Natural Language Processing with Transformers

2024–25

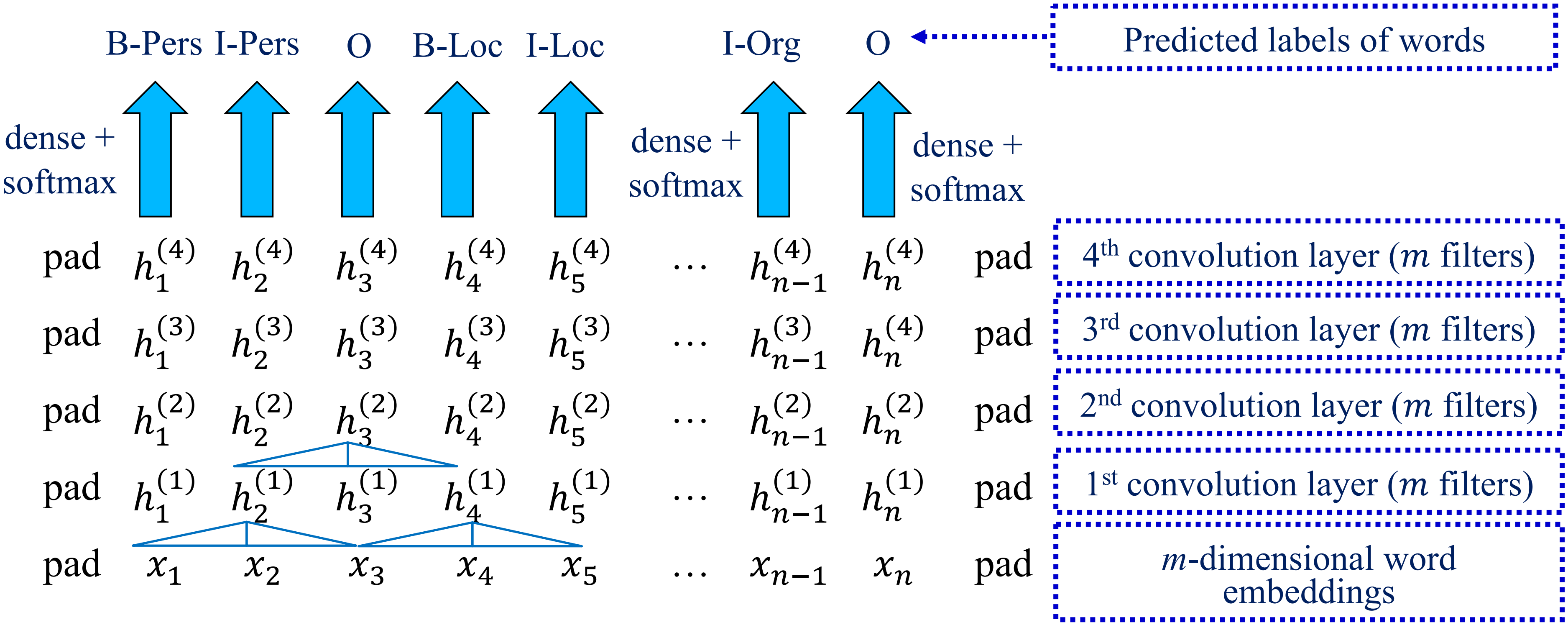
Ion Androutsopoulos

http://www.aueb.gr/users/ion/

Contents

- Transformer encoders and decoders.
- Pre-trained Transformers and Large Language Models (LLMs), BERT, SMITH, BART, T5, GPT-3, InstructGPT, ChatGPT, fine-tuning them, prompting them.
- Parameter efficient training, LoRA.
- Retrieval augmented generation (RAG), LLMs with tools.
- Data augmentation for NLP.
- Adding vision to LLMs

Reminder: CNNs for token classification



$$h_i^{(1)} = \mathrm{ReLU}\left(W^{(1)}[x_{i-1}; x_i; x_{i+1}] + b^{(1)}\right) + x_i \in \mathbb{R}^{m\times 1}$$

$$h_i^{(j)} = \mathrm{ReLU}\left(W^{(j)}\left[h_{i-1}^{(j-1)}; h_i^{(j-1)}; h_{i+1}^{(j-1)}\right] + b^{(j)}\right) + h_i^{(j-1)} \in \mathbb{R}^{m\times 1}$$

Transformers for token classification

$$
\begin{array}{cccccccc}
h_1^{(4)} & h_2^{(4)} & h_3^{(4)} & h_4^{(4)} & h_5^{(4)} & \cdots & h_{n-1}^{(4)} & h_n^{(4)} \\
h_1^{(3)} & h_2^{(3)} & h_3^{(3)} & h_4^{(3)} & h_5^{(3)} & \cdots & h_{n-1}^{(3)} & h_n^{(4)} \\
h_1^{(2)} & h_2^{(2)} & h_3^{(2)} & h_4^{(2)} & h_5^{(2)} & \cdots & h_{n-1}^{(2)} & h_n^{(2)} \\
h_1^{(1)} & h_2^{(1)} & h_3^{(1)} & h_4^{(1)} & h_5^{(1)} & \cdots & h_{n-1}^{(1)} & h_n^{(1)} \\
x_1 & x_2 & x_3 & x_4 & x_5 & \cdots & x_{n-1} & x_n
\end{array}
$$

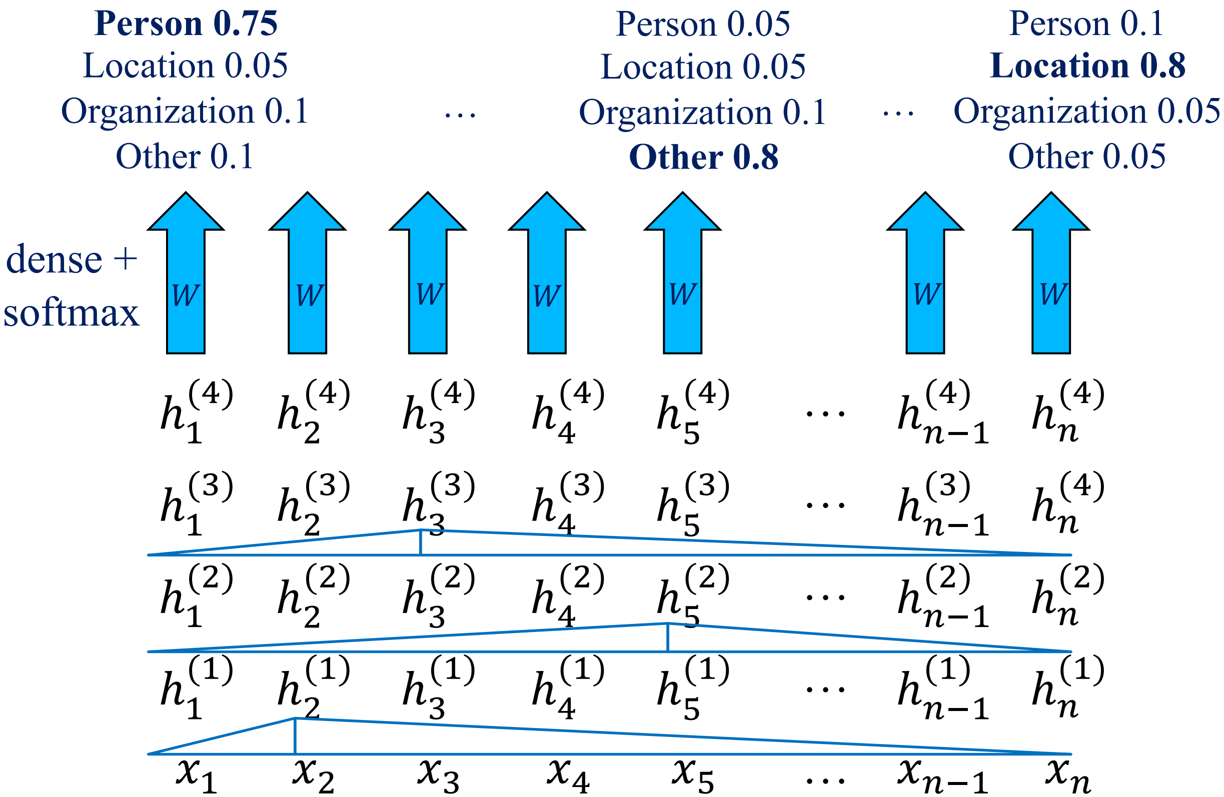
$a_{2,1}$ $a_{2,2}$ $a_{2,n}$

Initial *m*-dimensional word embeddings

$$h_i^{(1)} = \mathrm{MLP}^{(1)}\left(\sum_{r=1}^{n} a_{i,r}^{(1)} x_r\right) \in \mathbb{R}^m$$

To produce the **revised embedding for the *i*-th word** of a text, we **sum all the original embeddings** of the words of the text, but **weighted by attention scores**.

