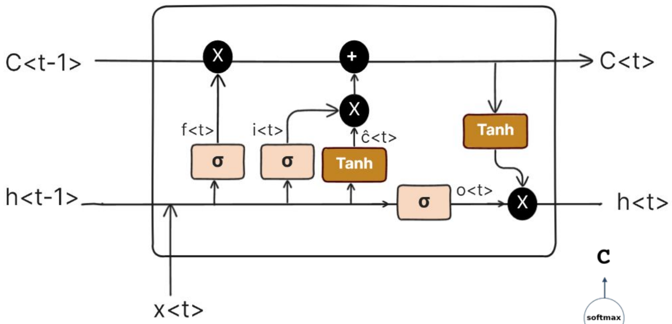
Session II: Application of Transformers

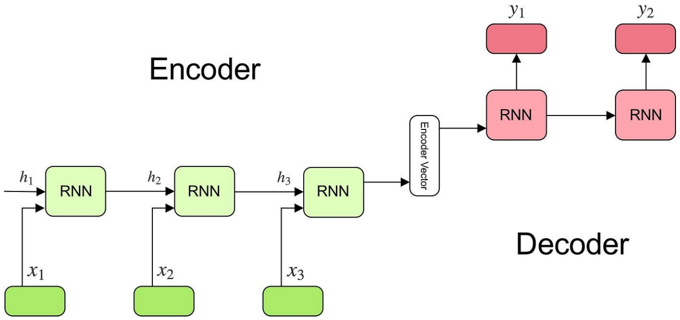
Pre-trained Models, Multi-task and Multimodal models

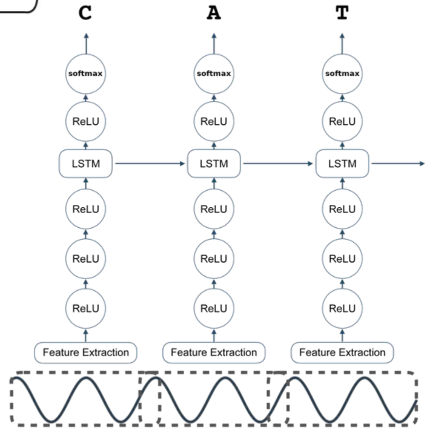


AI before Transformers - An architecture per task



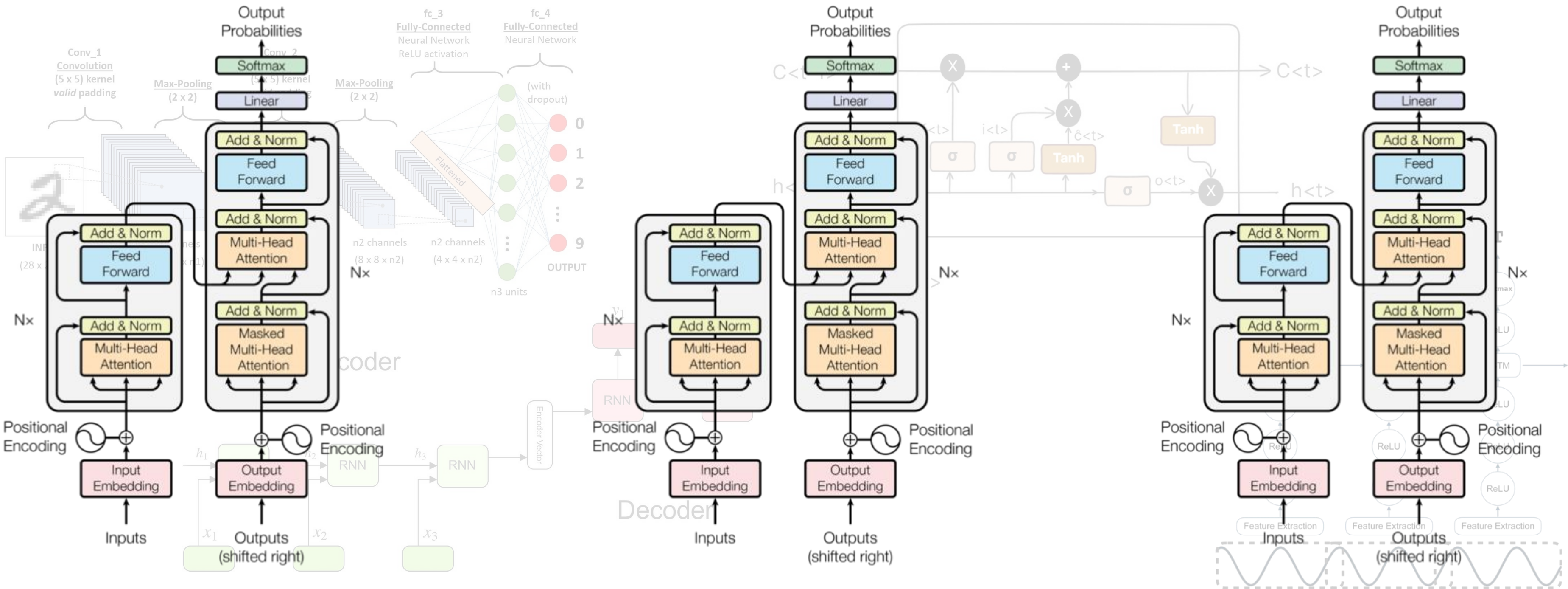






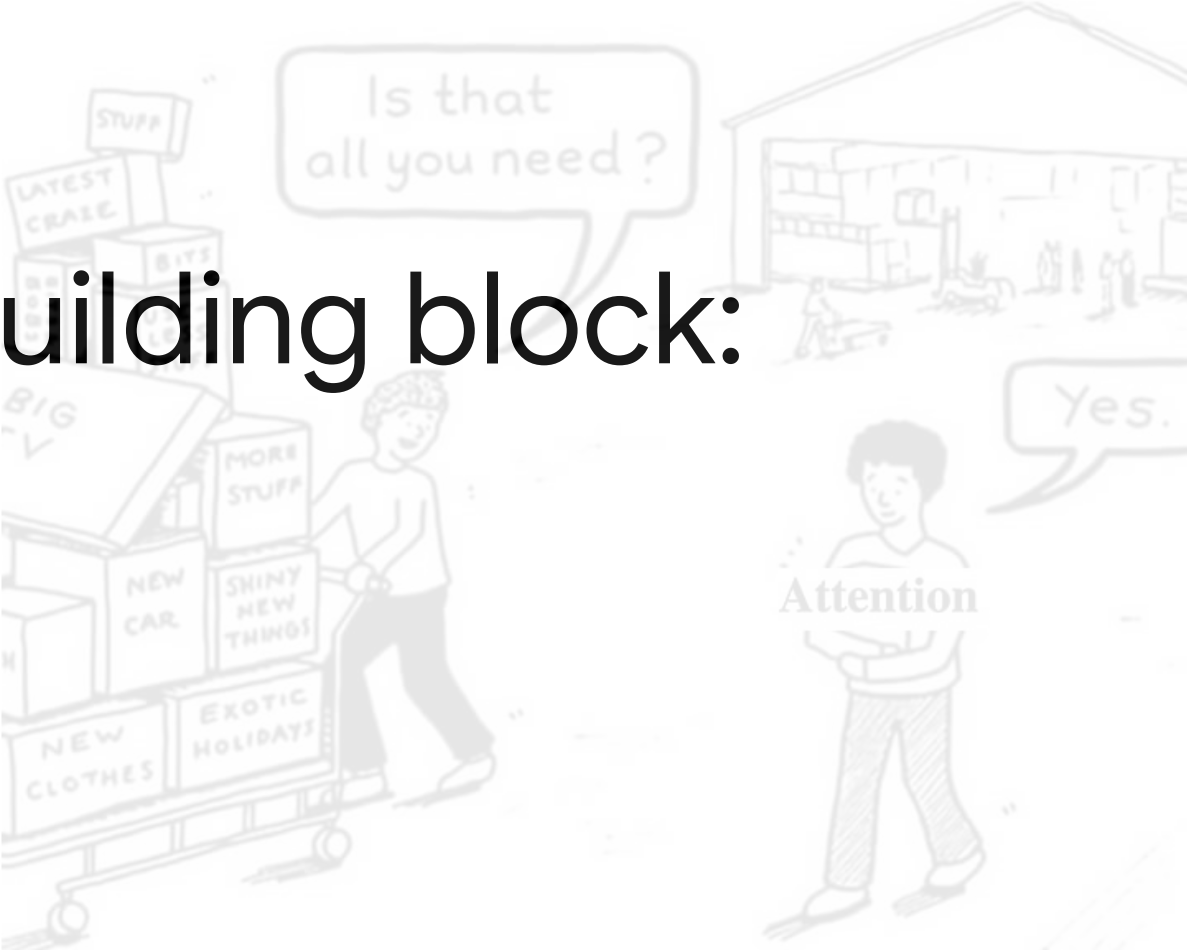
Slide Inspiration: Lucas Beyer

NLP after Transformers - One type fits all**



Slide Inspiration: Lucas Beyer

The core building block: Attention

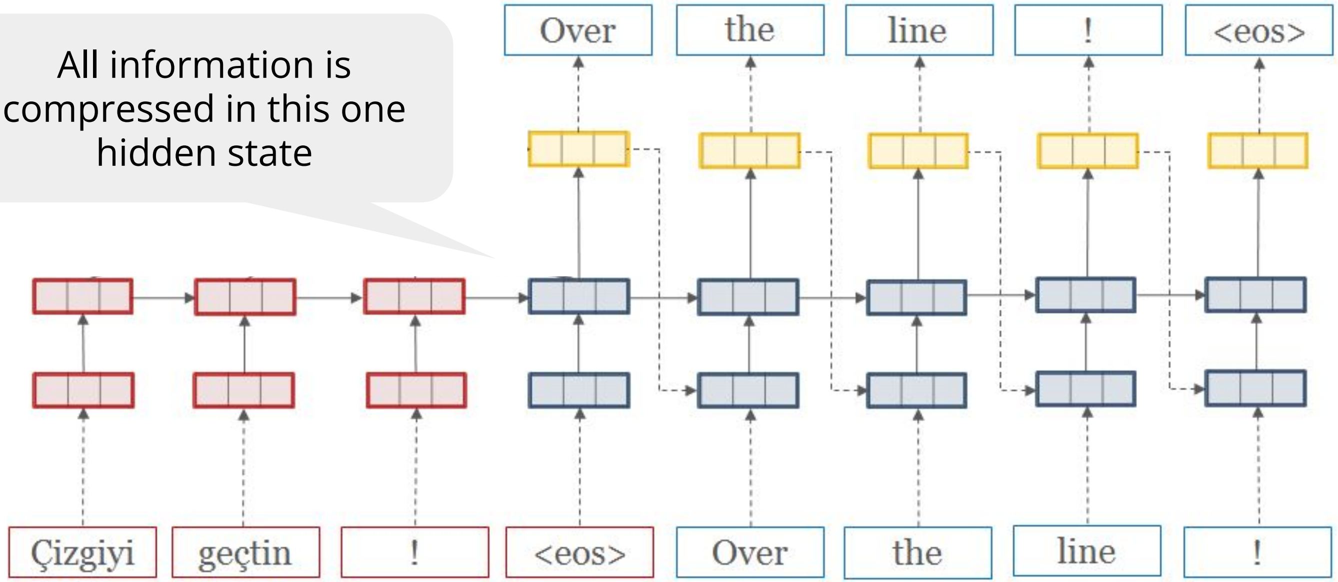


Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)

All information is compressed in this one hidden state



Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)

At each time step, ***attend*** to the parts of source that is important

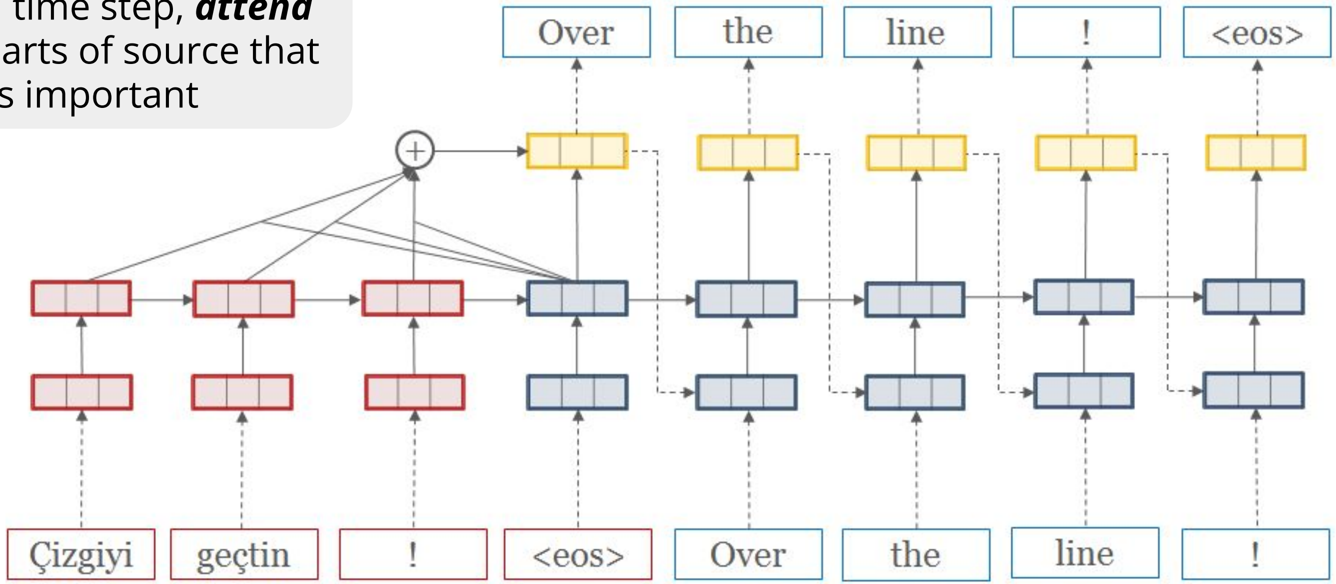
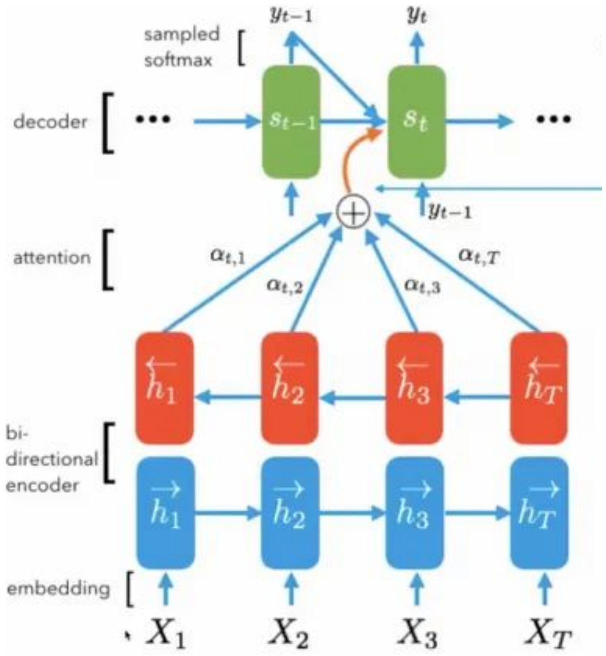


Image Credits: https://jalammar.github.io/illustrated-transformer/

Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)



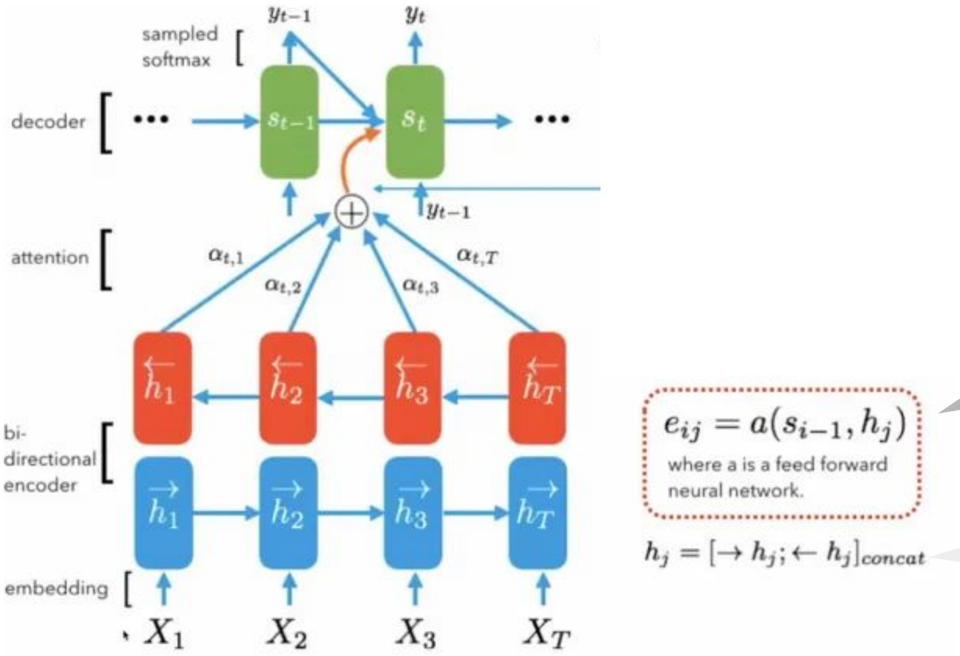
$h_j = [\rightarrow h_j; \leftarrow h_j]_{concat}$

Hidden representation of all input tokens H: [h_1, h_2, h_3, h_4]

Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)



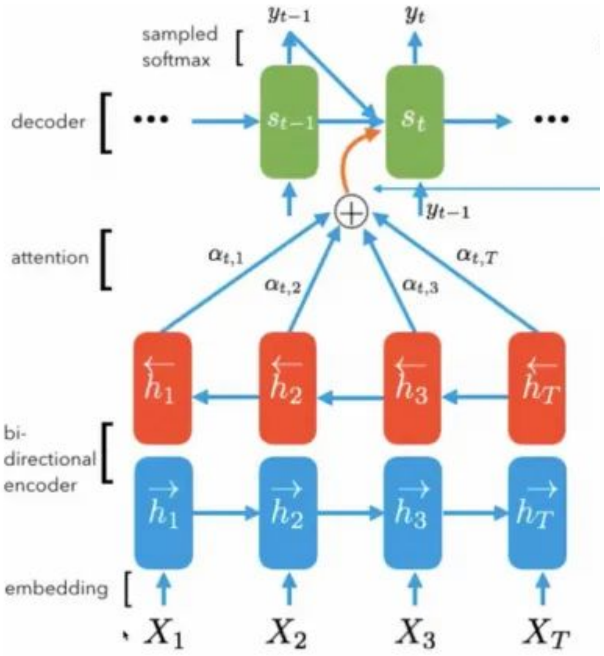
Attention scores: current decoder state $[s_{t-1}]$ and H

Hidden representation of all input tokens H: $[h_1, h_2, h_3, h_4]$

Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)



$$\alpha_{i,j} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

Attention distribution using softmax

$$e_{ij} = a(s_{i-1}, h_j)$$

where a is a feed forward neural network.

Attention scores: current decoder state [s_{t-1}] and H

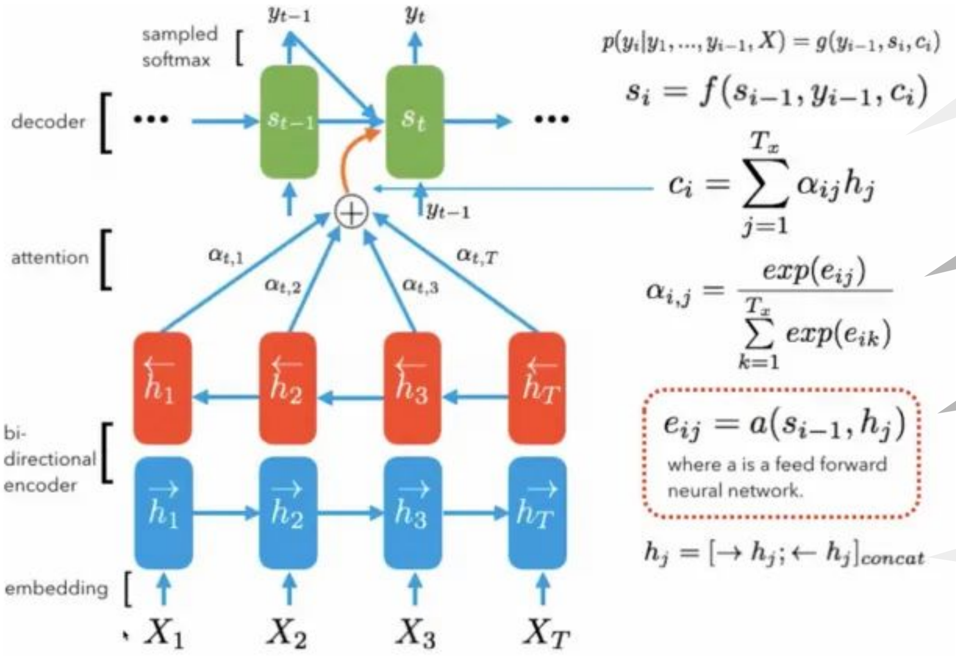
$$h_j = [\rightarrow h_j; \leftarrow h_j]_{concat}$$

Hidden representation of all input tokens H: [h_1, h_2, h_3, h_4]

Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)



Updated information: weighted representation of hidden states

Attention distribution using softmax

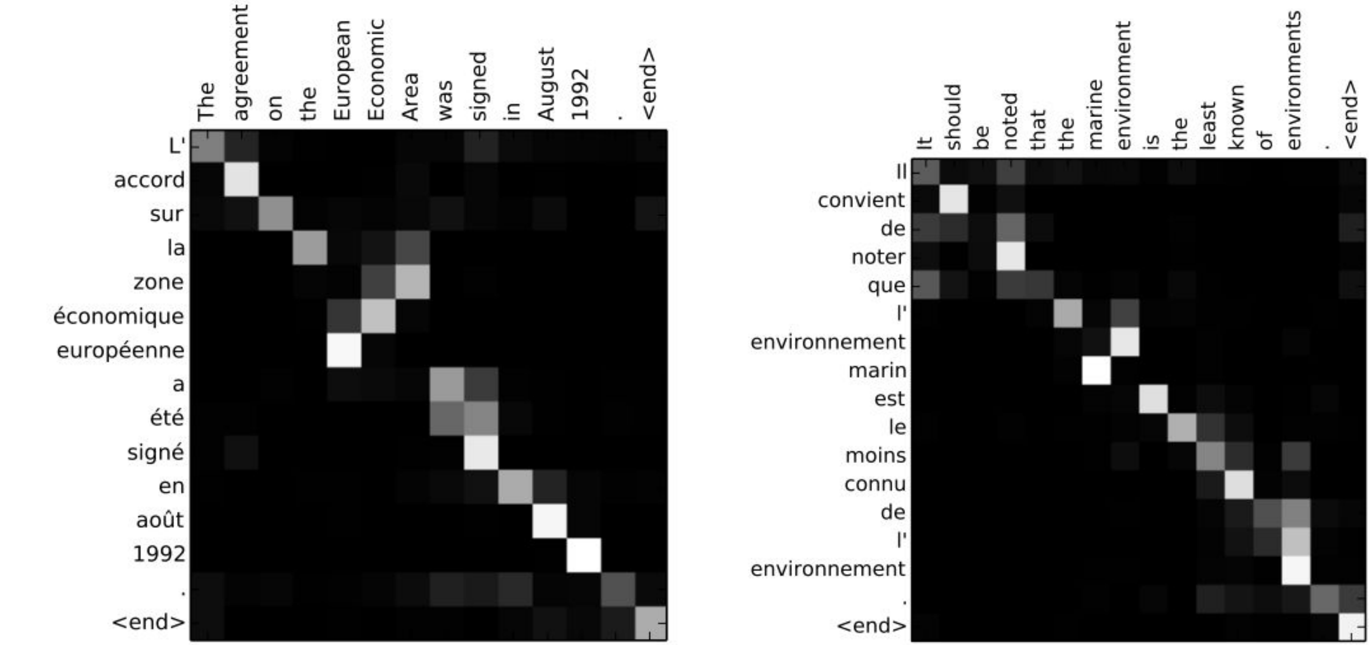
Attention scores: current decoder state $[s_{t-1}]$ and H

Hidden representation of all input tokens H: $[h_1, h_2, h_3, h_4]$

Introduction of Attention

Neural Machine Translation by Jointly Learning to Align and Translate

(Bahdanau et al., 2015)



(a) (b)

Image Credits: https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Attention is **all you need**

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc

… to attend to **all** positions in the decoder up to and including that position. **We need** to prevent … **We** implement this inside of scaled dot-product **attention** by masking out (setting to $-\infty$) …

Attention as a soft dictionary lookup

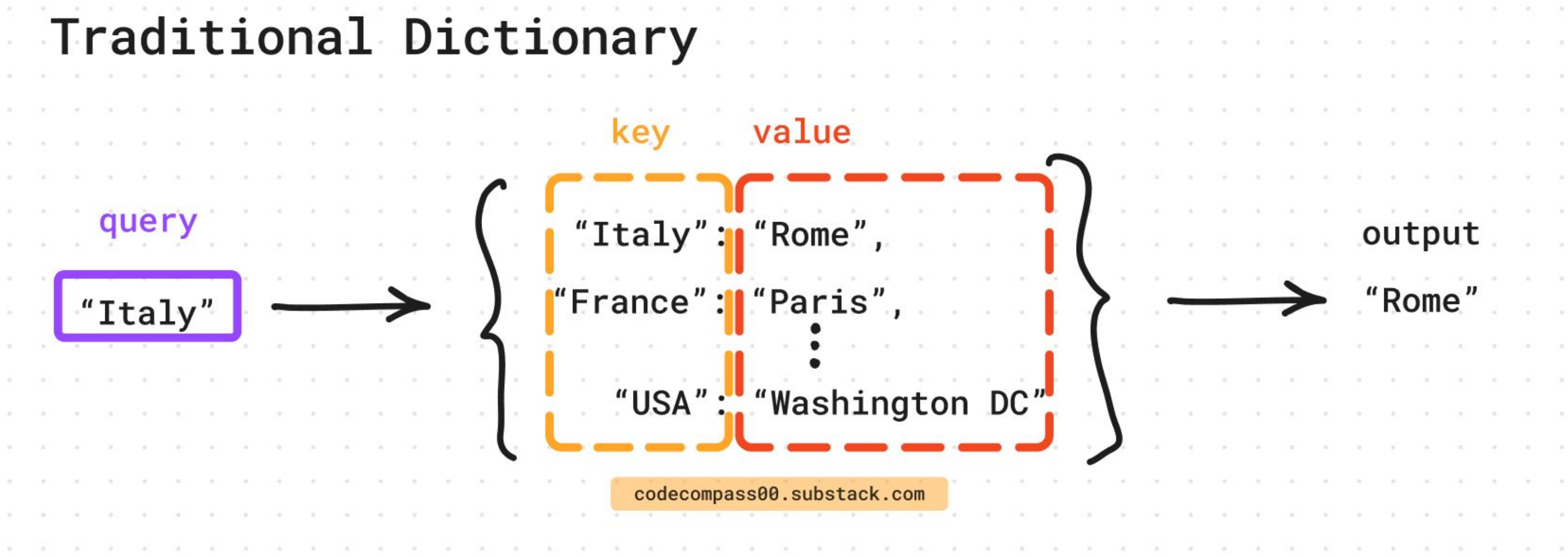


Image Credits: https://codecompass00.substack.com/p/visual-guide-attention-mechanism-transformers

Attention as a soft dictionary lookup

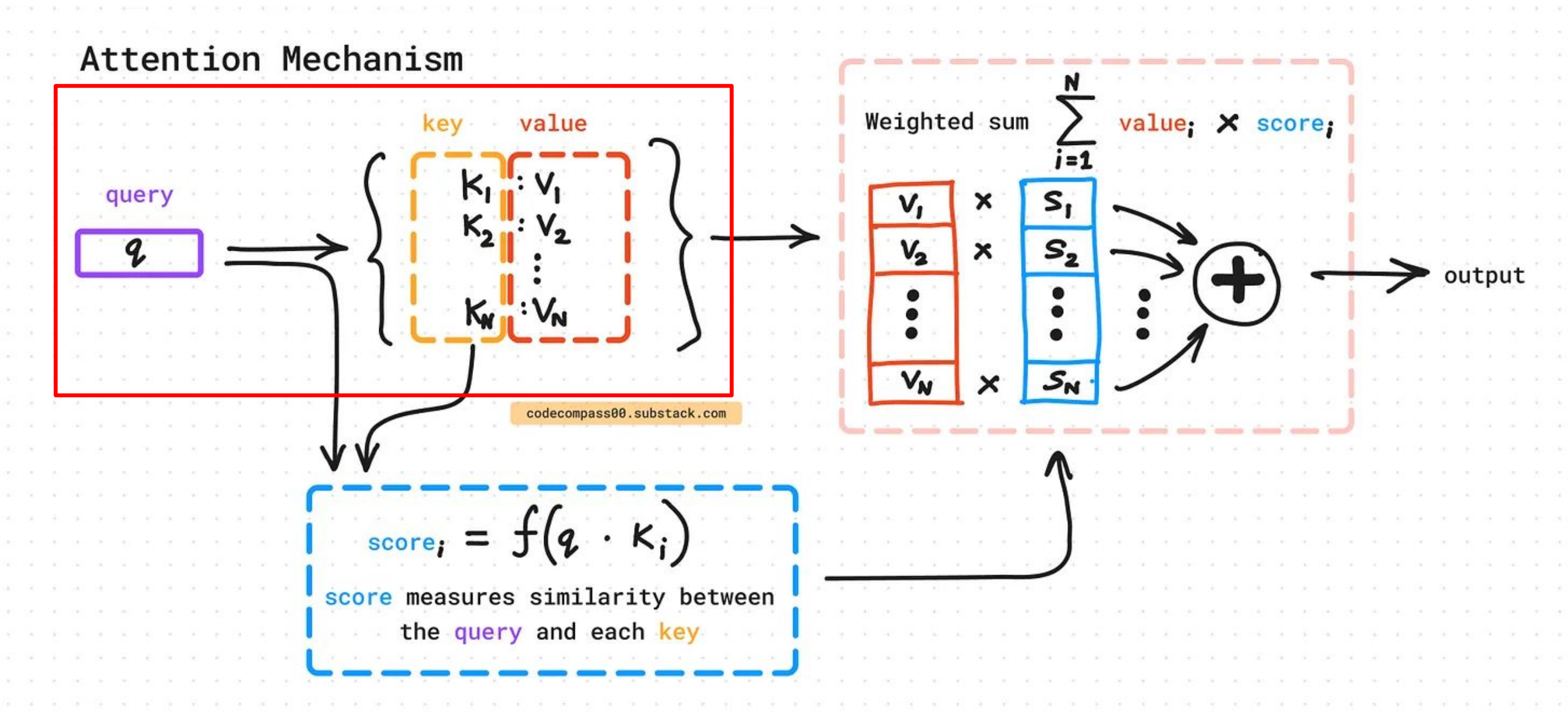


Image Credits: https://codecompass00.substack.com/p/visual-guide-attention-mechanism-transformers

Attention as a soft dictionary lookup

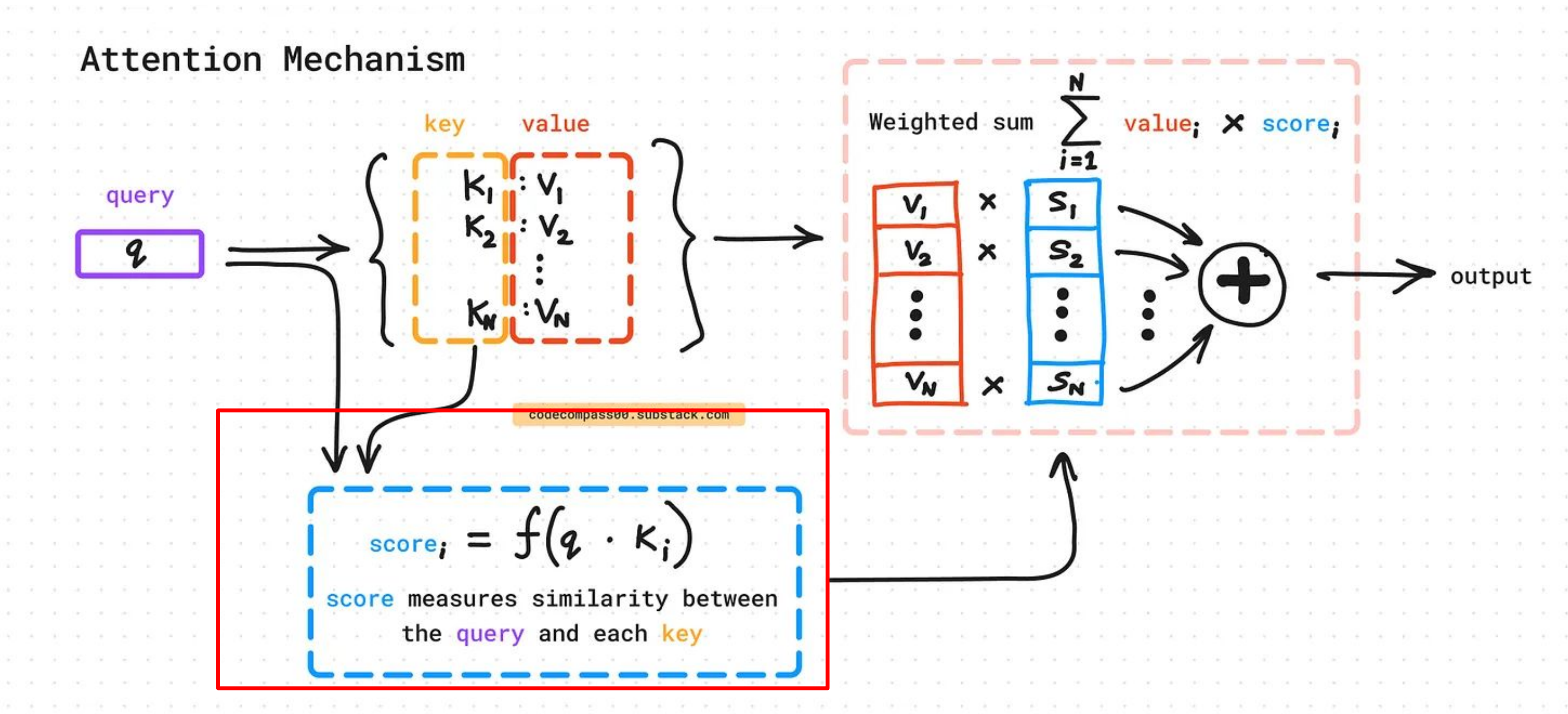


Image Credits: https://codecompass00.substack.com/p/visual-guide-attention-mechanism-transformers

Attention as a soft dictionary lookup

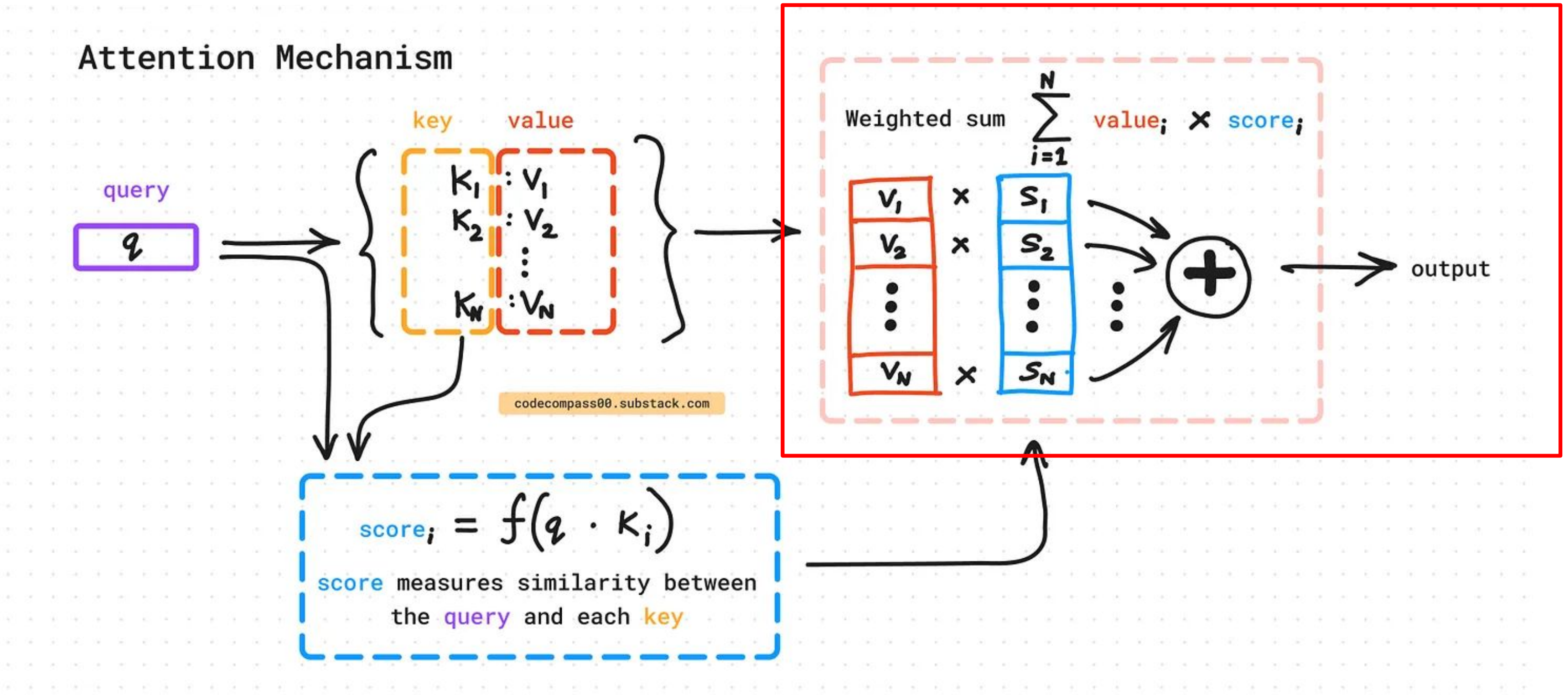
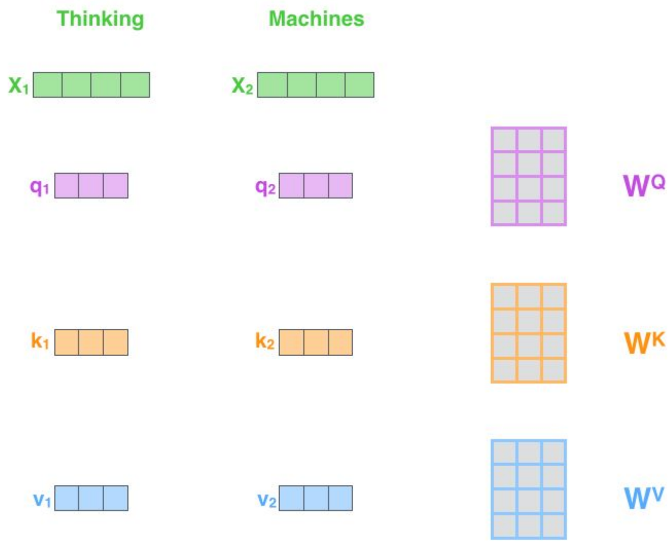