Transformers for token classification



Predicted labels of words

Compare to the correct predictions and **adjust the weights** of the **entire neural net**, including the bottom word (token) embeddings, which are randomly initialized.

Initial *m*-dimensional word embeddings

$$h_i^{(1)} = \mathrm{MLP}^{(1)}\left(\sum_{r=1}^{n} a_{i,r}^{(1)} x_r\right) \in \mathbb{R}^m$$

$$h_i^{(j)} = \mathrm{MLP}^{(j)}\left(\sum_{r=1}^{n} a_{i,r}^{(j)} h_r^{(j-1)}\right) \in \mathbb{R}^m$$

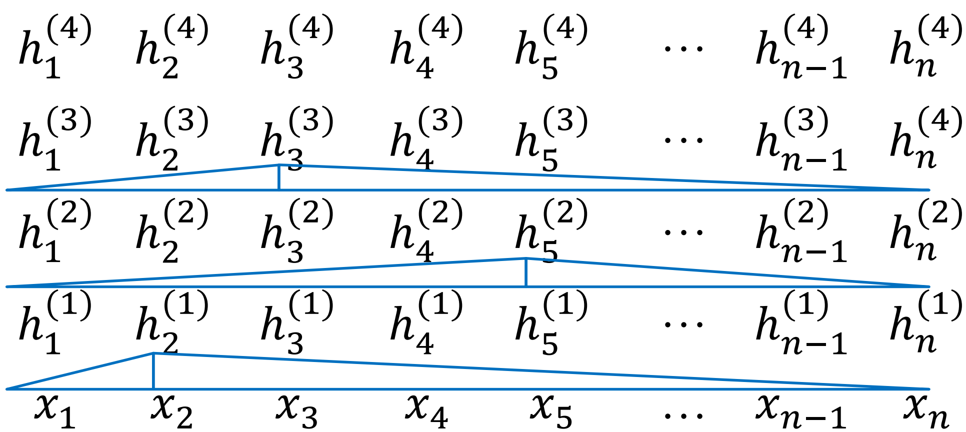
To produce the **revised embedding for the *i*-th word** of a text, we **sum all the original embeddings** of the words of the text, but **weighted by attention scores**.

Transformers for text classification

$$h^{max} = \left\langle \max\left(h_{*,1}^{(4)}\right), \max\left(h_{*,2}^{(4)}\right), \dots, \max(h_{*,m}^{(4)}) \right\rangle^{\mathrm{T}} \in \mathbb{R}^m$$

global max pooling
(max of each dimension)

Vector representing the **entire text**. We pass it through a **dense layer and softmax (or MLP)** to obtain a **probability per class**.



Compare to the correct predictions and **adjust the weights** of the **entire net**.

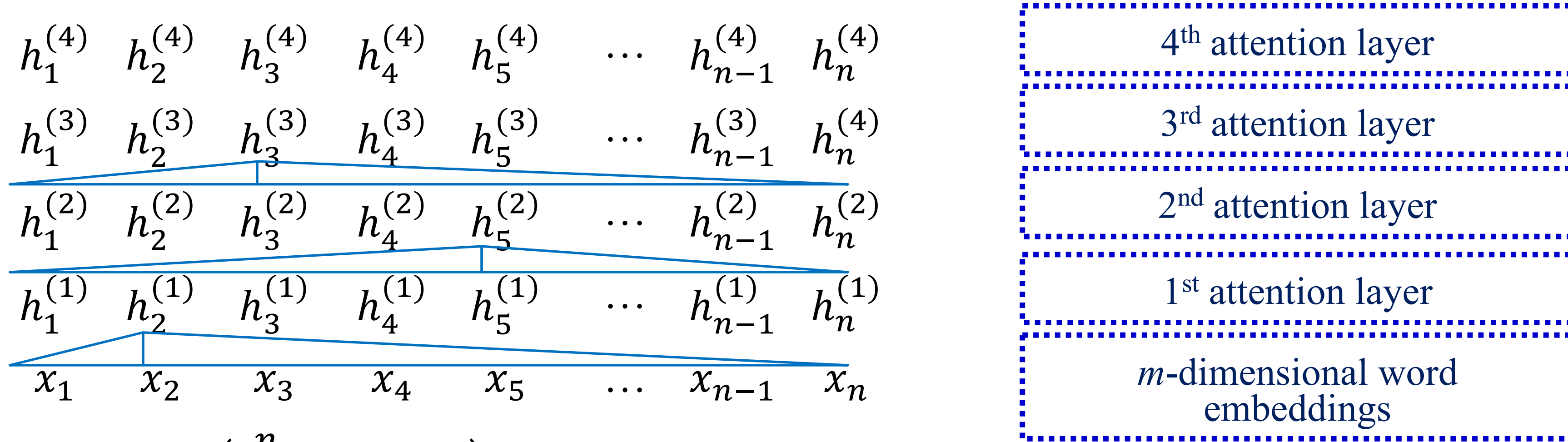
Initial m-dimensional word embeddings

$$h_i^{(1)} = \mathrm{MLP}^{(1)}\left(\sum_{r=1}^{n} a_{i,r}^{(1)} x_r\right) \in \mathbb{R}^m$$

$$h_i^{(j)} = \mathrm{MLP}^{(j)}\left(\sum_{r=1}^{n} a_{i,r}^{(j)} h_r^{(j-1)}\right) \in \mathbb{R}^m$$

Without the MLP (or at least a dense layer), each dimension of $h_i^{(j)}$ would only depend on the corresponding dimensions of the $h_r^{(j-1)}$ vectors.

Query-Key-Value self-attention



$$h_i^{(1)} = \mathrm{MLP}^{(1)}\left(\sum_{r=1}^{n} a_{i,r}^{(1)} v_r^{(1)}\right) =$$

$$= \mathrm{MLP}^{(1)}\left(\sum_{r=1}^{n} \mathrm{softmax}\left(q_i^{(1)T} k_r^{(1)}\right) v_r^{(1)}\right) \in \mathbb{R}^{m\times 1}$$

$$q_i^{(1)} = W^{Q,(1)} x_i$$

$$k_r^{(1)} = W^{K,(1)} x_r$$

$$v_r^{(1)} = W^{V,(1)} x_r$$

$$h_i^{(j)} = \mathrm{MLP}^{(j)}\left(\sum_{r=1}^{n} a_{i,r}^{(j)} v_r^{(j)}\right) =$$