Image Credits: https://codecompass00.substack.com/p/visual-guide-attention-mechanism-transformers

Attention as a dictionary lookup



keys and values - derived from the same input x

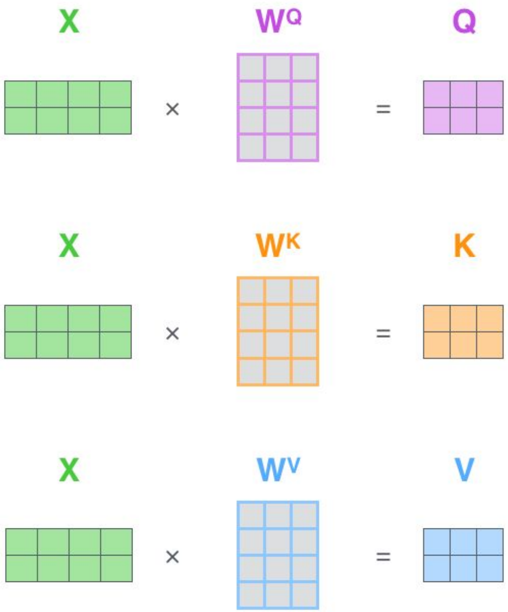
$k = W_k \cdot x \quad v = W_v \cdot x$

queries can be from same or different

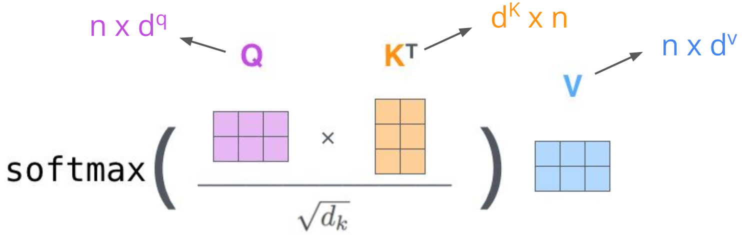
$q = W_q \cdot x$ or $W_q \cdot y$

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

Attention as a dictionary lookup

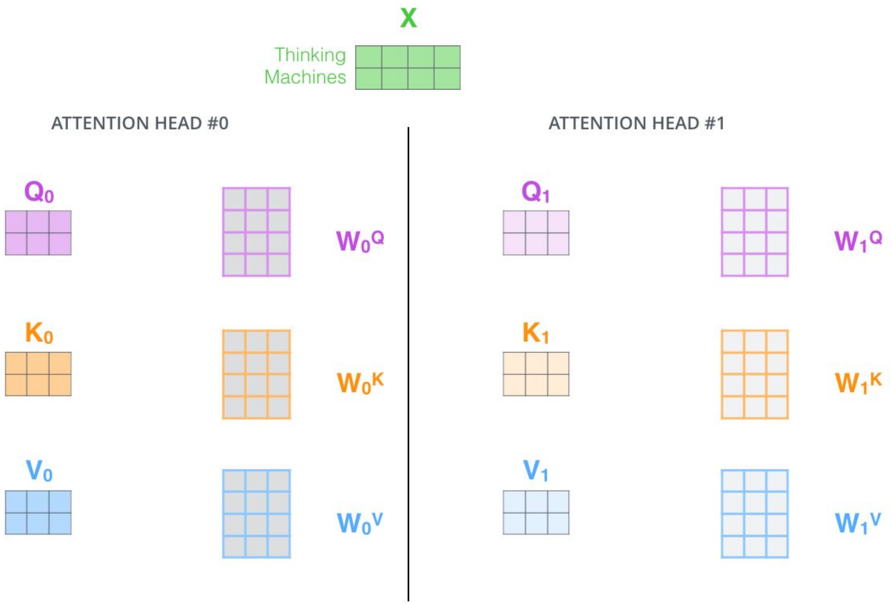


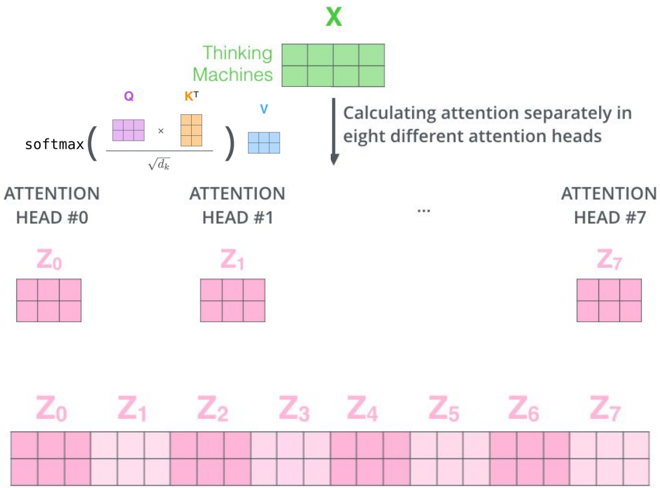
In practice, we have several queries $q_{[1:m]}$



prevents dot product to become too large

Multi-head attention

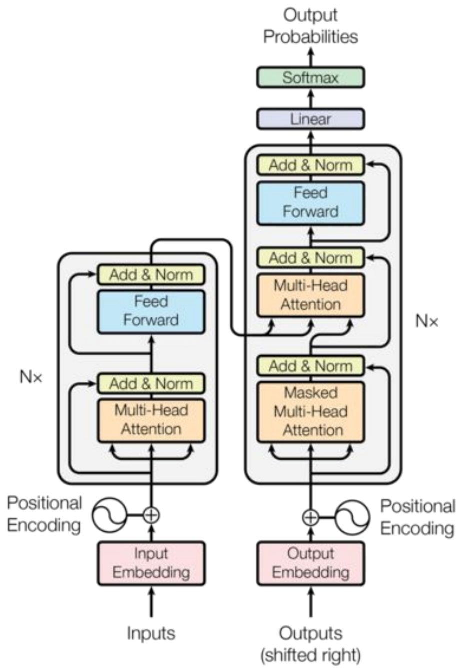




$d^q = d^K = d^v = d/m$

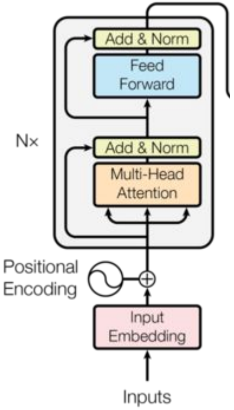
where m is the number of heads

Let’s now try to understand the architecture



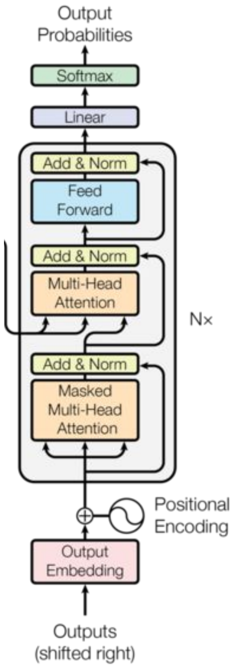
The Transformer Architecture

produces representation of x that captures the meaning in context.



Encoder

The Transformer Architecture

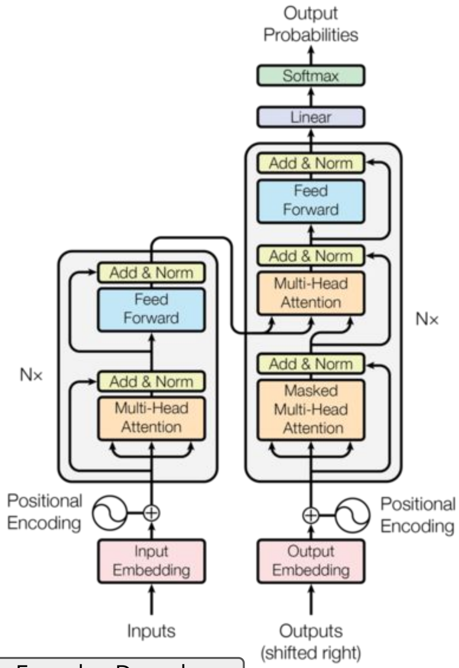


generates tokens step-by-step, $p(y_i | x, y_{<i})$, predicting the next token based on:

- Encoder output (for tasks like translation)
- Previously generated tokens (via self-attention)

Decoder

The Transformer Architecture



Encoder-Decoder

The Annotated Transformer

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

- *v2022: Austin Huang, Suraj Subramanian, Jonathan Sum, Khalid Almubarak, and Stella Biderman.*
- *Original: Sasha Rush.*

Input Tokenization

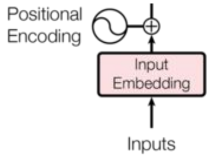
*The **monkey** ate the banana because **it** was hungry.*

[_The] [_mon] [_key] ...

Convert to indices

[723 619 ..]

Encode each entry to a d-dimension vector using the embedding look-up table



Encoder

Positional Encoding - Order matters in language

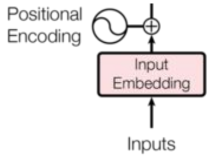
*The **monkey** ate the banana*

vs

*The **banana** ate the monkey*

[0 1 2 ...]

Encode into d-dimension vector and shift input embeddings by this vector



Encoder

Relative Positional Embeddings (Shaw et al., 2018)

Capture the relative distance between key and value pairs

$$
\begin{cases}
q_i = x_i W^Q \\
k_j = x_j W^K + a^K_{j-i} \\
v_j = x_j W^V + a^V_{j-i}
\end{cases}
$$

$$\mathrm{softmax}((x_i W^Q)(x_j W^K + a^K_{j-i})^T)$$

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a^V_{ij})$$

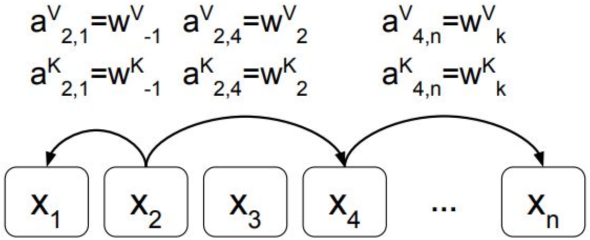
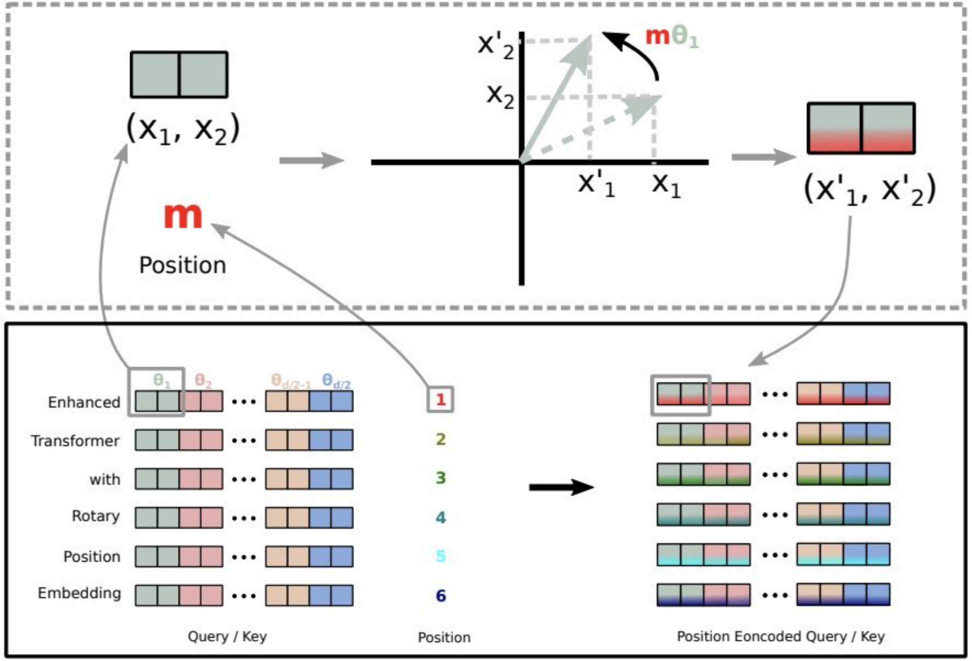


Figure 1: Example edges representing relative positions, or the distance between elements. We learn representations for each relative position within a clipping distance k. The figure assumes $2 <= k <= n - 4$. Note that not all edges are shown.

Rotatory Positional Embeddings (Su et al., 2021)

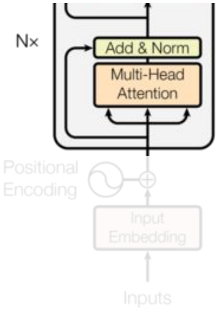


Uses both absolute and relative information

$$f_q(\boldsymbol{x}_m, m) = (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta}$$
$$f_k(\boldsymbol{x}_n, n) = (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta}$$
$$g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) = \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^* e^{i(m-n)\theta}]$$

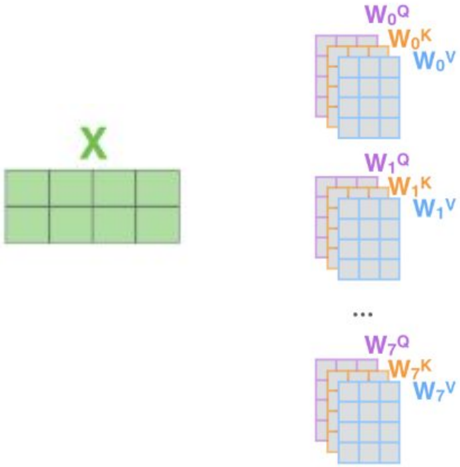
Multi-headed “**Self-attention**”

Encode each input element as Key, Query and Values



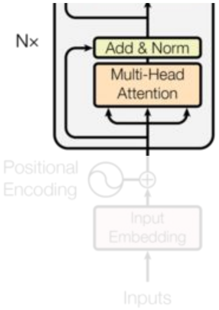
Encoder

mon
key



Multi-headed “**Self-attention**”

Encode each input element as Key, Query and Values



Encoder

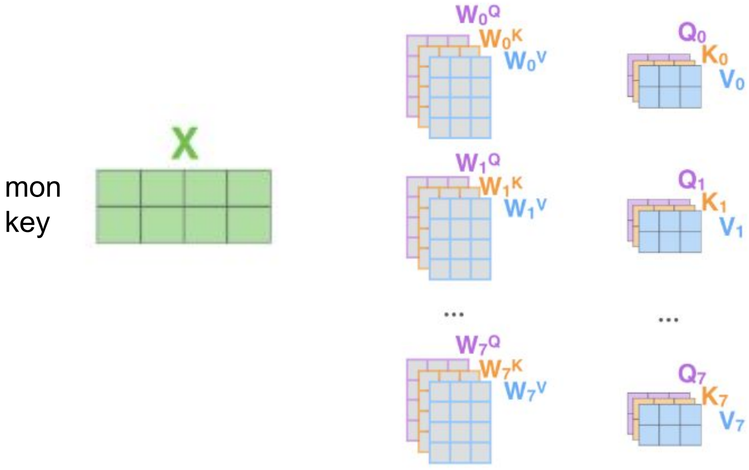
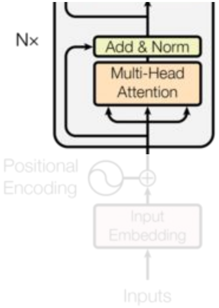


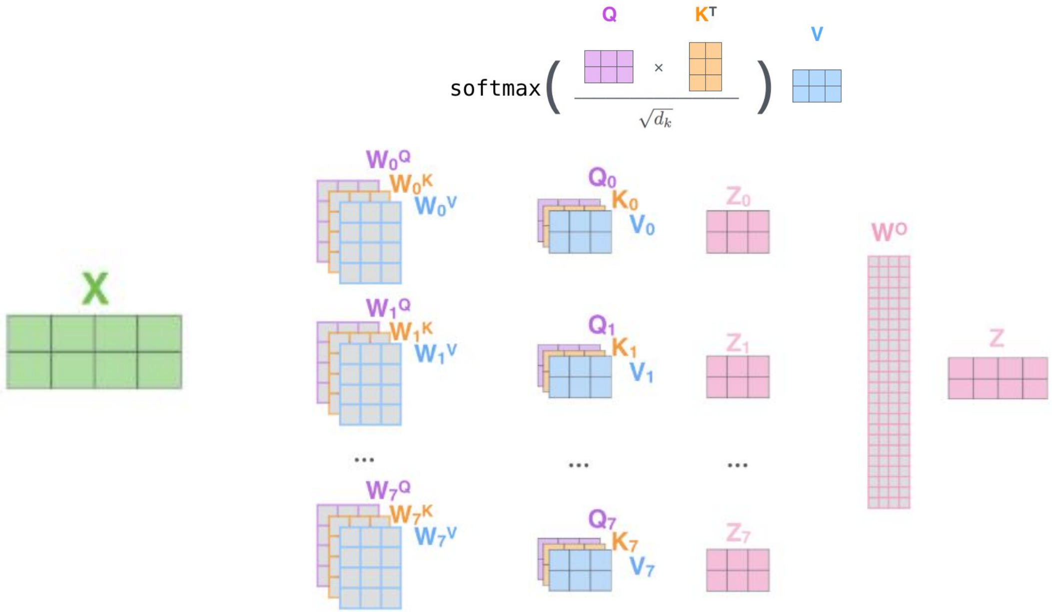
Image Credits: https://jalammar.github.io/illustrated-transformer/

Multi-headed “**Self-attention**”

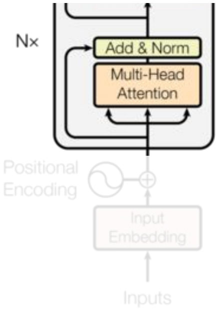


Encoder

mon
key



Multi-headed “**Self-attention**”



Encoder

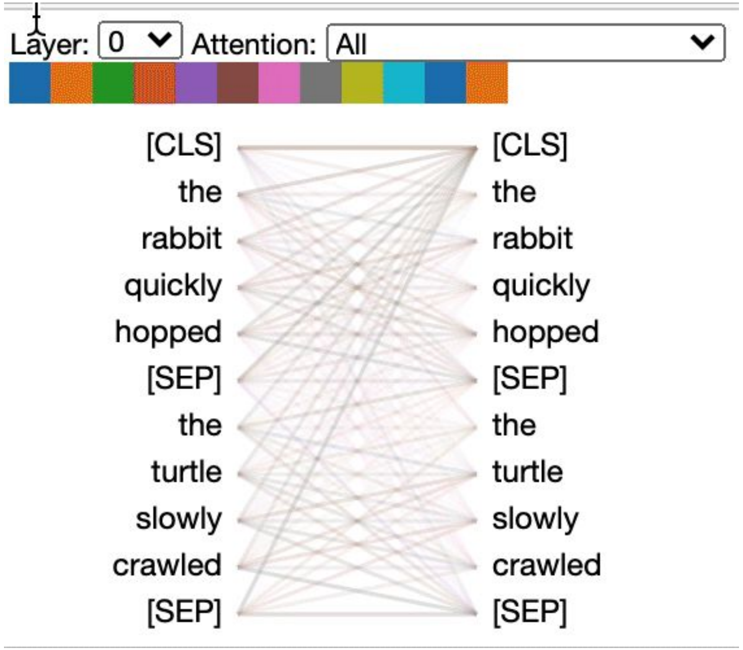
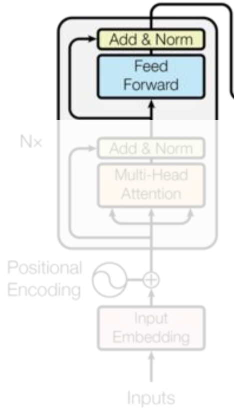


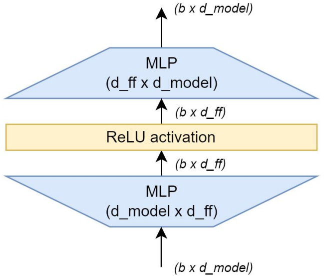
Image Credits: https://jalammar.github.io/illustrated-transformer/
Tensor2Tensor Visualization: https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/visualization/TransformerVisualization.ipynb

Feed Forward - A simple transformation to each token



Encoder

This is where most of the work happens..