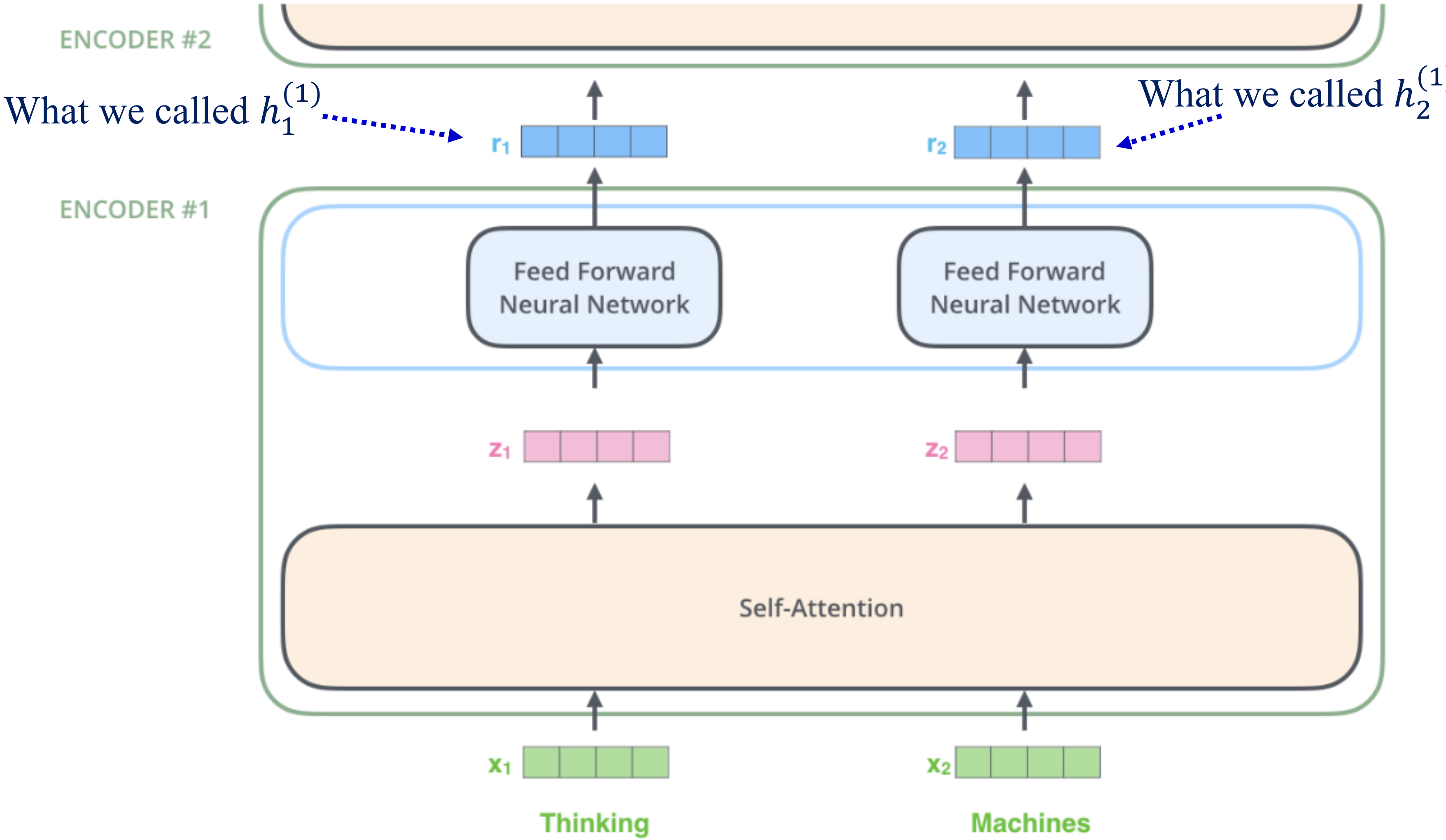
$$= \mathrm{MLP}^{(j)}\left(\sum_{r=1}^{n} \mathrm{softmax}\left(q_i^{(j)T} k_r^{(j)}\right) v_r^{(j)}\right) \in \mathbb{R}^{m\times 1}$$

$$q_i^{(j)} = W^{Q,(j)} h_i^{(j-1)}$$

$$k_r^{(j)} = W^{K,(j)} h_r^{(j-1)}$$

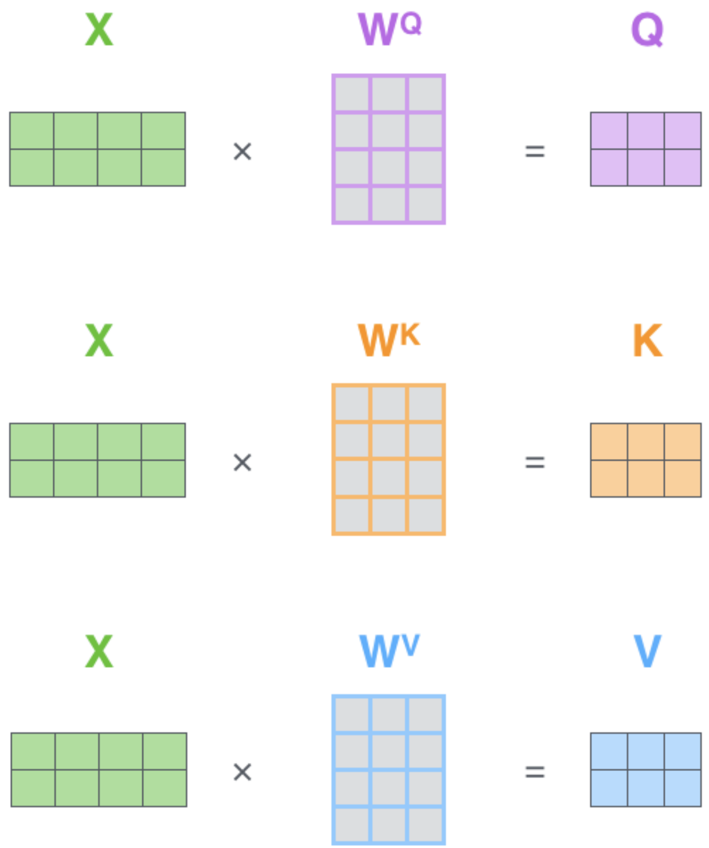
$$v_r^{(j)} = W^{V,(j)} h_r^{(j-1)}$$

Stacking Transformer Encoders

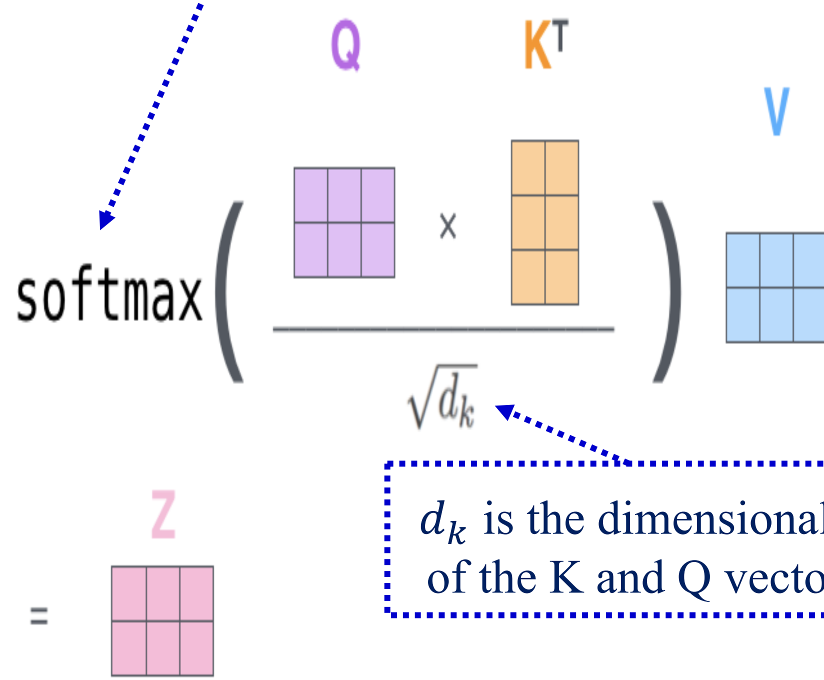


Figures from J. Alammar's "The Illustrated Transformer" (https://jalammar.github.io/illustrated-transformer/). Transformers paper: Vaswani et al., "Attention is All You Need", 2017 (https://arxiv.org/abs/1706.03762).

Query-Key-Value attention via matrices



Dropout also applied to the attention scores (after the softmax).