$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

d_ff = 4 x d

Feed Forward - A simple transformation to each token

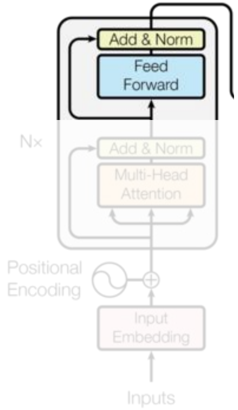
This is where most of the work happens..

Transformer feed-forward layers are key-value memories

M Geva, R Schuster, J Berant, O Levy - arXiv preprint arXiv:2012.14913, 2020 - arxiv.org

… **Feed-forward** layers constitute two-thirds of a **transformer** model's parameters, yet their **role** in the network remains under-explored. We show that **feed-forward** layers in **transformer**…

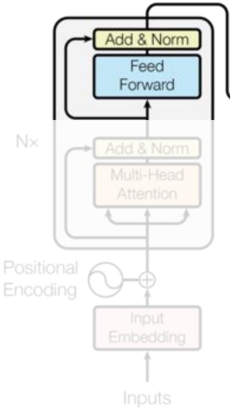
Save Cite Cited by 843 Related articles All 6 versions



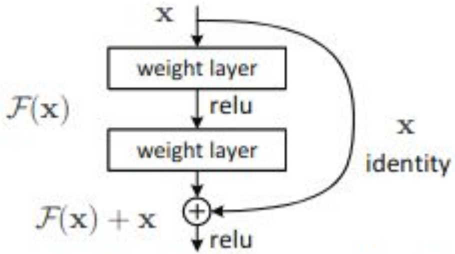
Encoder

Add & Norm

Add: A residual connection (skip connection): Helps gradients flow through deep networks → combats vanishing gradients



Encoder

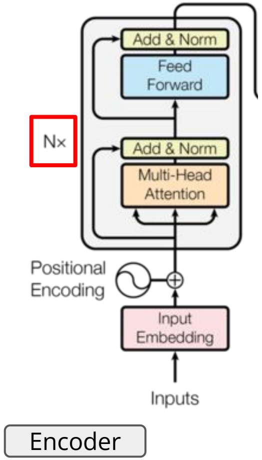


Deep Residual Learning for Image Recognition

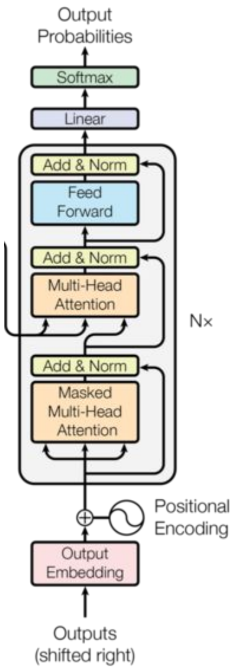
Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Norm: A Layer Normalization which keeps output scale consistent, → speeds up and stabilizes training [Pre- Vs Post- Norm]

The Encoder: heavily processed version of “input”



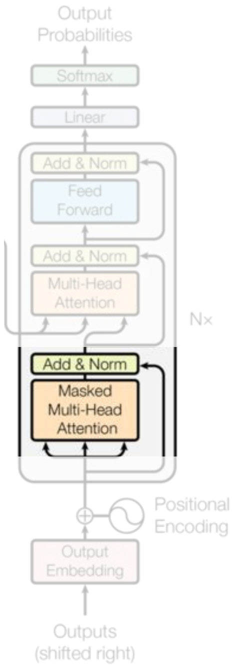
The Decoder



generates tokens step-by-step, predicting the next token probability $p(y_i|x, y_{<i})$ based on:

Decoder

The Decoder

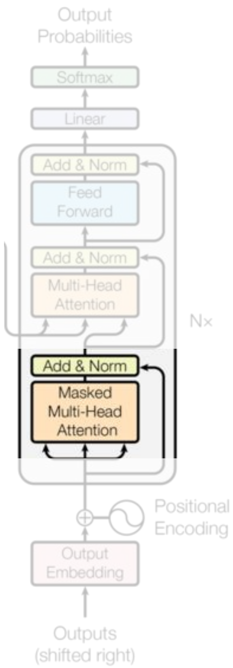


generates tokens step-by-step, predicting the next token probability $p(y_i | x, y_{<i})$ based on:

- Previously generated tokens (via masked self-attention)

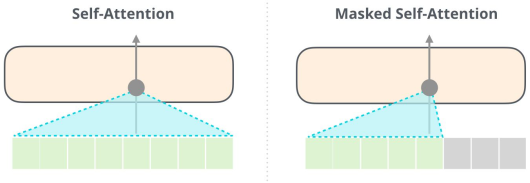
Decoder

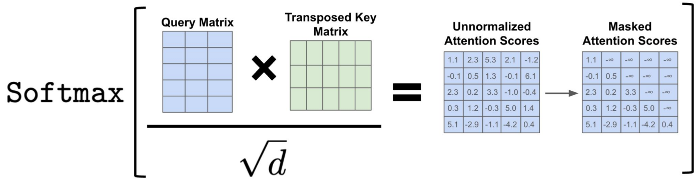
The Decoder - Masked Multi-head self-attention



generates tokens step-by-step, predicting the next token probability $p(y_i | x, \mathbf{y}_{<i})$ based on:

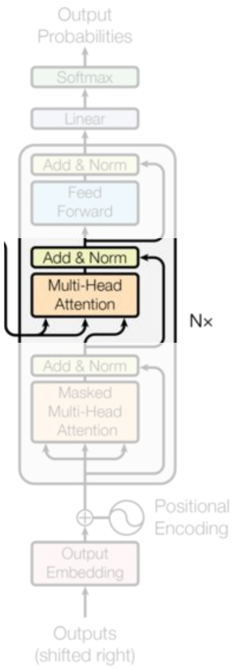
- **Previously generated tokens**





Decoder

The Decoder



generates tokens step-by-step, predicting the next token probability $p(y_i | x, y_{<i})$ based on:

- Previously generated tokens (via masked self-attention)
- **Encoder output (for tasks like translation)**

keys and values - derived from the hidden states of the encoder x

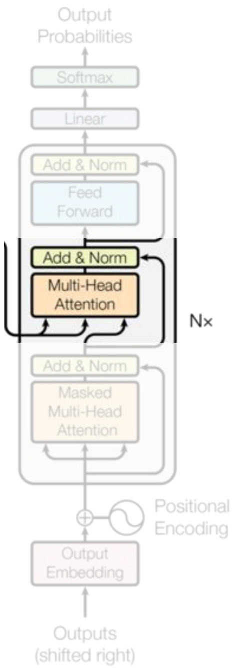
$k = W_k \cdot h_x \quad v = W_v \cdot h_x$

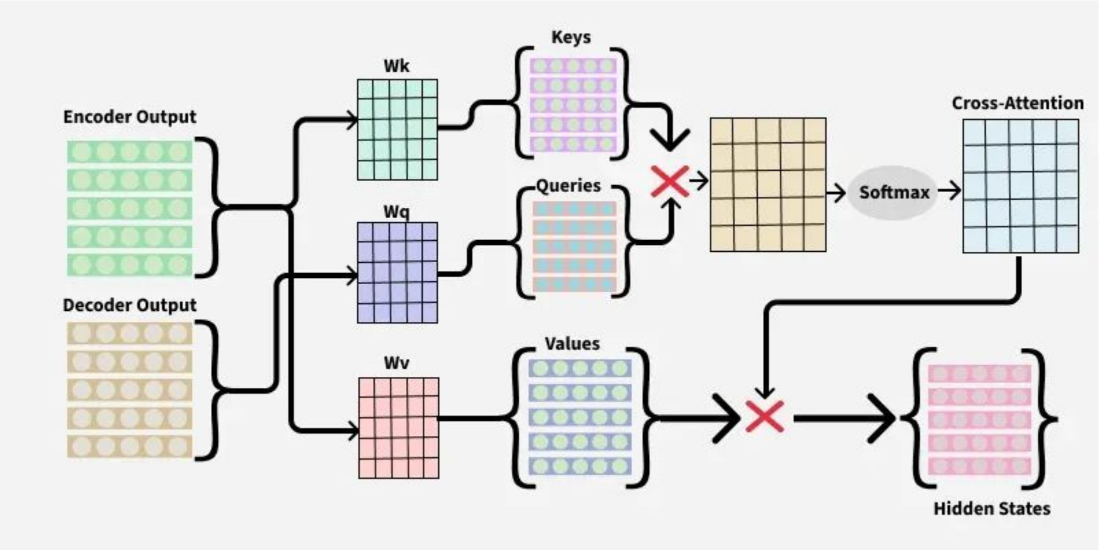
queries are from hidden state of the decoder

$q = W_q \cdot h_y$

Decoder

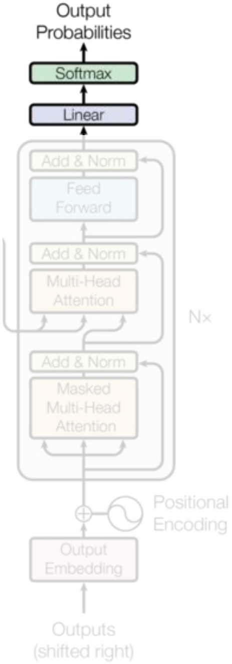
The Decoder - Cross Attention



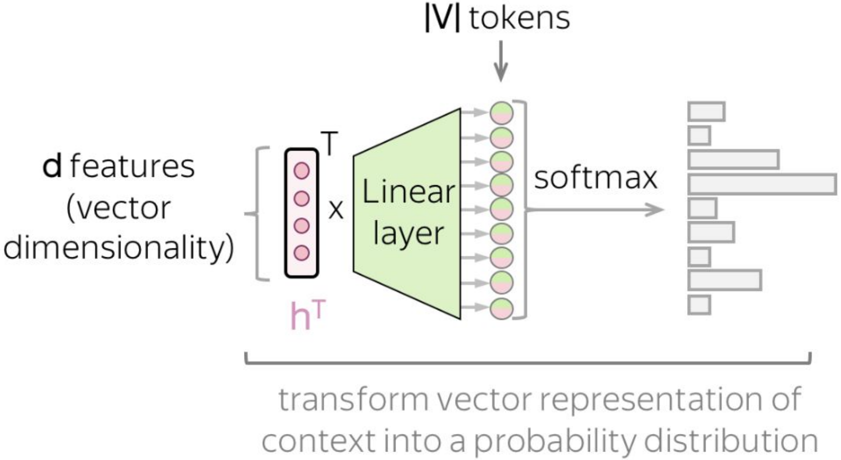


Decoder

The Decoder

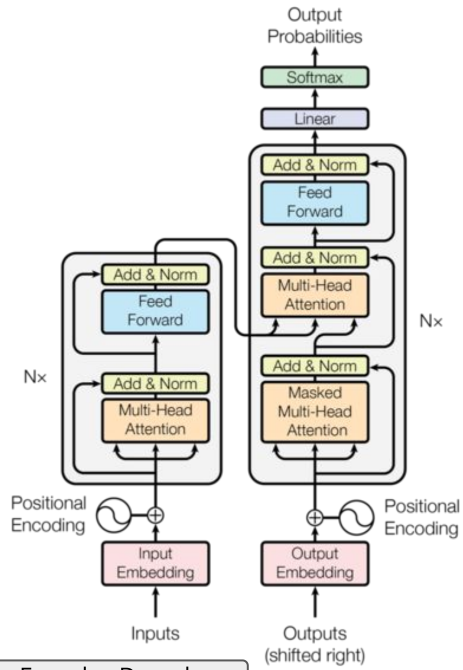


predict the next token probability $p(y_i | x, y_{<i})$



Decoder

Training with Cross-entropy loss



Encoder-Decoder

we want the model to predict this

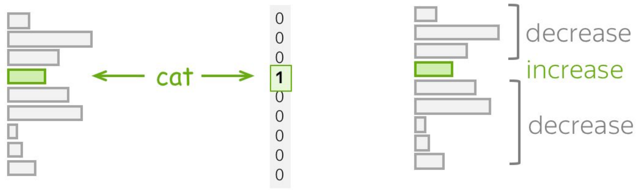
Training example: **I saw a** cat on a mat <eos>

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\hat{y}_i)$$

Model prediction: p(* | **I saw a**)

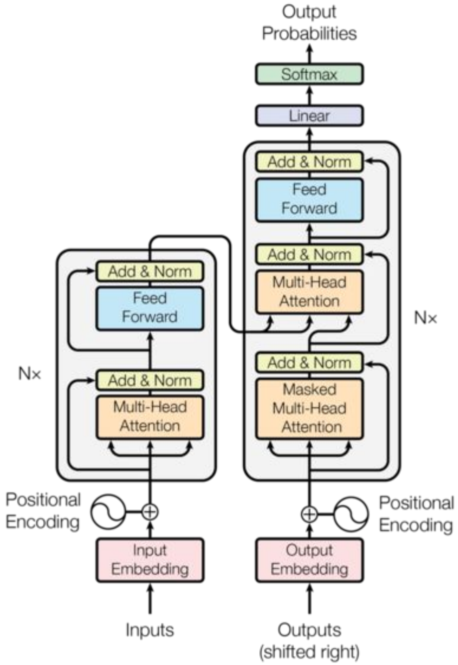
Target

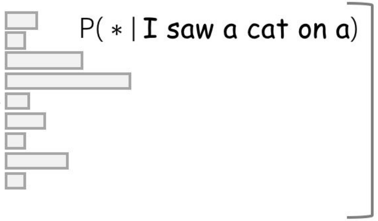
Loss = -log (p(**cat**)) → min



https://poloclub.github.io/transformer-explainer/

At generation time - One token at a time

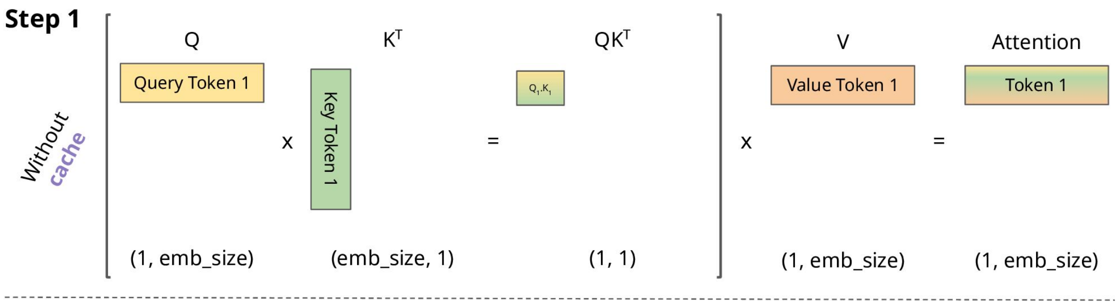




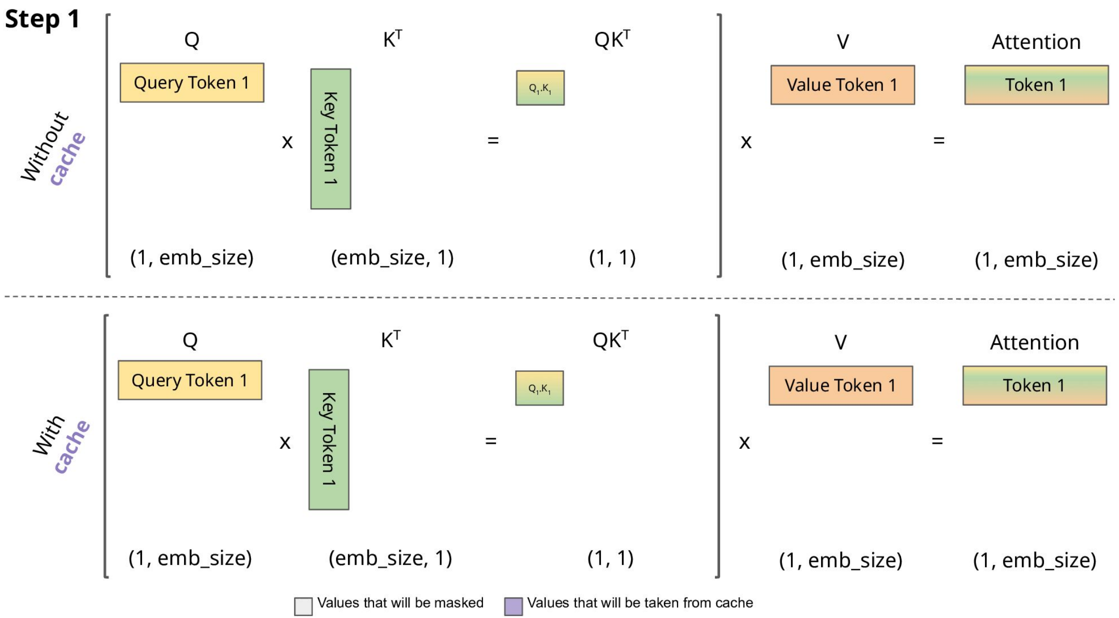
get probability distribution for the next token

$$\hat{y}_t = \arg\max_{y \in V} P(y \mid y_{<t})$$

At generation time - One token at a time

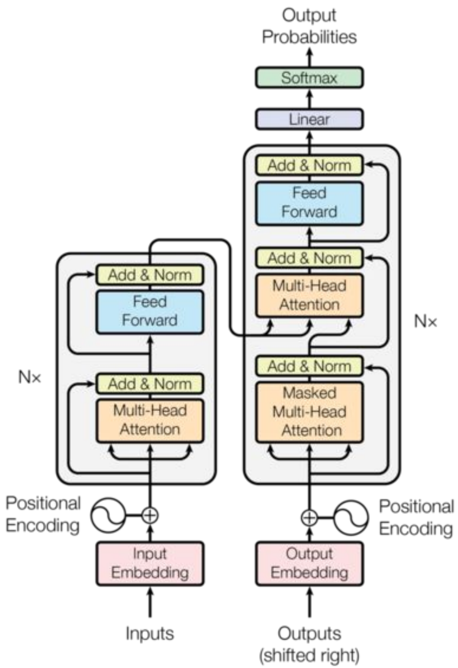


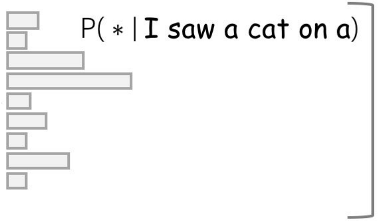
KV cache



https://huggingface.co/blog/not-lain/kv-caching

At generation time - One token at a time

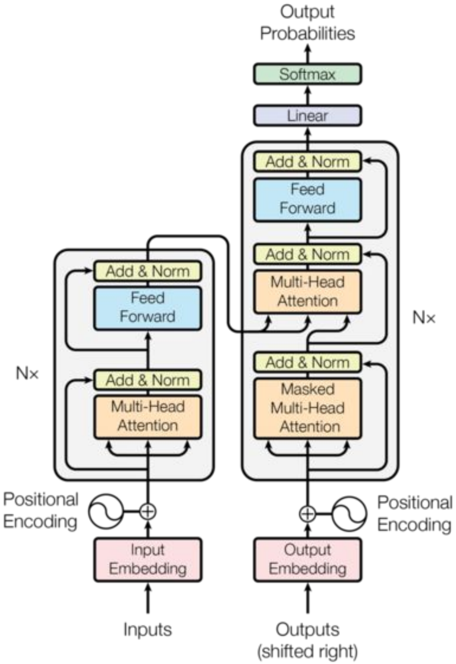


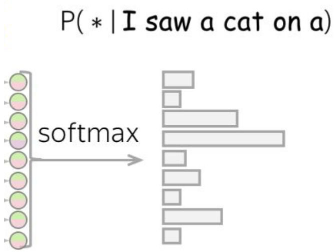


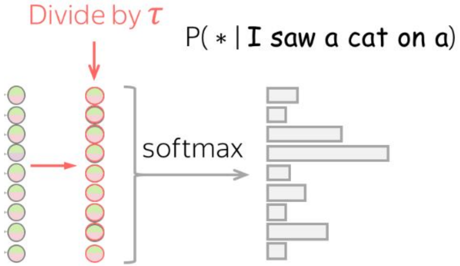
get probability distribution for the next token

$$\hat{y}_t = \arg\max_{y \in V} P(y \mid y_{<t})$$

At generation time - One token at a time [Sampling]

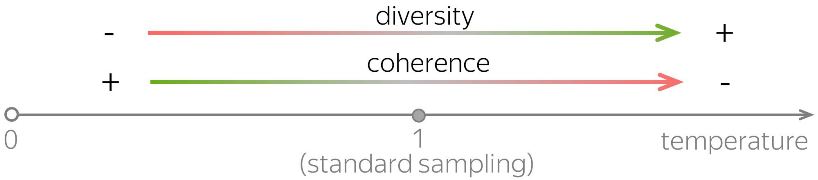




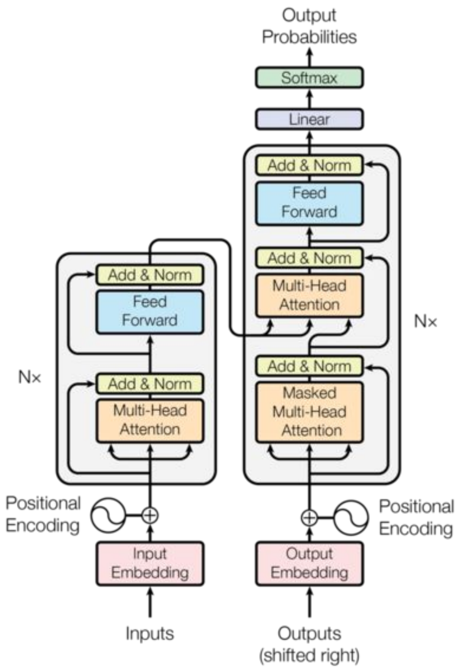


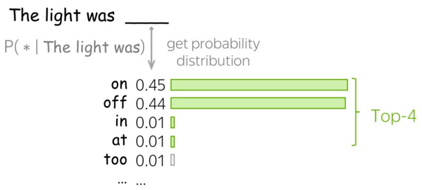
$$\frac{\exp(h^T w)}{\sum_{w_i \in V} \exp(h^T w_i)} \rightarrow \frac{\exp\left(\frac{h^T w}{\tau}\right)}{\sum_{w_i \in V} \exp\left(\frac{h^T w_i}{\tau}\right)}$$

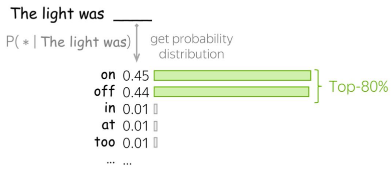
τ - softmax temperature



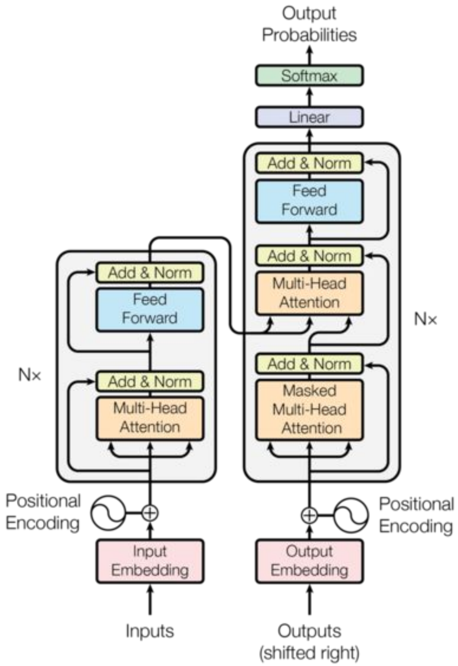
At generation time - One token at a time [Topk, Top-p]



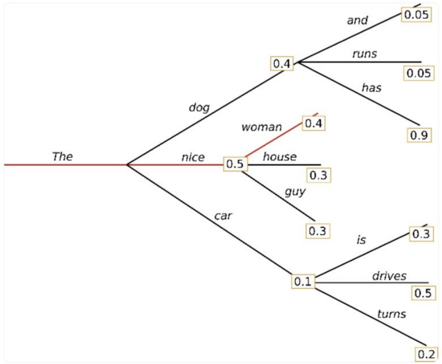




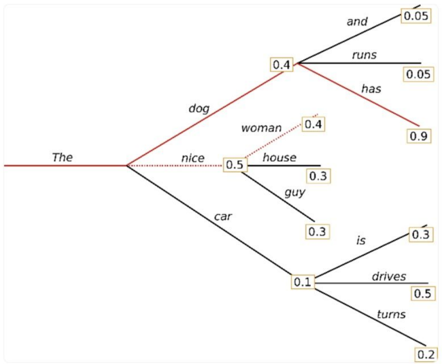
At generation time - Search [Beam]



Greedy



Beam



Transformer outperforms many diverse architecture and is efficient

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Transformer outperforms many diverse architecture and is efficient

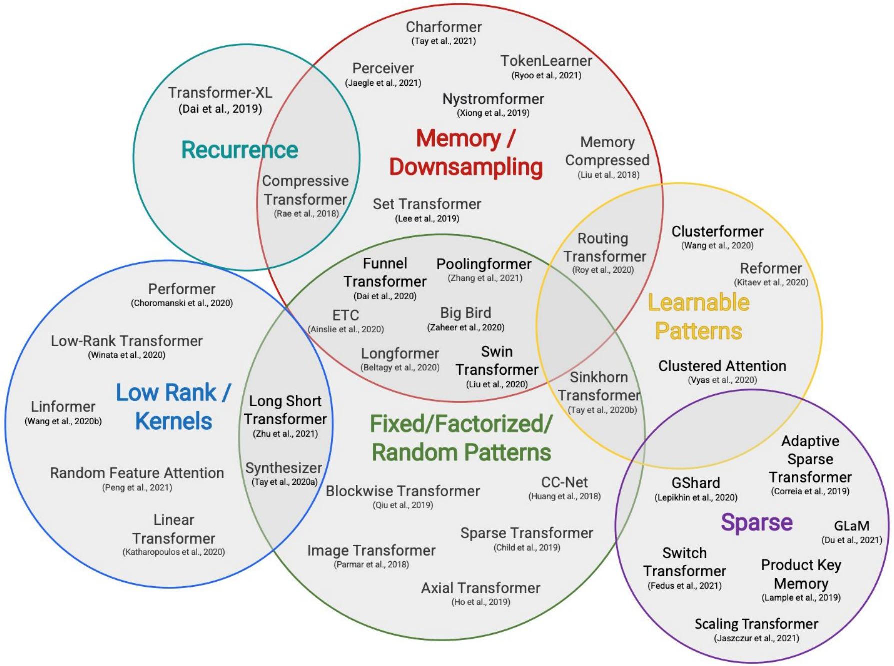
Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

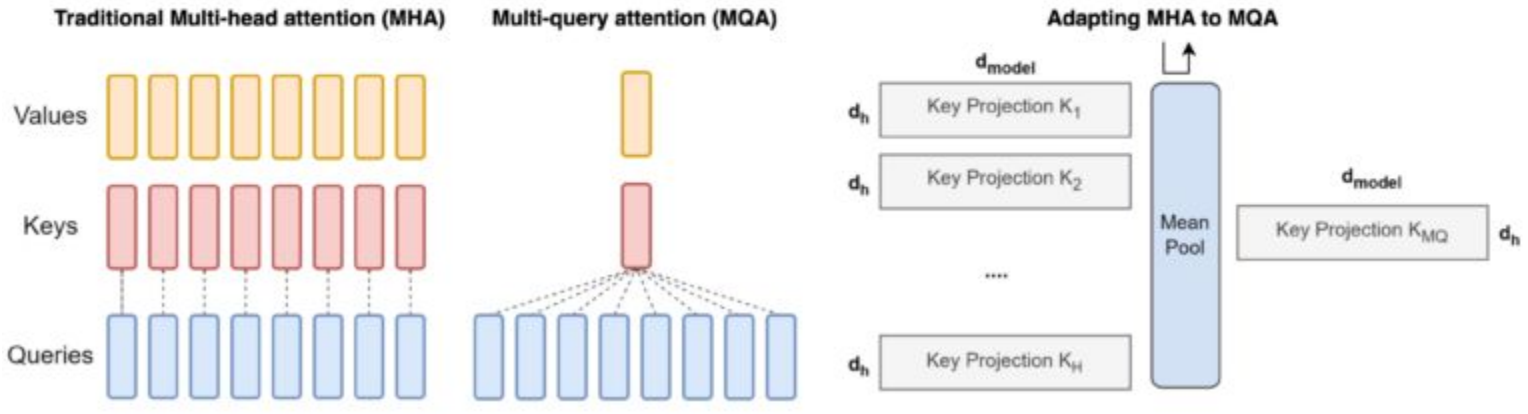
Is it truly efficient?

$$O(nd^2+n^2d) + O(nd^2)$$

Efficient Transformers



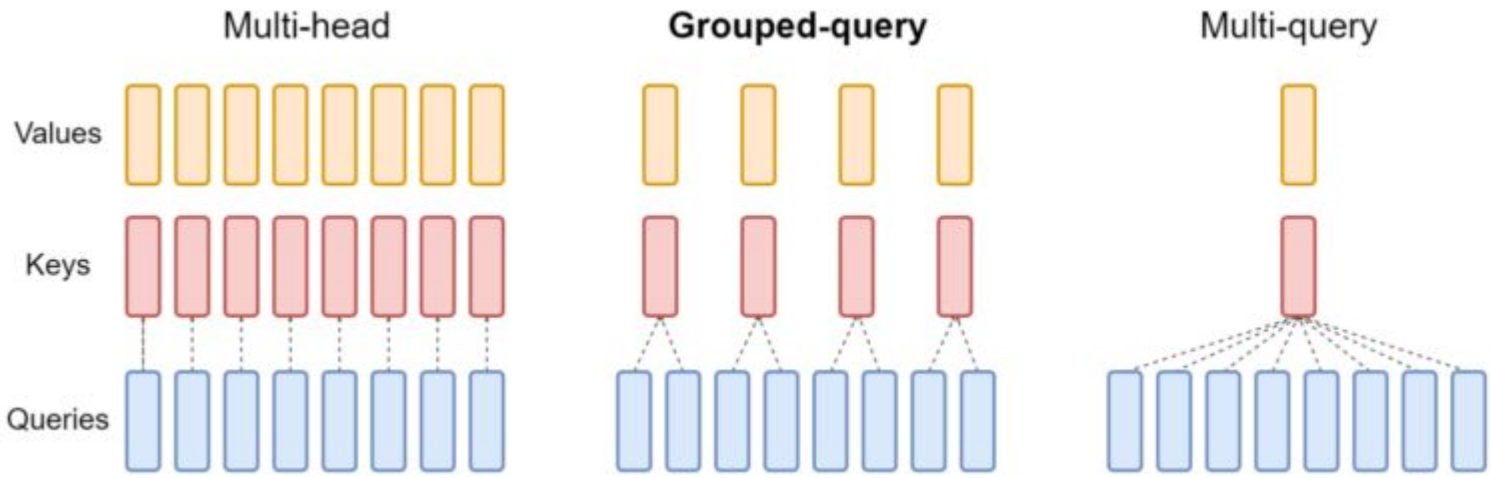
The Inference Bottleneck: The KV Cache Size
Introduce: Multi-Query Attention



In multi-query attention, the heads for keys and values are averaged so that all query heads share the same key and value head.

Fast Transformer Decoding: One Write-Head is All You Need [Shazeer et al., 2019]

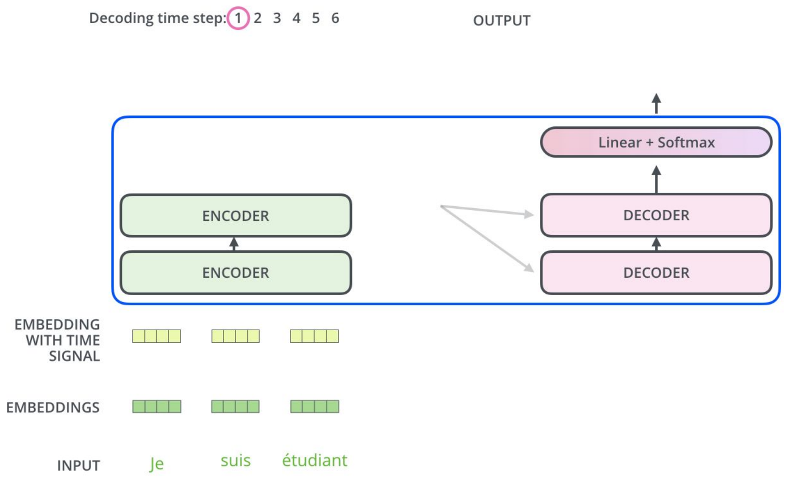
The Inference Bottleneck: The KV Cache Size Introduce: Grouped-Query Attention



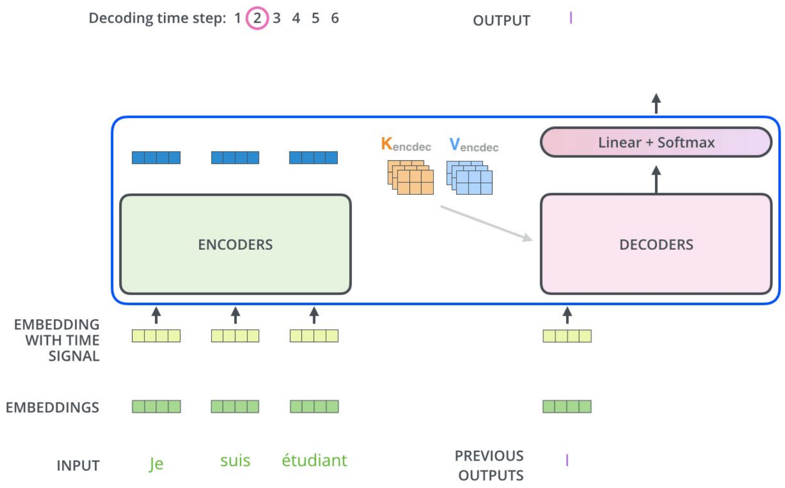
In GQA instead of all heads sharing one K/V pair, a few heads share a K/V pair.

GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints [Ainslie et al., 2023]

Summary Visualization



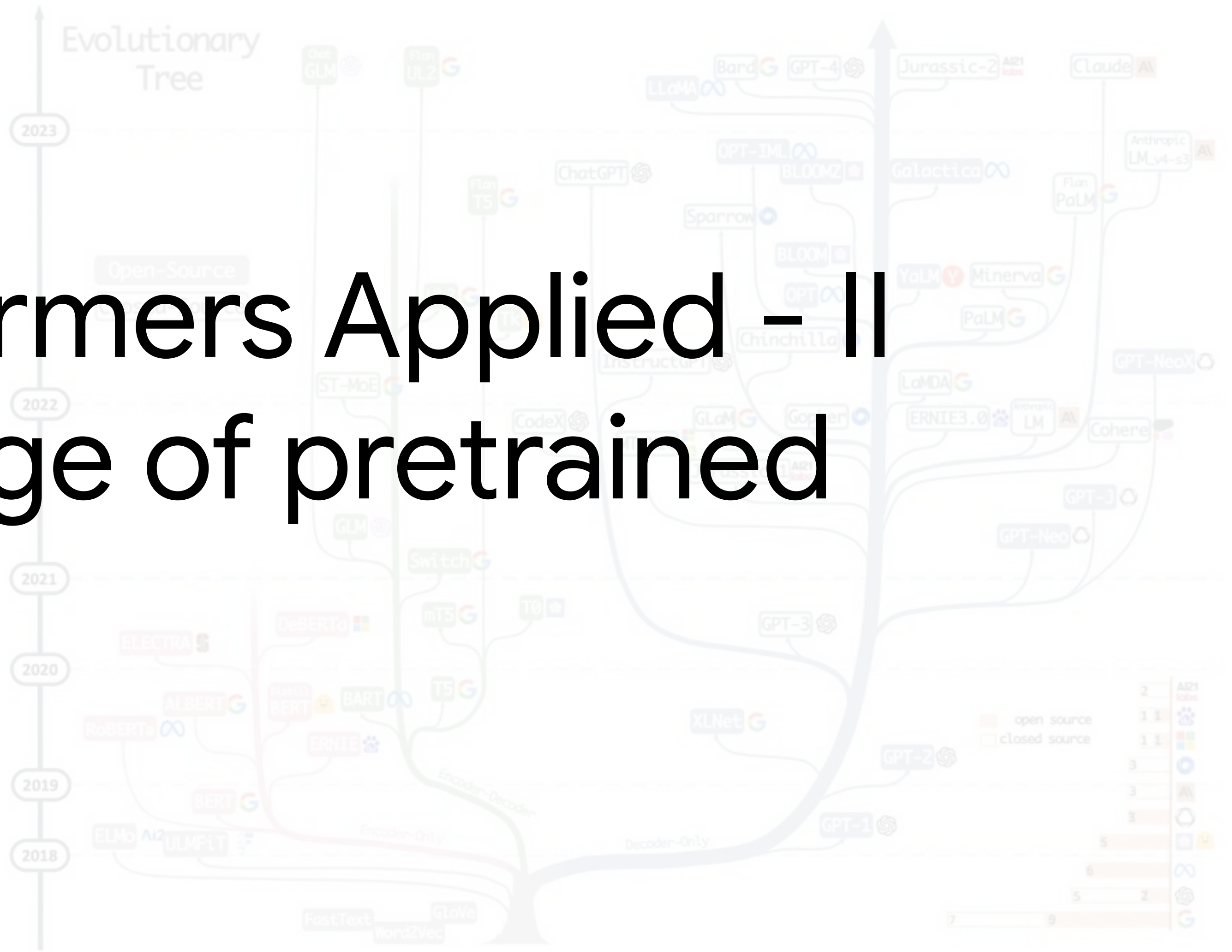
Encoder



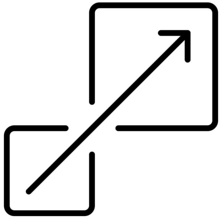
Decoder

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

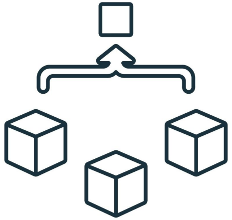
Transformers Applied - II
The surge of pretrained models



Change of Paradigm
What did transformers enable?



Scalability



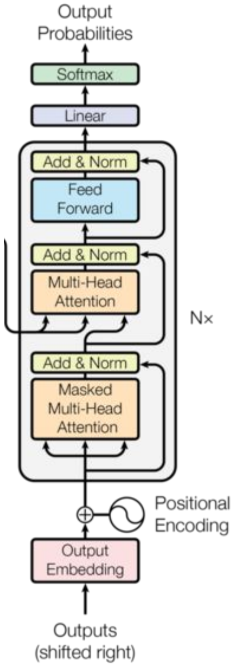
Generalizability

Reasoning

Long range

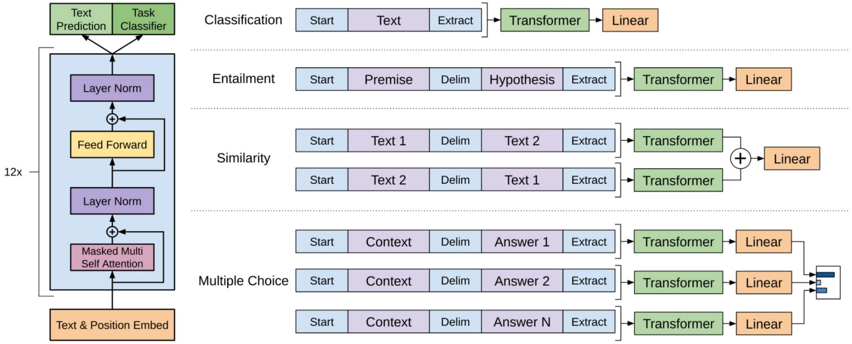
Multimodal

GPT-1: Generative Pre-trained Transformer Architecture



- 117 million parameters
- decoder-only
- trained on the Common Crawl, a massive dataset of web pages with billions of words, and the BookCorpus dataset, a collection of over 11,000 books on a variety of genres.

GPT-1: Discriminative Finetuning



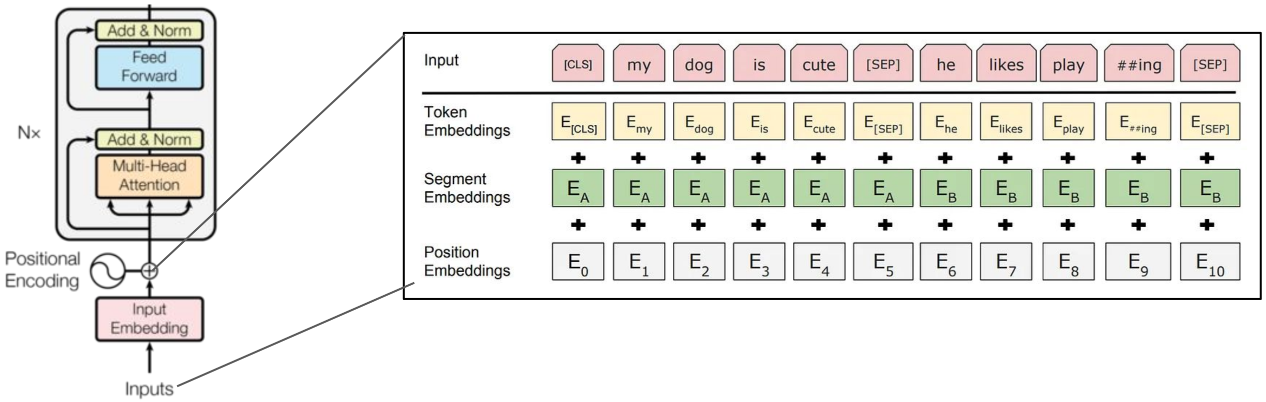
Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned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

✅ Unsupervised Pre-training Works.
✅ A Single Model Can Generalize.

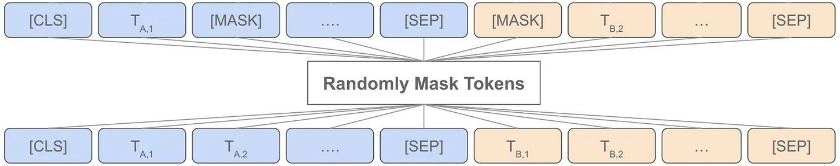
Enter BERT: Bidirectional Encoder Representations from Transformers

Introduction of deeply “contextualized word embeddings”

Pre-training over a large amount of “text” data

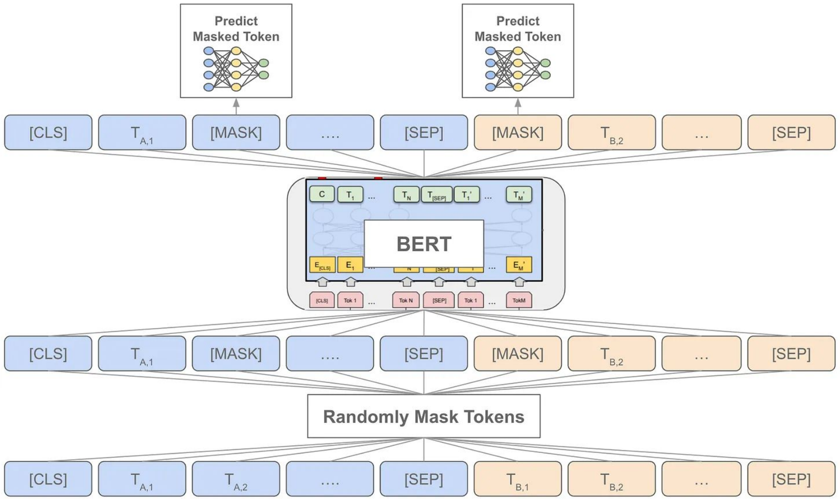


BERT Training



Masked Language Prediction

BERT Training: Masked Language Modeling

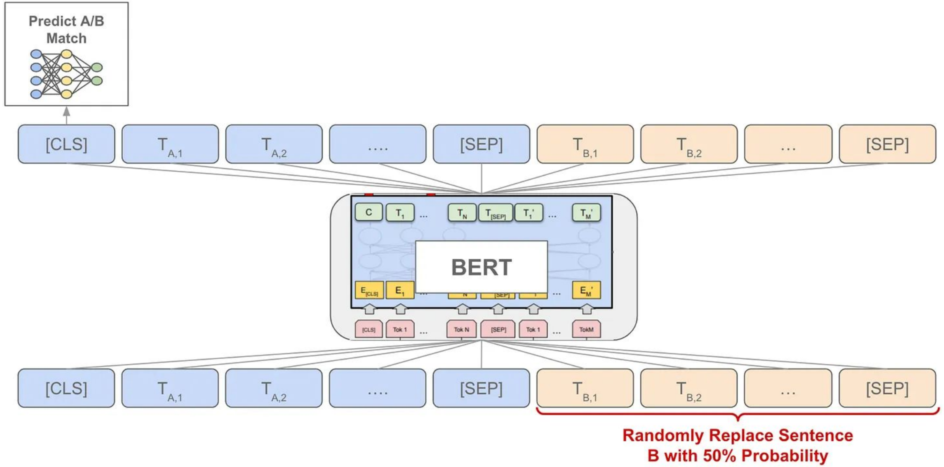


Masked Language Prediction

select 15% of the input tokens and follow the 80/10/10 rule:

- 80% of the time with the "[MASK]" token.
- 10% of the time with a random token from the vocabulary.
- 10% of the time with the original token itself.

BERT Training: Next Sentence prediction



Next Sentence Prediction

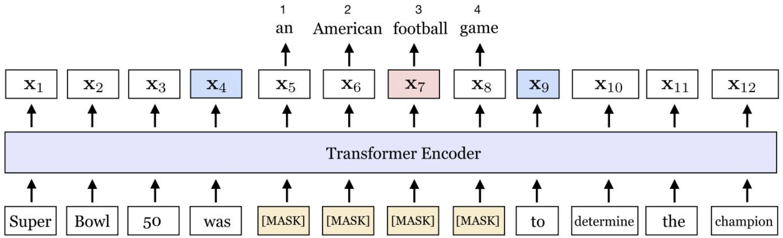
Sample two segments and predict whether B follows A

- 50% of the time sample a text segment of 512 tokens (Yes)
- 50% of time a segment of 256 tokens followed by unrelated text segment of 256 tokens (No)

NSP is unnecessary (Joshi et al., 2019, Liu et al. 2019)

What to mask and how much to mask?

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$

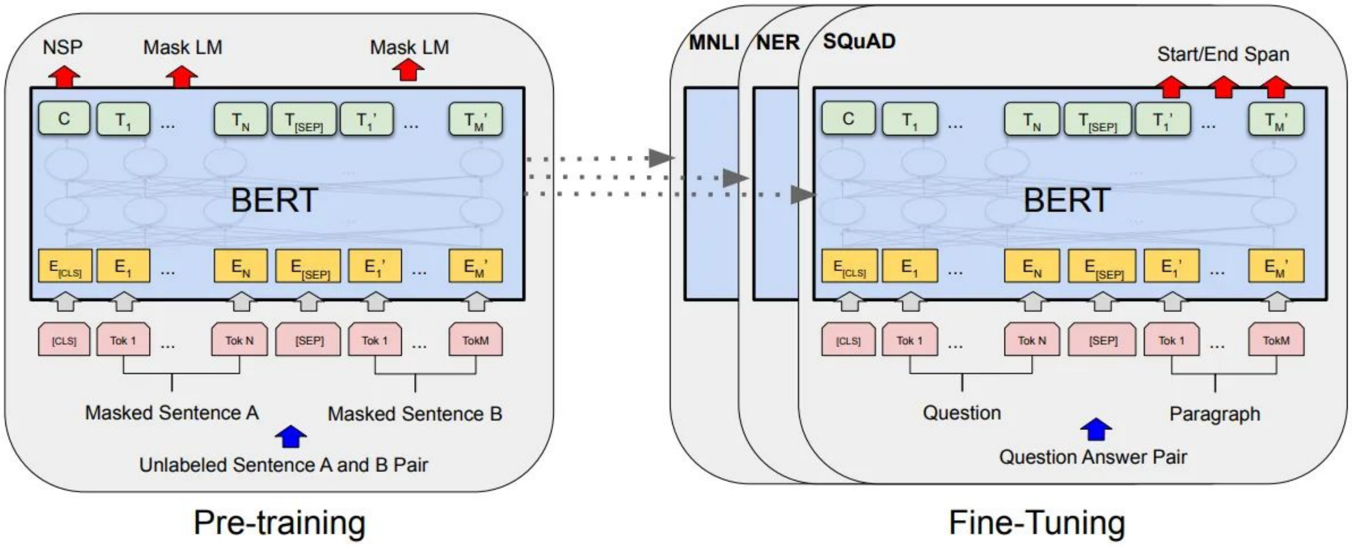


	Pre-training		Fine-tuning		
m	Example	PPL	MNLI	QNLI	SQuAD[3]
15%	We study high ing rates pre-training language models .	17.7	84.2	90.9	88.0
40%	We study high rates pre- models .	69.4	**84.5** ↑0.3	**91.6** ↑0.7	**89.8** ↑1.8
80%	We high models	1141.4	80.8 ↓3.4	87.9 ↓3.0	86.2 ↓1.8
	Random initialization		61.5 ↓22.7	60.9 ↓30.0	10.8 ↓77.2

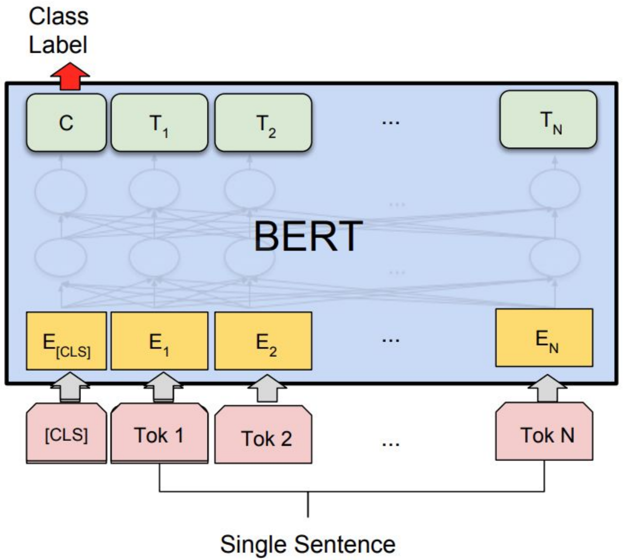
SpanBERT (Joshi et al., 2019)

Wettig et al., 2023

Using BERT for downstream tasks



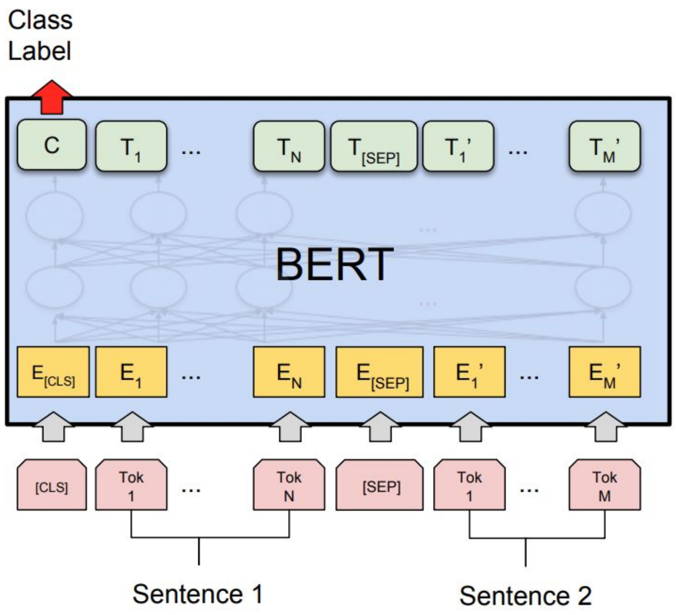
Using BERT for downstream tasks



🧾 **Sentence/Document-Level Tasks**

- Use **[CLS] token** as the **summary** of the input.
- Pass it to a classifier (d x |C|) for **overall prediction**. *E.g.,* Sentiment Analysis.

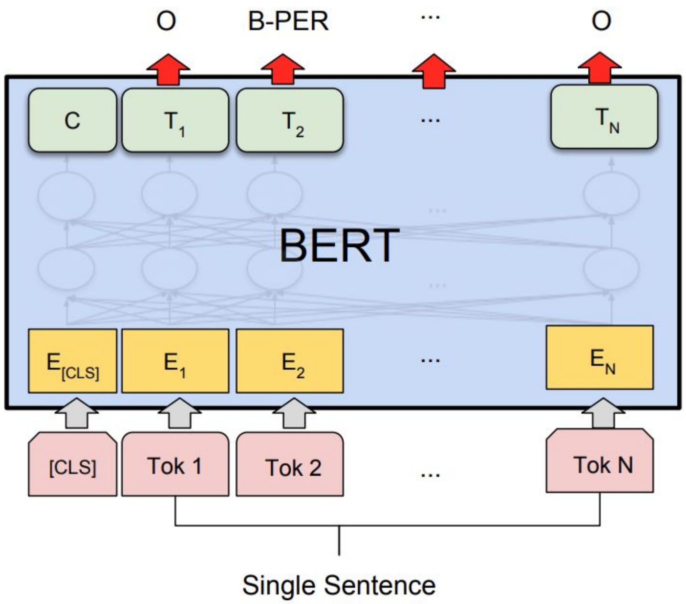
Using BERT for downstream tasks



🔁 **Text Pair Tasks**

- Format input as:
  `[CLS] Sentence A [SEP] Sentence B [SEP]`
- Use **[CLS] token output** for final prediction.
  E.g., Natural Language Inference.

Using BERT for downstream tasks



🧩 **Token-Level Tasks**

- Use **each token's output** from BERT
- Apply a classifier **on each token**
- *E.g.,* Named Entity Recognition.