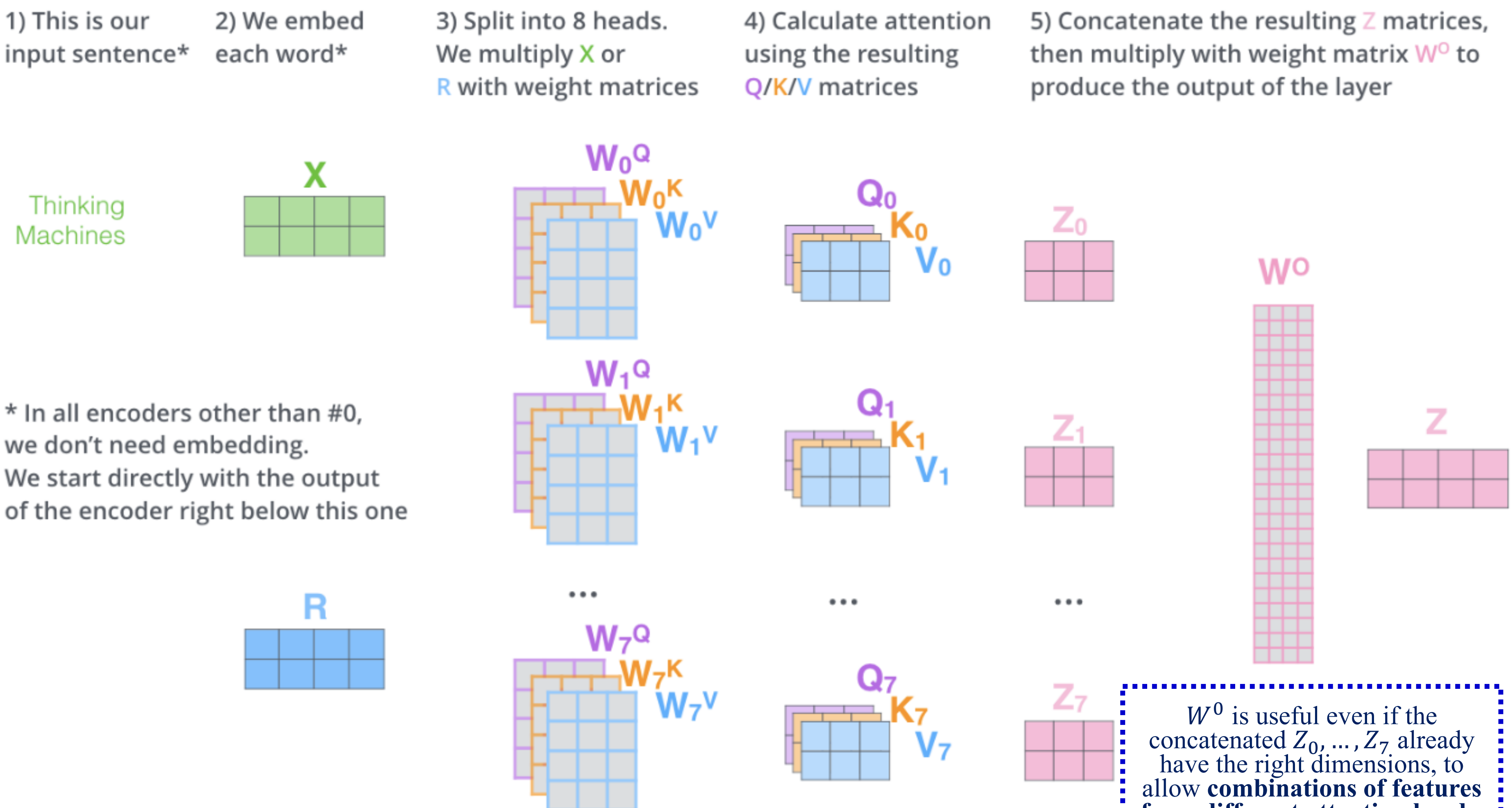


d_k is the dimensionality of the K and Q vectors.

Figures from J. Alammar's "The Illustrated Transformer" (https://jalammar.github.io/illustrated-transformer/). Transformers paper: Vaswani et al., "Attention is All You Need", 2017 (https://arxiv.org/abs/1706.03762).

Multiple attention heads

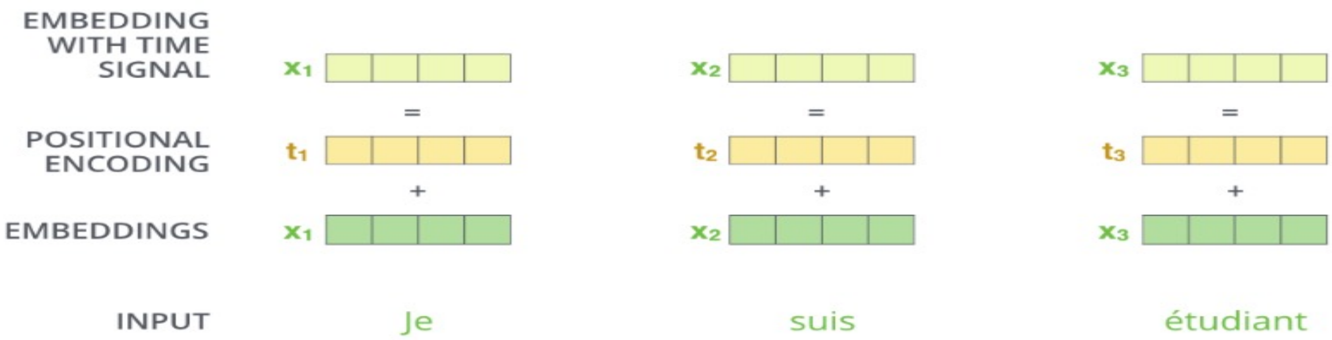
Because of the softmax, **each attention head mostly considers only one token**. So, let's use multiple attention heads.



W^0 is useful even if the concatenated $Z_0, \dots, Z_7$ already have the right dimensions, to allow **combinations of features from different attention heads**.

Figures from J. Alammar's "The Illustrated Transformer" (https://jalammar.github.io/illustrated-transformer/). Transformers paper: Vaswani et al., "Attention is All You Need", 2017 (https://arxiv.org/abs/1706.03762).

Positional encodings



Positional encodings needed to capture the **word order/positions**.

- **Without them**, Transformers are **unaware of word order**.
- **Sinusoid functions** used to produce them in the **original paper**.
- But can also be **position embeddings** learned during training.
  - Embedding of **position 1**, embedding of **position 2** etc.
- **Relative position embeddings** can also be used.
  - They consider the **distance** from the **current** to the **attended position** in the **self-attention blocks**. (https://paperswithcode.com/method/relative-position-encodings).

Figures from J. Alammar's "The Illustrated Transformer" (https://jalammar.github.io/illustrated-transformer/). Transformers paper: Vaswani et al., "Attention is All You Need", 2017 (https://arxiv.org/abs/1706.03762).