BERT variants

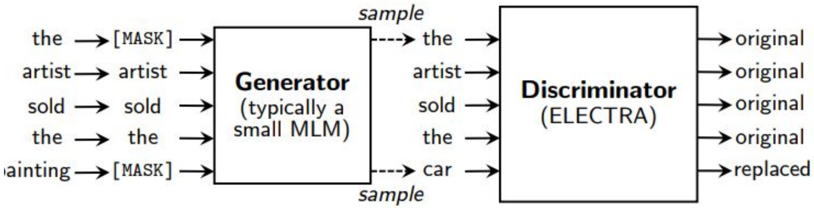
RobertA

- Dynamic masking: recompute masks at each epoch
- use 160GB data instead of 16GB

DistilBERT - distil inform from a “big” teacher model to a “small” student model

ELECTRA

- generator/discriminator framework
- more efficient



BERTology

What Knowledge Does BERT Have? Syntactic, Semantic or World

What can we learn from looking at its attention heads?

What can we learn about training (efficient) BERT?

A Primer in BERTology: What We Know About How BERT Works

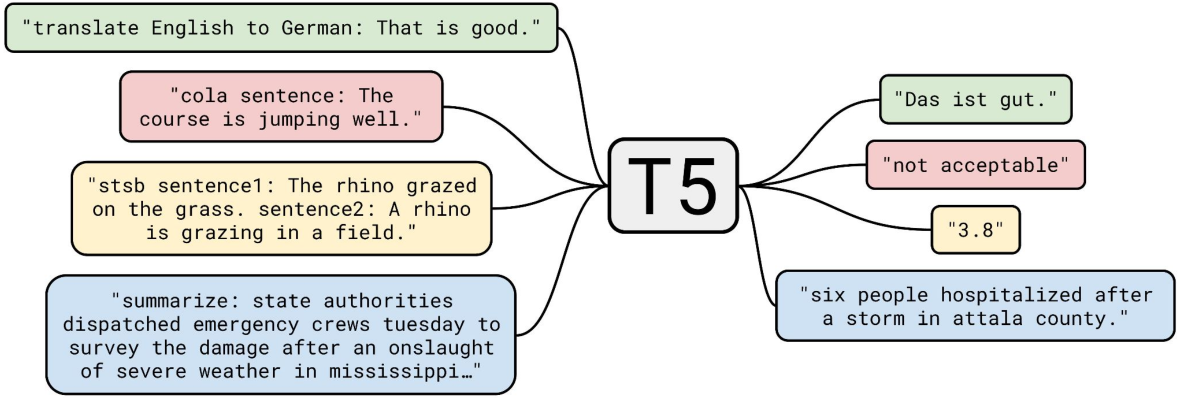
Anna Rogers
Center for Social Data Science
University of Copenhagen
arogers@sodas.ku.dk

Olga Kovaleva
Dept. of Computer Science
University of Massachusetts Lowell
okovalev@cs.uml.edu

Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

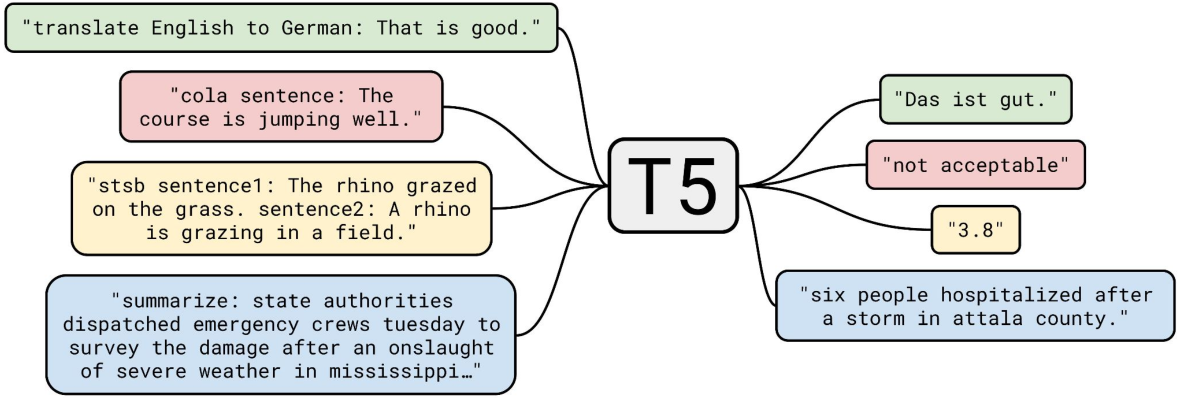
T5 Architecture (Raffel et al., 2019)

“ Convert every task — classification, summarization, translation, QA — into a text generation task. ”

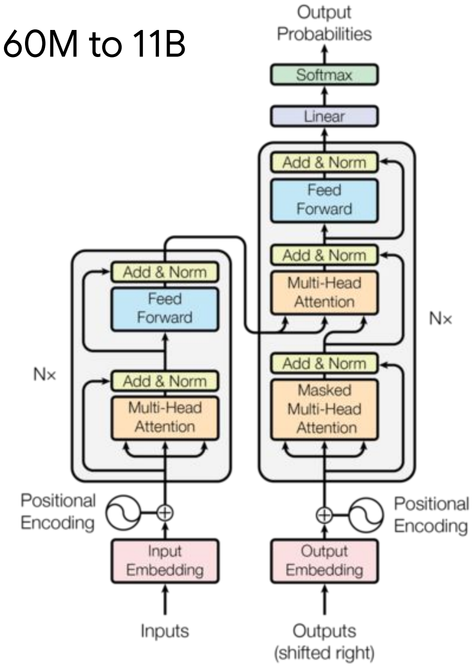


T5 Architecture (Raffel et al., 2019)

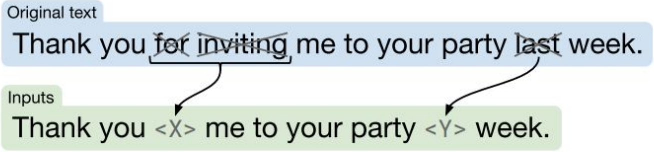
❝ Convert every task — classification, summarization, translation, QA — into a text generation task. ❞



60M to 11B



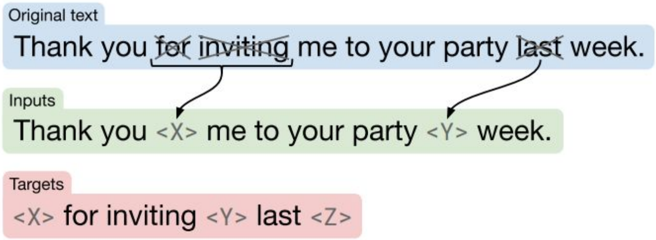
T5 Training (Raffel et al., 2019)



C4

The C4 dataset we created for unsupervised pre-training is available in TensorFlow Datasets, but it requires a significant amount of bandwidth for downloading the raw Common Crawl scrapes (~7 TB) and compute for its preparation (~335 CPU-days). We suggest you take advantage of the Apache Beam support in TFDS, which enables distributed preprocessing of the dataset and can be run on Google Cloud Dataflow. With 500 workers, the job should complete in ~16 hours.

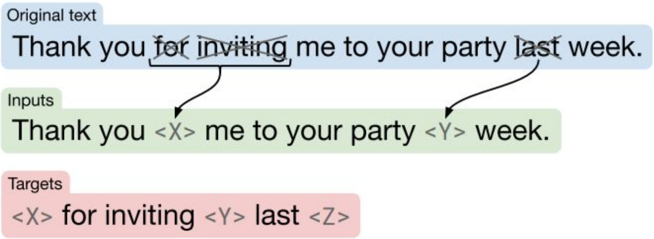
T5 Training (Raffel et al., 2019)



C4

The C4 dataset we created for unsupervised pre-training is available in TensorFlow Datasets, but it requires a significant amount of bandwidth for downloading the raw Common Crawl scrapes (~7 TB) and compute for its preparation (~335 CPU-days). We suggest you take advantage of the Apache Beam support in TFDS, which enables distributed preprocessing of the dataset and can be run on Google Cloud Dataflow. With 500 workers, the job should complete in ~16 hours.

T5 Training (Raffel et al., 2019)



C4

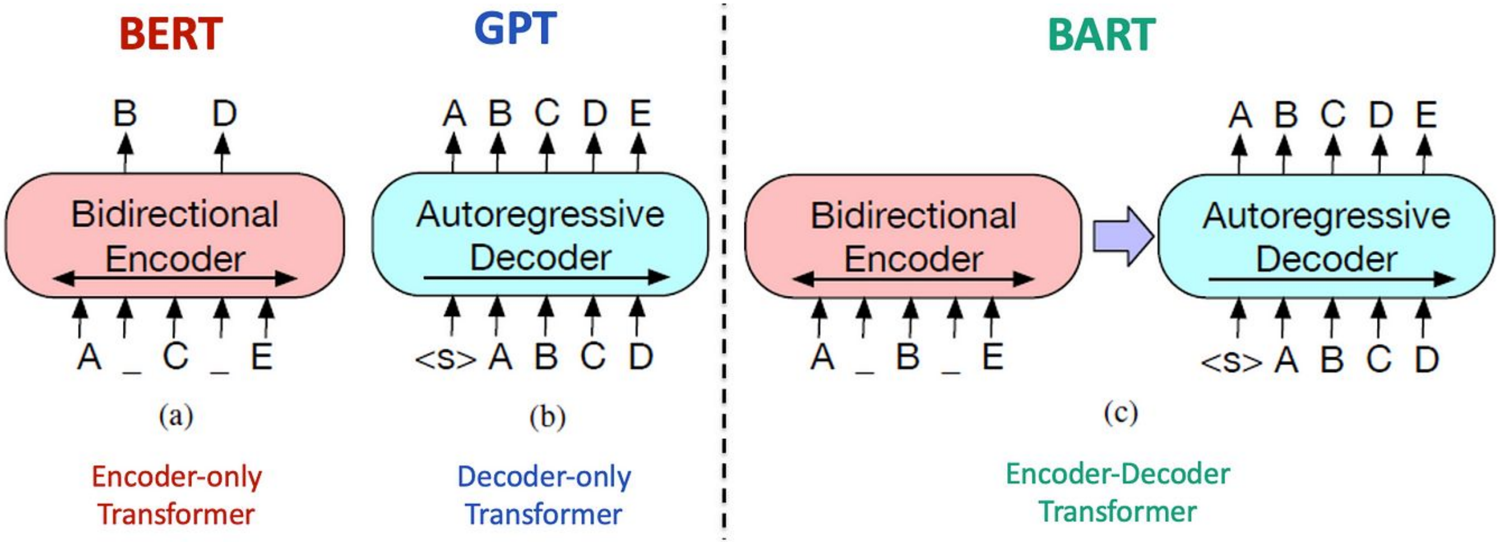
The C4 dataset we created for unsupervised pre-training is available in TensorFlow Datasets, but it requires a significant amount of bandwidth for downloading the raw Common Crawl scrapes (~7 TB) and compute for its preparation (~335 CPU-days). We suggest you take advantage of the Apache Beam support in TFDS, which enables distributed preprocessing of the dataset and can be run on Google Cloud Dataflow. With 500 workers, the job should complete in ~16 hours.

Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you <M> <M> me to your party <M> week .	*(original text)*
I.i.d. noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you <X> to <Y> week .	<X> for inviting me <Y> your party last <Z>

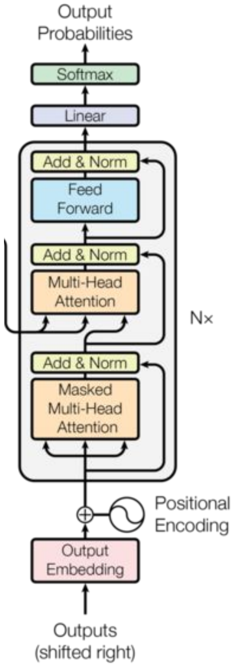
Impact of T5

T5

BART Model (Lewis et al., 2019)



The GPT-3 Era



Scale Is All You Need: Model with 175 billion parameters, trained on a broad corpus of web text

General-Purpose Model: One model, many tasks — without task-specific training

Prompt-Based Learning: Shifted NLP from fine-tuning to in-context learning

Language Models are Few-Shot Learners [Brown et al., 2020]

The GPT-3 Era

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

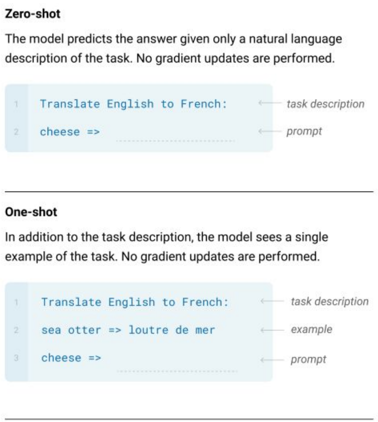
```
Translate English to French:
cheese =>
```

task description

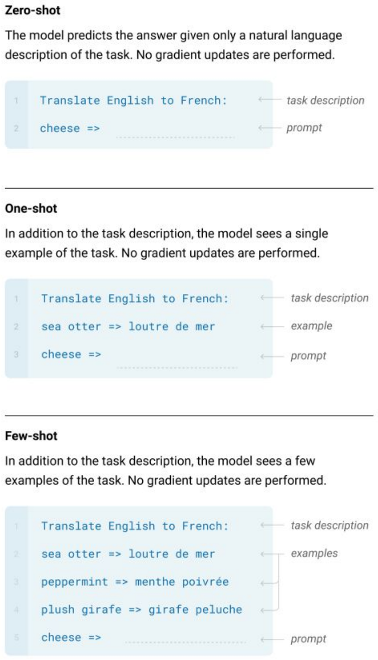
prompt

Language Models are Few-Shot Learners [Brown et al., 2020]

The GPT-3 Era


Zero-shot
The model predicts the answer given only a natural language description of the task. No gradient updates are performed.
Translate English to French:
task description
cheese =>
prompt
One-shot
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.
Translate English to French:
task description
sea otter => loutre de mer
example
cheese =>
prompt

The GPT-3 Era


Zero-shot
The model predicts the answer given only a natural language description of the task. No gradient updates are performed.
Translate English to French:
task description
cheese =>
prompt
One-shot
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.
Translate English to French:
task description
sea otter => loutre de mer
example
cheese =>
prompt
Few-shot
In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.
Translate English to French:
task description
sea otter => loutre de mer
examples
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
prompt

The GPT-3 Era

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1 Translate English to French:      ← task description
2 cheese => ..........              ← prompt
```

One-shot

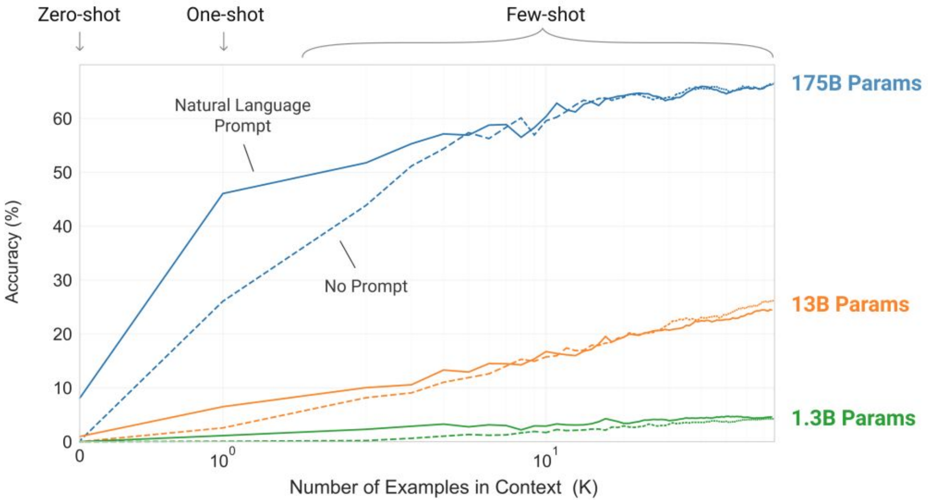
In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1 Translate English to French:      ← task description
2 sea otter => loutre de mer        ← example
3 cheese => ..........              ← prompt
```