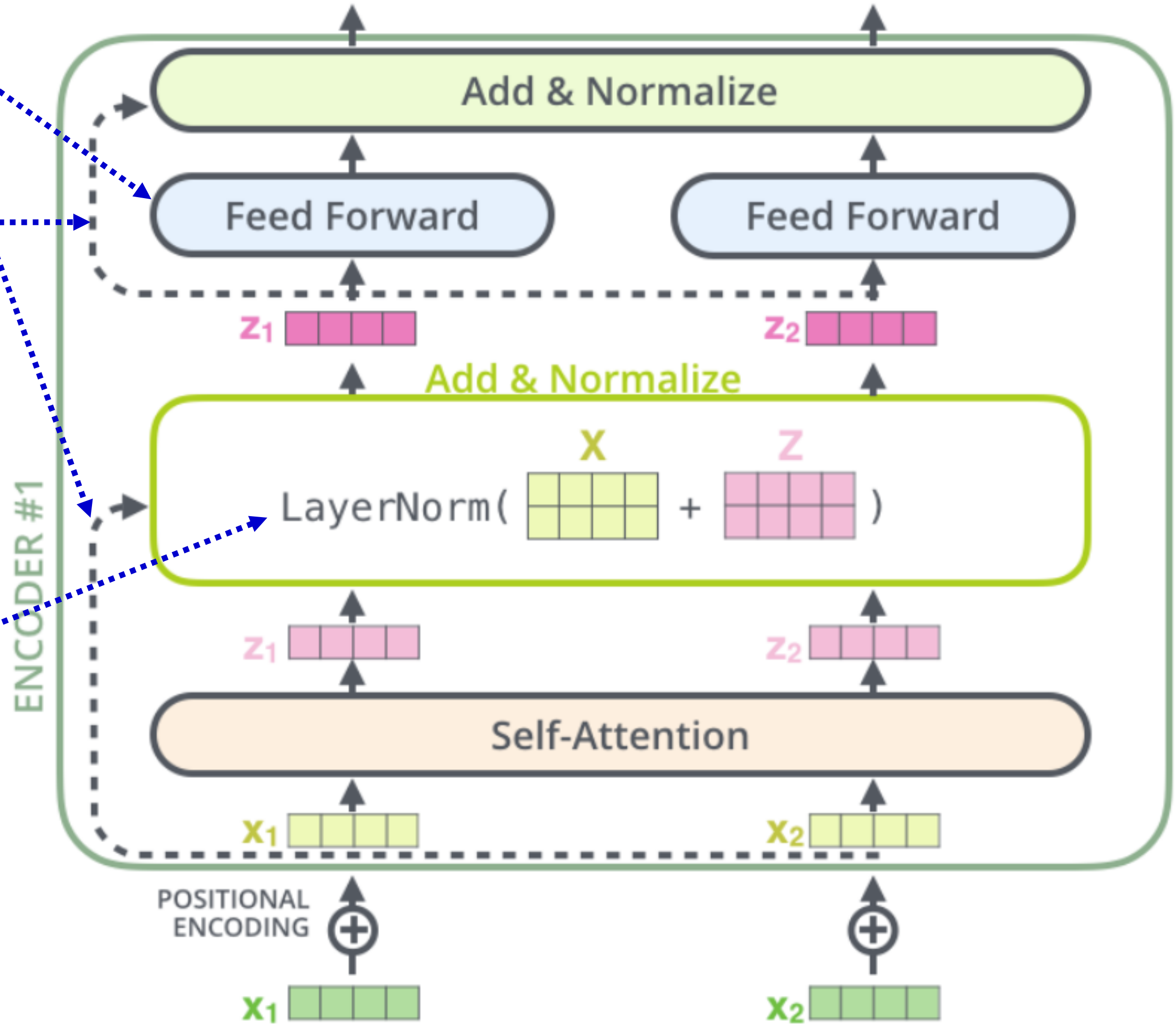
Complete Transformer encoder block

"Feed Forward": the **same MLP** at all word positions

"Add": **residual** connections

Layer Normalization (see Part 3). Here, we subtract from each cell $(X+Z)_{r,c}$ of (X+Z) the mean μ_r of its row, divide by the std. dev σ_r of the row, and multiply by a learned column-specific parameter g_c.

Dropout applied to the output of the self-attention and feed forward sublayers (before adding the residual and normalizing), inside the feed forward net, and after adding positional embeddings.

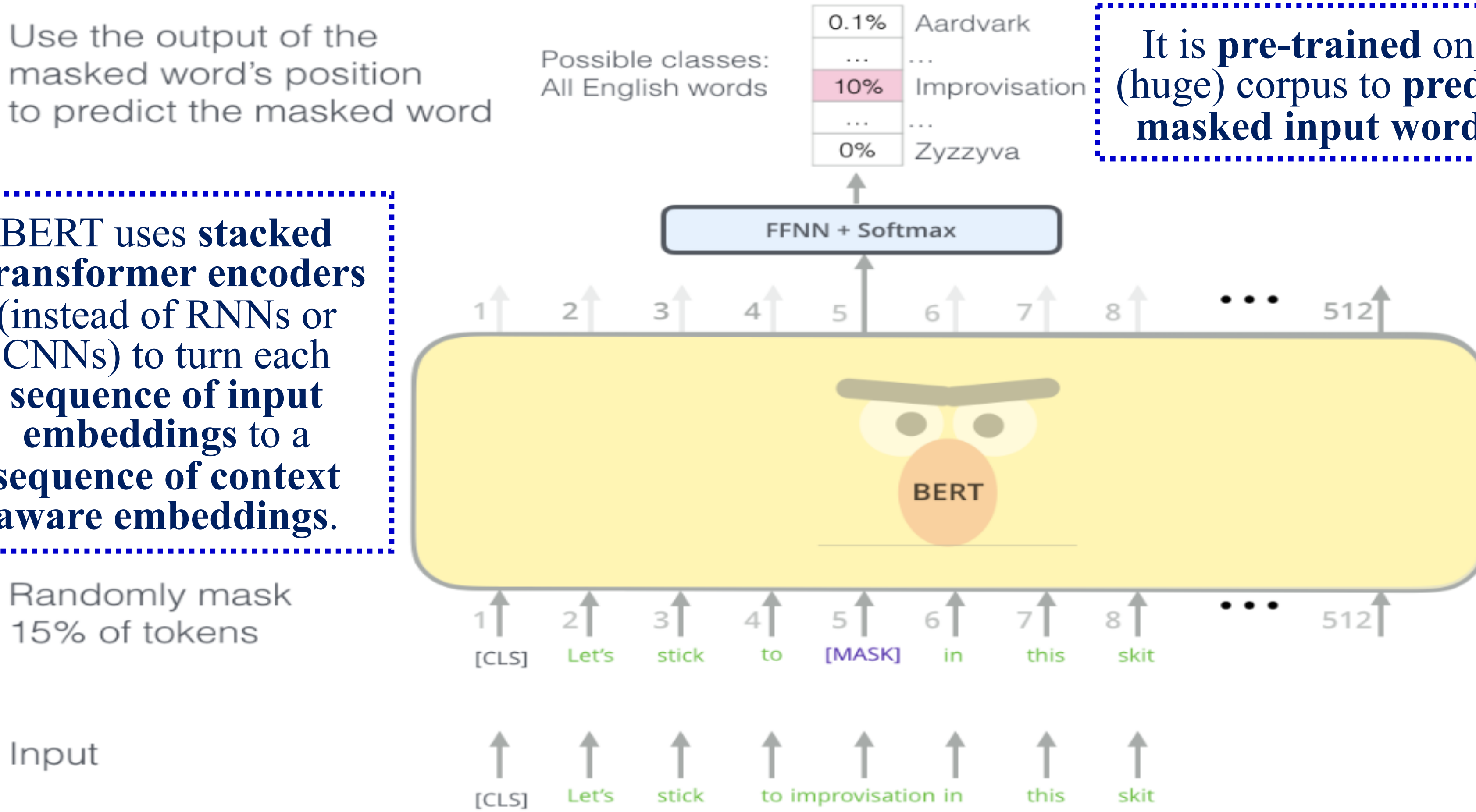


Figures from J. Alammar's "The Illustrated Transformer" (https://jalammar.github.io/illustrated-transformer/). Transformers paper: Vaswani et al., "Attention is All You Need", 2017 (https://arxiv.org/abs/1706.03762).

BERT – Pretraining to predict masked words

It is **pre-trained** on a (huge) corpus to **predict masked input words**.

BERT uses **stacked Transformer encoders** (instead of RNNs or CNNs) to turn each **sequence of input embeddings** to a **sequence of context aware embeddings**.



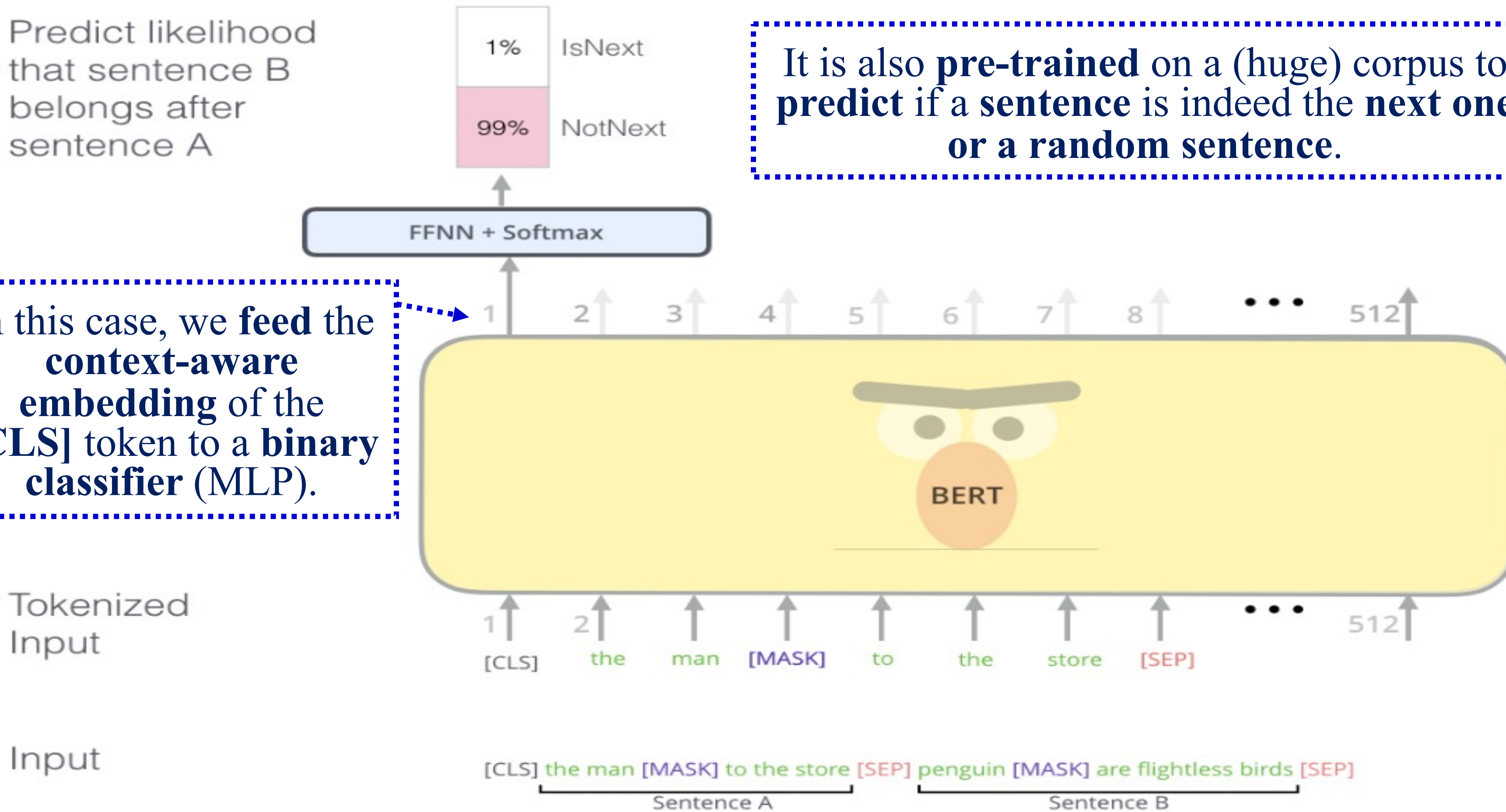
BERT's clever language modeling task masks 15% of words in the input and asks the model to predict the missing word.

Figures from J. Alammar's "The Illustrated BERT, ELMo, and co." (http://jalammar.github.io/illustrated-bert/). BERT paper: Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018 (https://arxiv.org/abs/1810.04805).

BERT – Pretraining to predict the next sentence

It is also **pre-trained** on a (huge) corpus to **predict** if a **sentence** is indeed the **next one or a random sentence**.

In this case, we **feed** the **context-aware embedding** of the **[CLS]** token to a **binary classifier** (MLP).



The second task BERT is pre-trained on is a two-sentence classification task. The tokenization is oversimplified in this graphic as BERT actually uses WordPieces as tokens rather than words --- so some words are broken down into smaller chunks.

Figures from J. Alammar's "The Illustrated BERT, ELMo, and co." (http://jalammar.github.io/illustrated-bert/). BERT paper: Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018 (https://arxiv.org/abs/1810.04805).

BERT – Fine-tuning for sentence classification

We **feed** the **context-aware embedding** of the **[CLS]** token of each **sentence** to a **task-specific classifier** (e.g., MLP) that classifies the sentence (e.g., **Positive**, **Neutral**, **Negative** etc.)

Starting from the **pre-trained BERT**, we **jointly train BERT (further)** and the **task-specific classifier** on (possibly few) **task-specific training examples** (e.g., tweets + opinion labels).

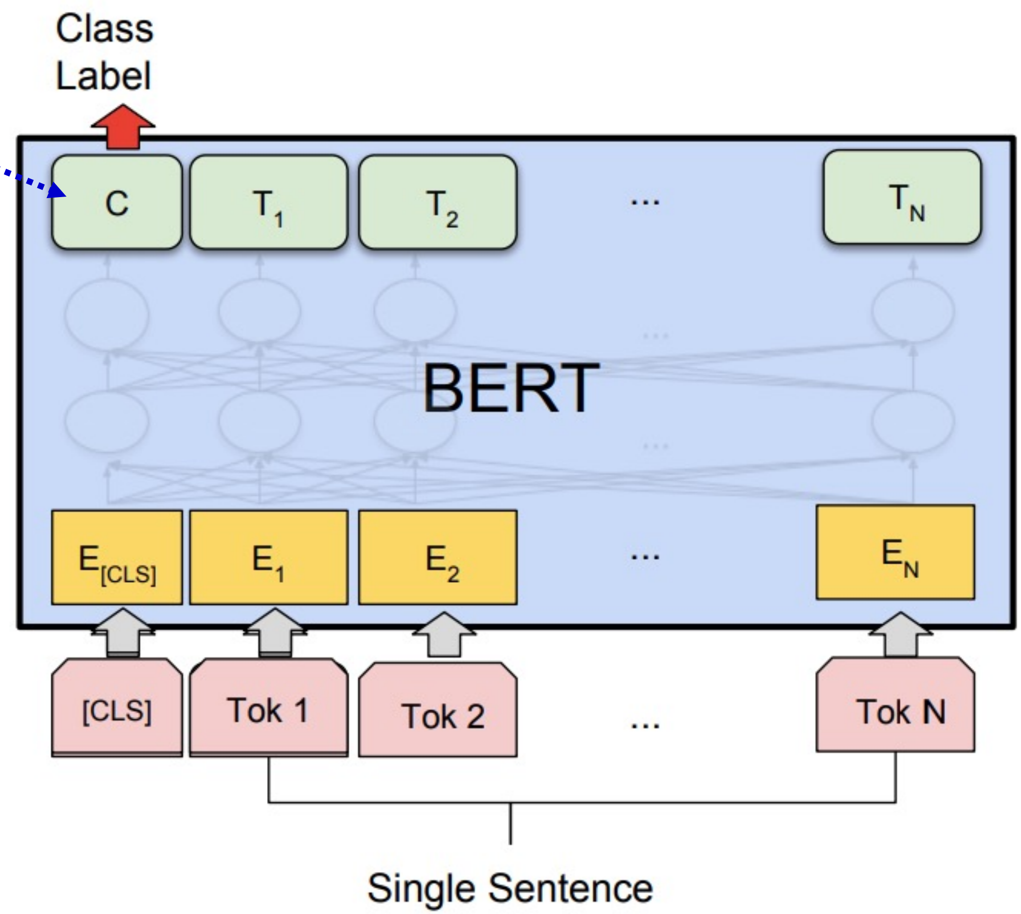


Figure from Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018 (https://arxiv.org/abs/1810.04805).

BERT – Fine-tuning for token classification

We **feed** the **context-aware embeddings** of the sentence's words to a **classifier** (e.g., MLP) that classifies them as **B-Per**, **I-Per**, **B-Org**, **I-Org**, …, **Other**.

Starting from the **pre-trained BERT**, we **jointly train BERT (further)** and the **task-specific classifier** on (possibly few) **task-specific training examples** (manually labeled sentences).