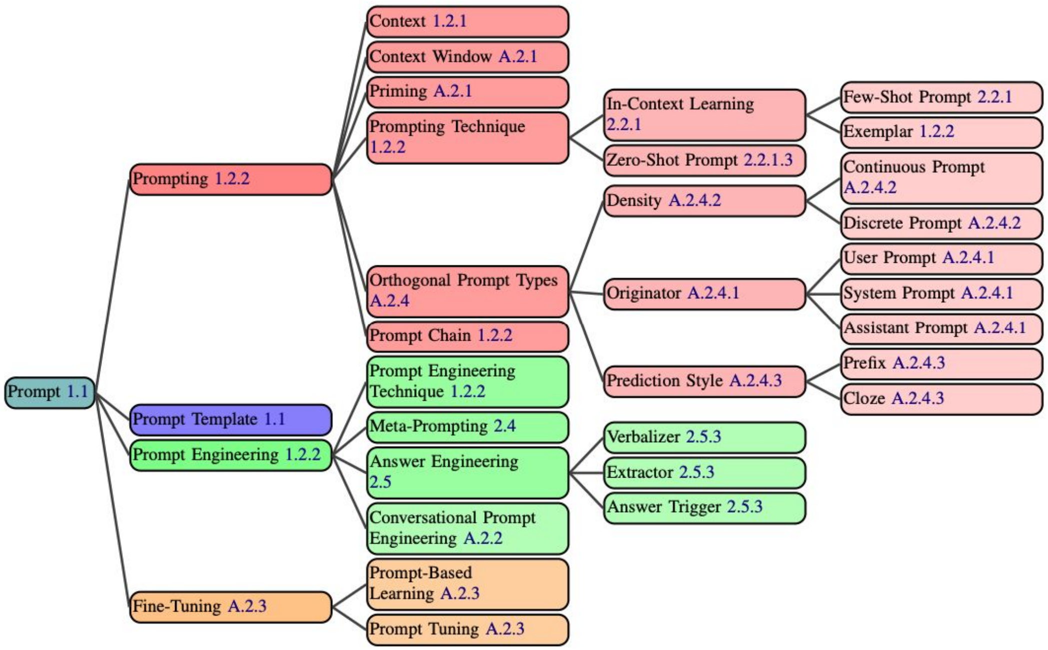
Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1 Translate English to French:      ← task description
2 sea otter => loutre de mer        ← examples
3 peppermint => menthe poivrée      ←
4 plush girafe => girafe peluche    ←
5 cheese => ..........              ← prompt
```



The Prompt Report: A Systematic Survey of Prompt Engineering Techniques (Schulhoff et al., 2024)



What makes in-context learning work?

➔ Task-recognition

The prompt Translate English to French. sea otter => loutre de mer... acts as a query.

➔ Task-learning

It then activates the specific neural pathways related to the "translation" skill and applies that learned pattern to your new input (cheese => fromage).

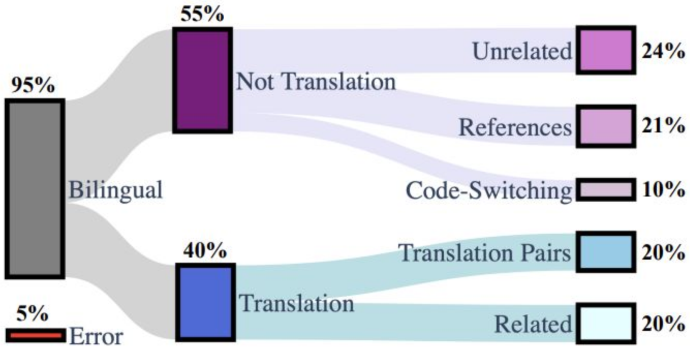
Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, Luke Zettlemoyer

What makes in-context learning work?

Searching for Needles in a Haystack: On the Role of Incidental Bilingualism in PaLM's Translation Capability

Eleftheria Briakou, Colin Cherry, George Foster



Findings We find that 0.34% of PaLM's training instances contain at least one translation pair.

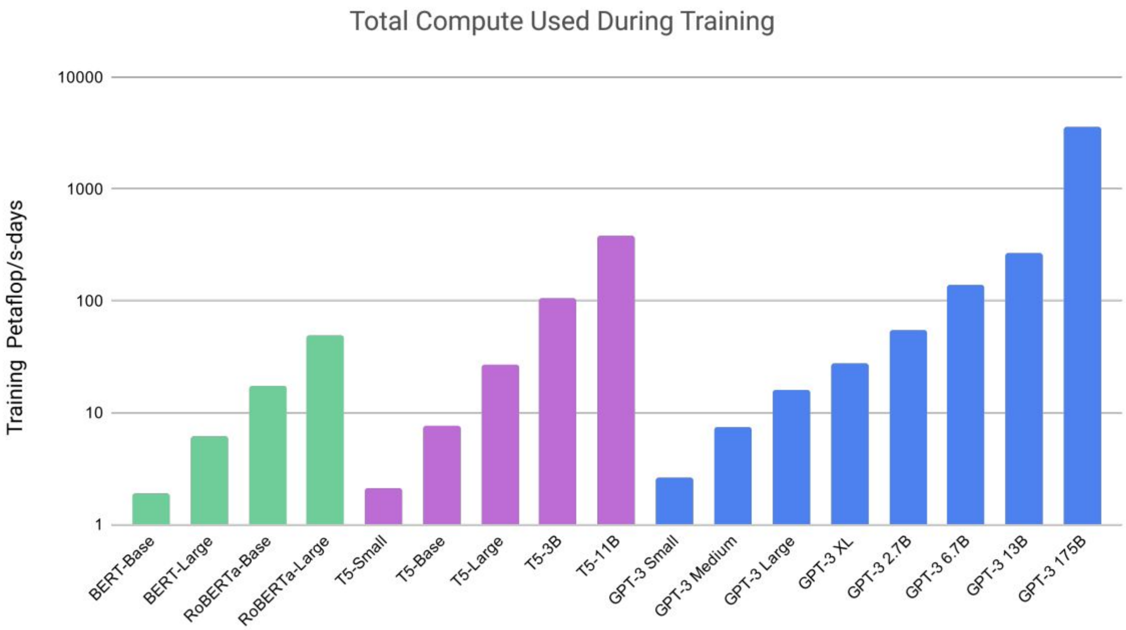
Results

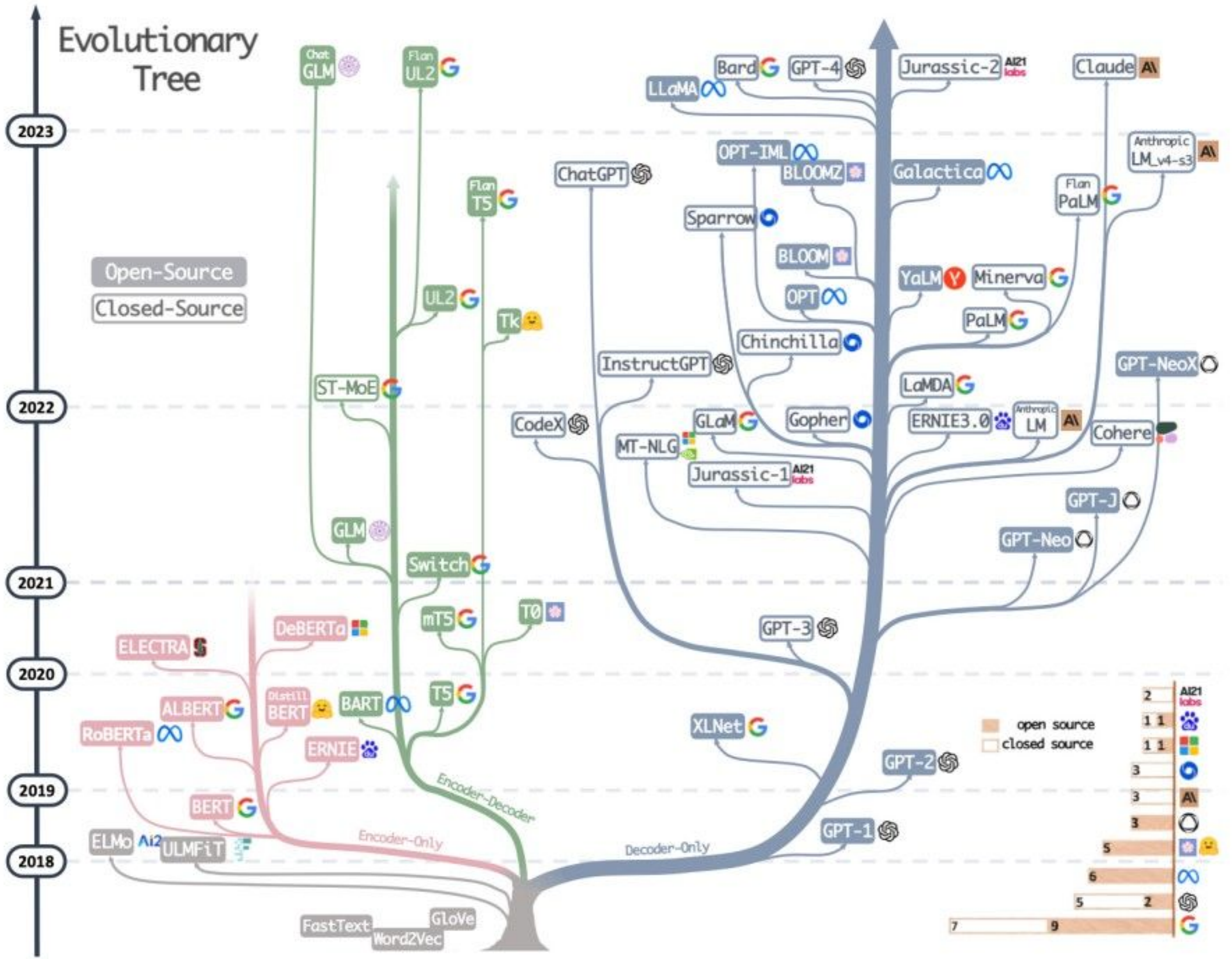
Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

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**

Computational Cost





The ERA of
Pretrained Models

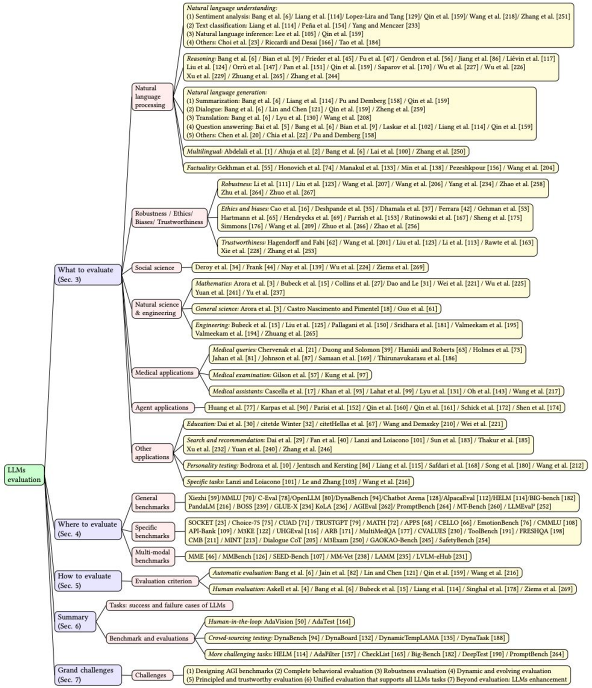
Evaluation of LLMs & The surge of “Benchmarks”

EVALUATIONS of LARGE LANGUAGE MODELS

A collection of papers and resources related to evaluations on large language models.

Yupeng Chang[*1] Xu Wang[*1] Jindong Wang[#2] Yuan Wu[#1] Kaijie Zhu[3] Hao Chen[4] Linyi Yang[5] Xiaoyuan Yi[2]
Cunxiang Wang[5] Yidong Wang[6] Wei Ye[6] Yue Zhang[5] Yi Chang[1] Philip S. Yu[7] Qiang Yang[8] Xing Xie[2]

[1] Jilin University, [2] Microsoft Research, [3] Institute of Automation, CAS [4] Carnegie Mellon University, [5] Westlake University, [6] Peking University, [7] University of Illinois, [8] Hong Kong University of Science and Technology
(*: Co-first authors, #: Co-corresponding authors)



Transformers in Vision

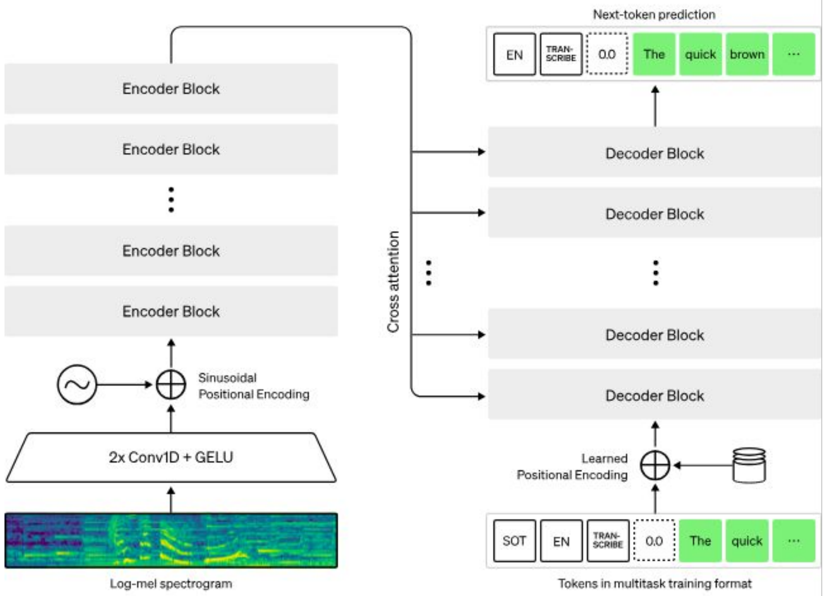
An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

ICLR 2021 · Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby · Edit social preview

Input image $R^{H \times W \times C}$, where H,W,C are height, width, channel

split into square-shaped patches of type $R^{P \times P \times C}$

Transformers in Audio



Conformer: Convolution-augmented Transformer for Speech Recognition

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, Ruoming Pang

---

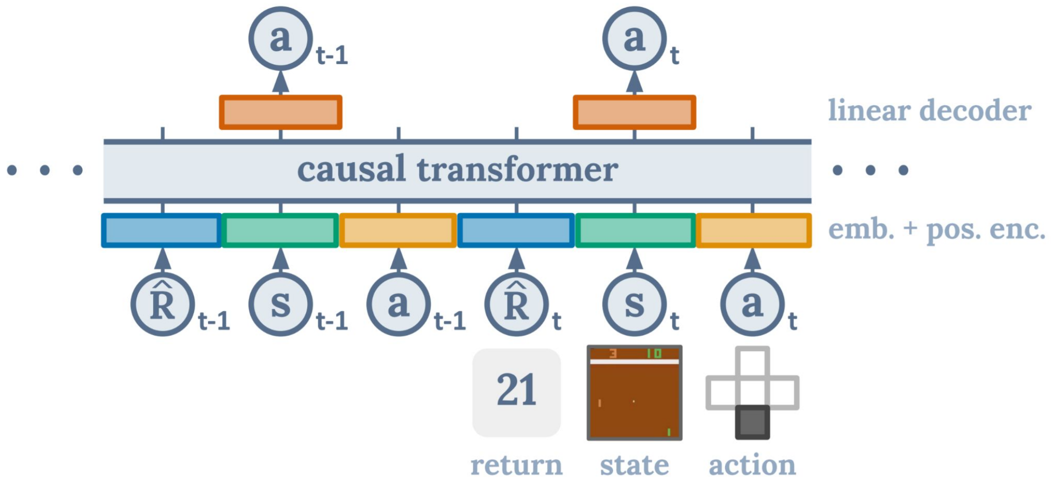
Robust Speech Recognition via Large-Scale Weak Supervision

Alec Radford [*1] **Jong Wook Kim** [*1] **Tao Xu** [1] **Greg Brockman** [1] **Christine McLeavey** [1] **Ilya Sutskever** [1]

Reinforcement Learning

Decision Transformer: Reinforcement Learning via Sequence Modeling

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, Igor Mordatch

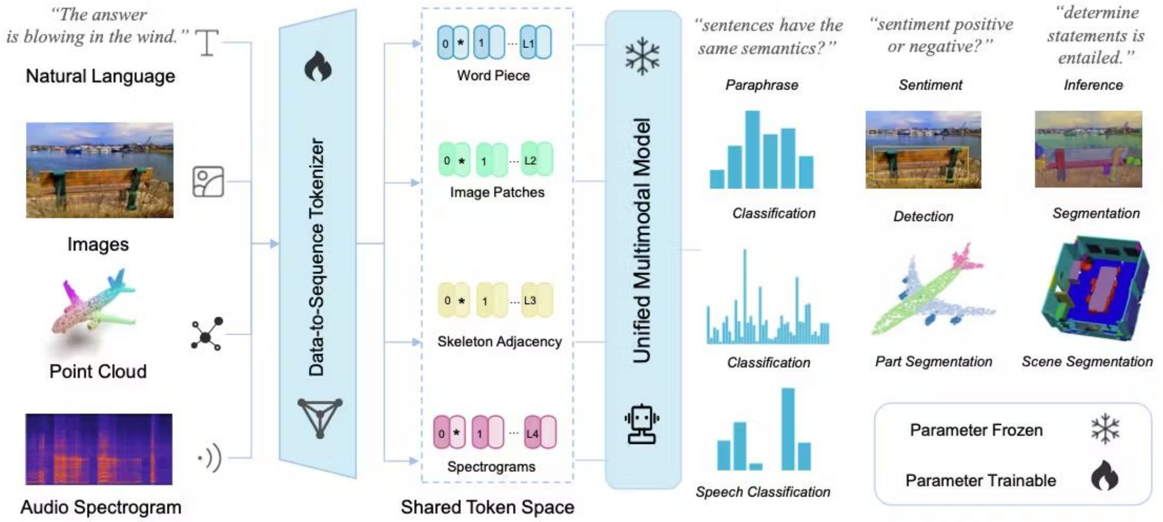


Cast any RL problem as a sequence modeling task

Slide Inspiration: Lucas Beyer

Transformers - Everything everywhere all at once

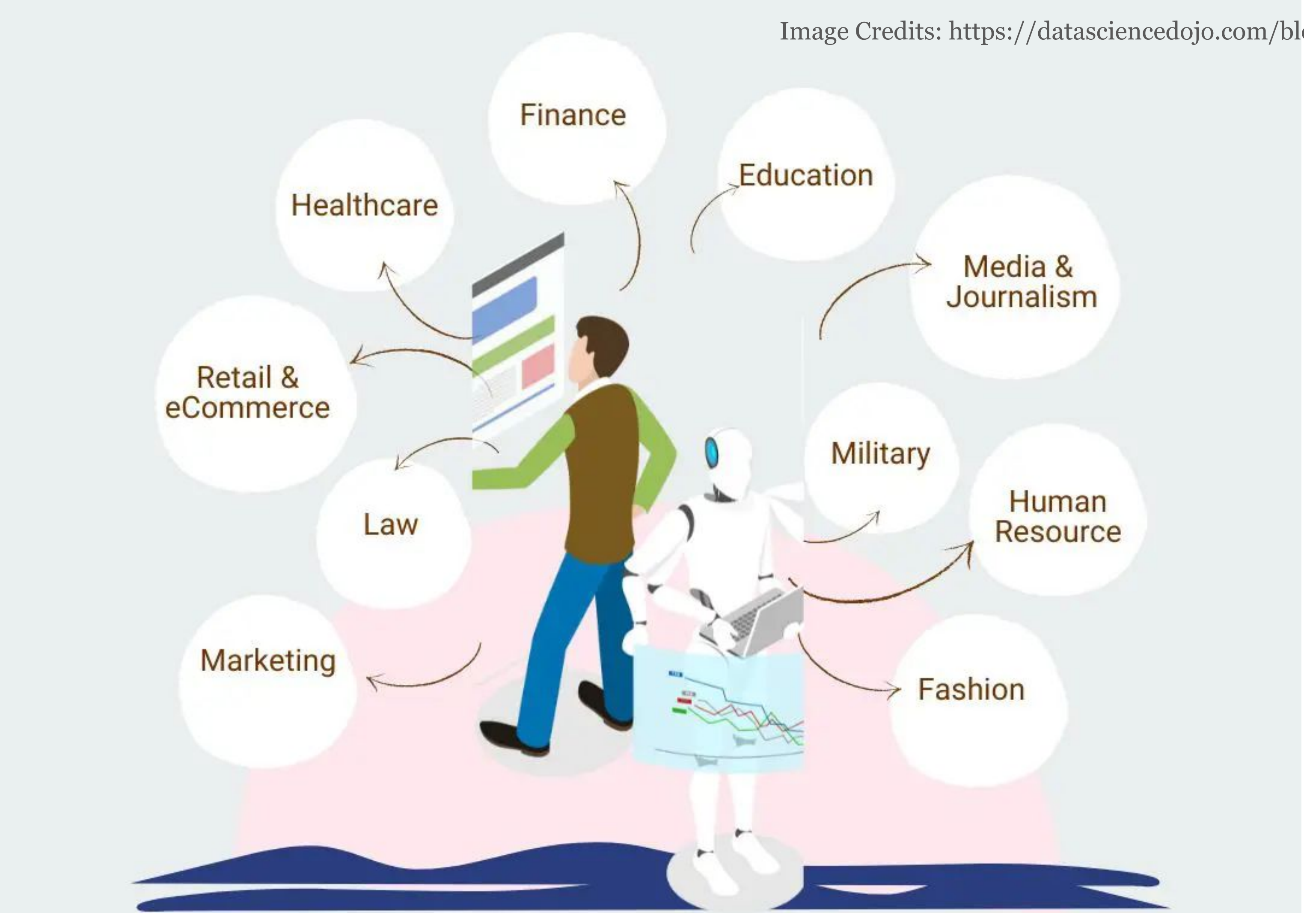
Anything once tokenized, can be passed through transformers



Slide Inspiration: Lucas Beyer

Transformers - Everything everywhere all at once

Image Credits: https://datasciencedojo.com/blog/llm-use-cases-top-10/



Thank you!