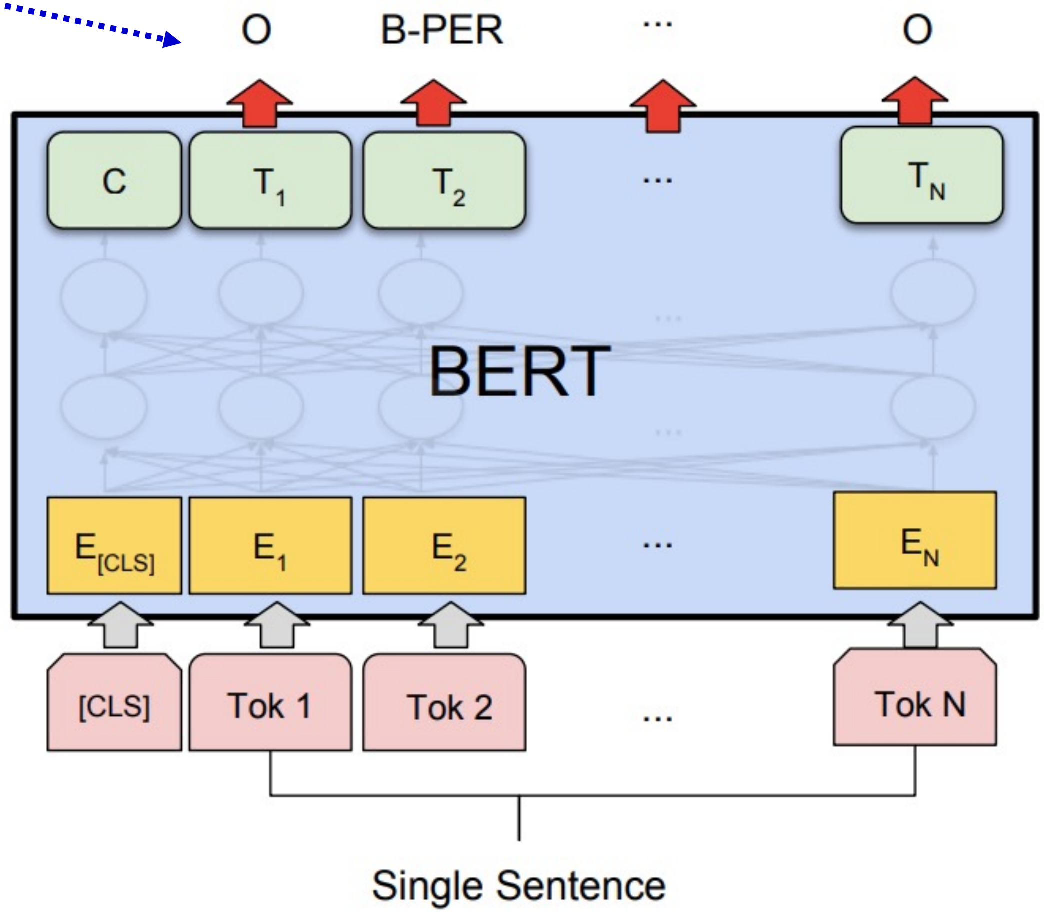


Figure from Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018 (https://arxiv.org/abs/1810.04805).

BERT – Fine-tuning for textual entailment

We **feed** the **context-aware embedding** of the **[CLS]** token of each **sentence pair** to a **task-specific classifier** (e.g., MLP) that classifies the pair as **Entailment**, **Contradiction**, **Neutral**. E.g., "Mary plays in the garden" entails "Mary is in the garden" but contradicts "Mary is asleep".

Starting from the **pre-trained BERT**, we **jointly train BERT (further)** and the **task-specific classifier** on (possibly few) **task-specific training examples** (annotated sentence pairs).

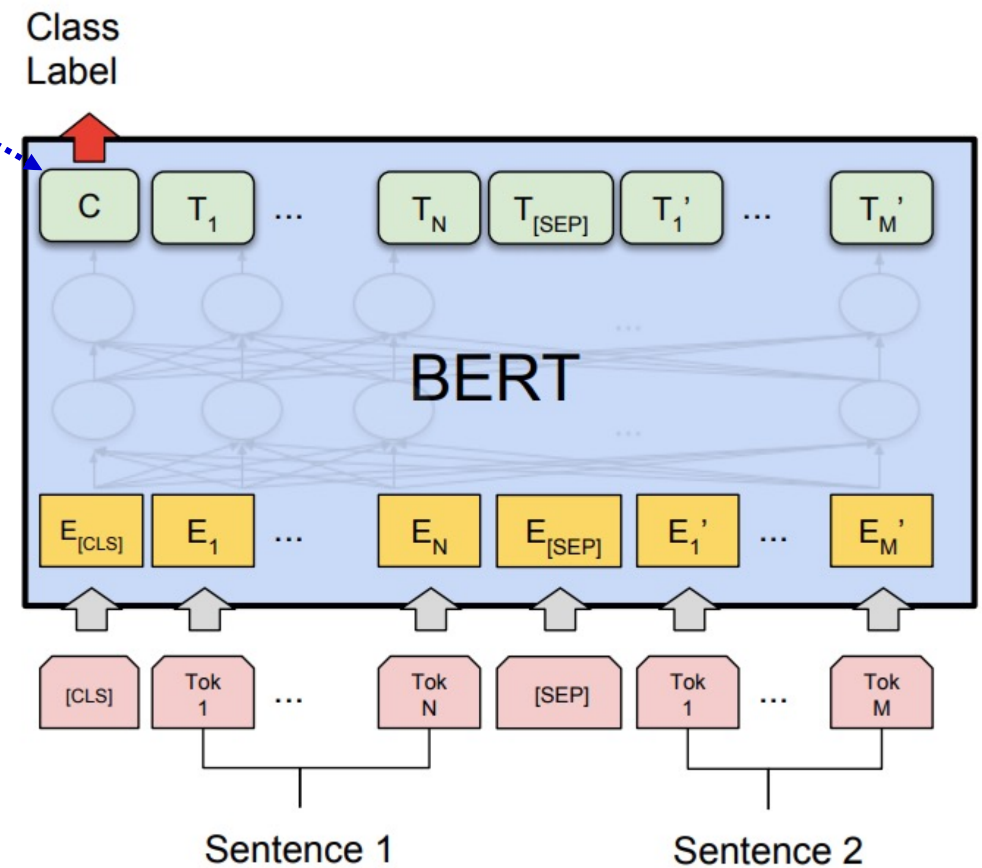


Figure from Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018 (https://arxiv.org/abs/1810.04805).

Machine Reading Comprehension (MRC)

Paragraph

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

Question

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Figure from P. Rajpurkar et al., "SQuAD: 100,000+ Questions for Machine Comprehension of Text.", EMNLP 2016 (https://aclweb.org/anthology/D16-1264).

BERT – Fine-tuning for MRC

We **feed** the **context-aware embeddings** of the paragraph's words to a **classifier** (e.g., MLP) that classifies them as **Start-Answer**, **End-Answer**, **Other**.

Starting from the **pre-trained BERT**, we **jointly train BERT (further)** and the **task-specific classifier** on (possibly few) **task-specific training examples** (paragraph-question pairs).

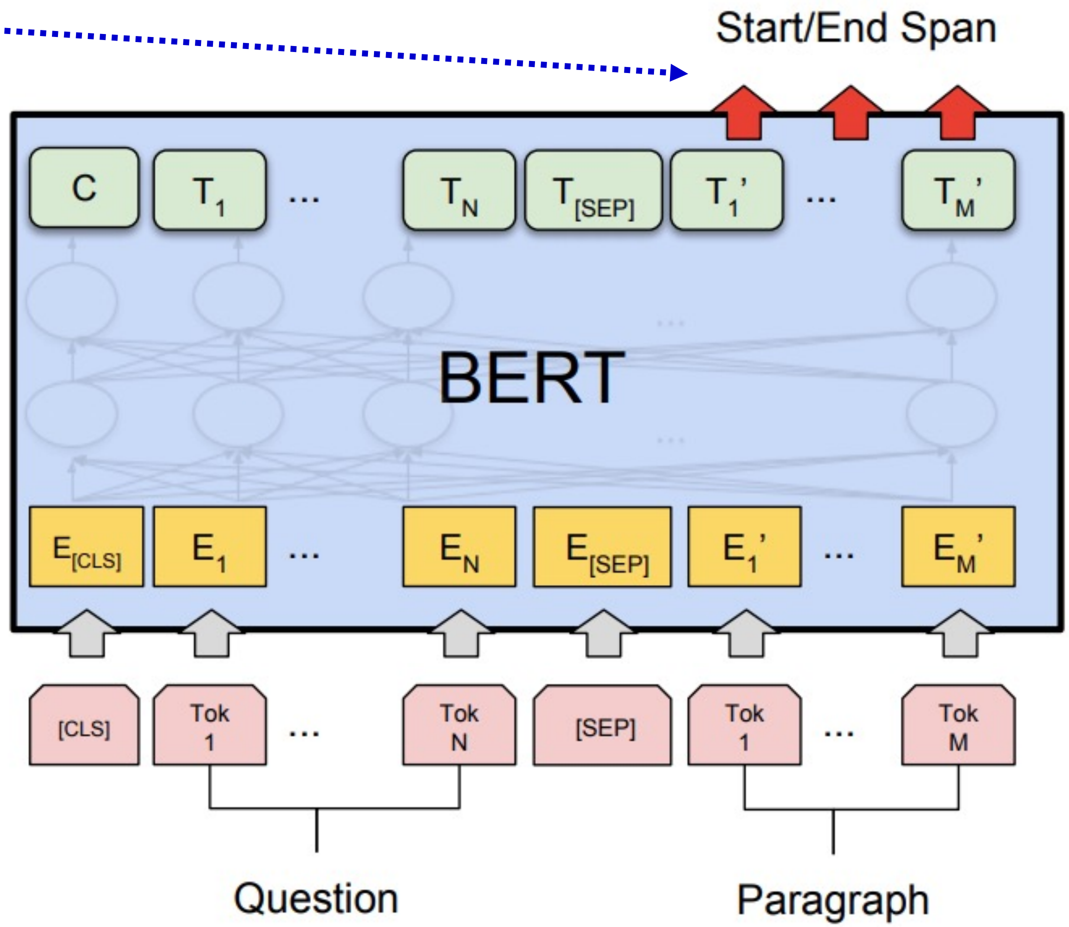


Figure from Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018 (https://arxiv.org/abs/1810.04805).

SMITH (hierarchical BERT)

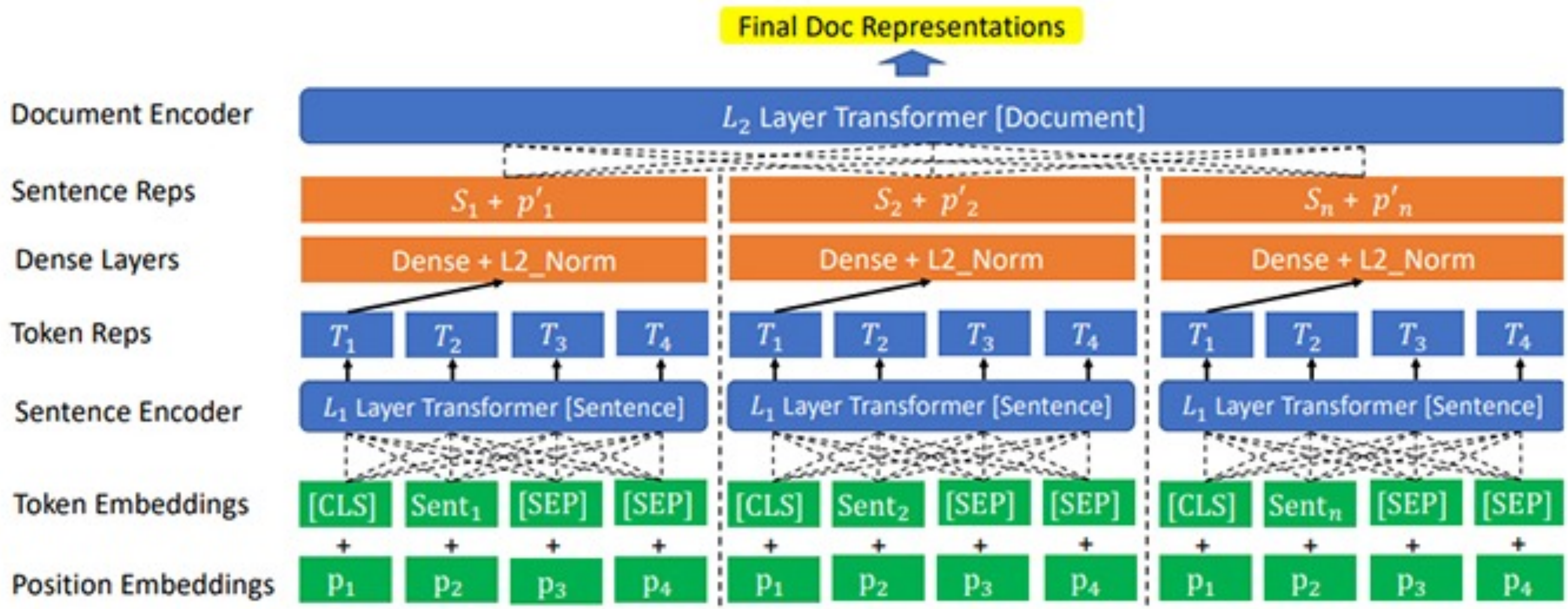
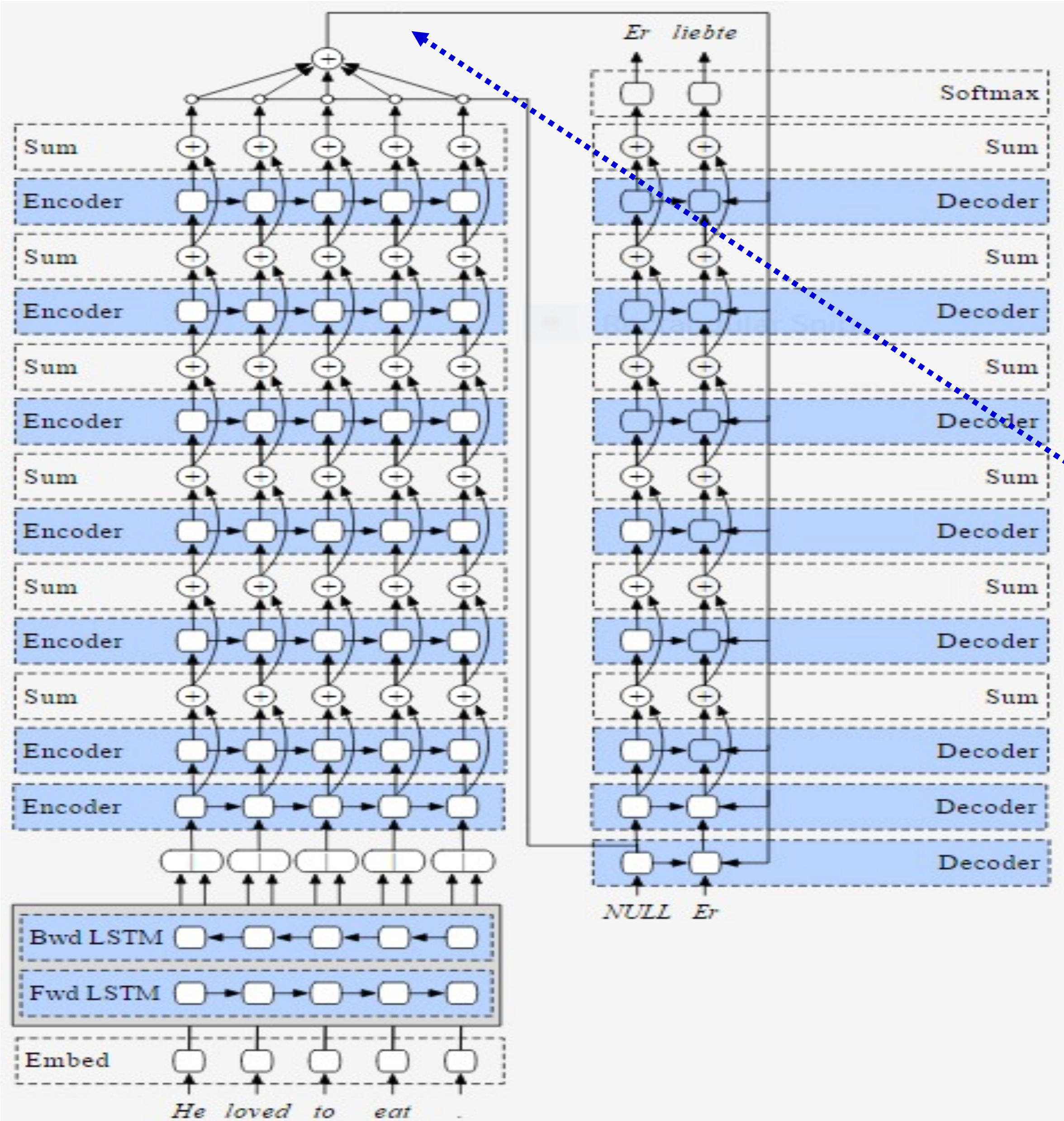


Figure from Yang et al., “Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching”, CIKM 2020 (https://dl.acm.org/doi/10.1145/3340531.3411908).

BERT variants for long documents include, for example, also **Longformer** (https://arxiv.org/abs/2004.05150) and **Big Bird** (https://arxiv.org/abs/2007.14062), which are not hierarchical, but use **sparse attention** to avoid quadratic complexity (to the input length). See also, e.g., **FlashAttention** (https://arxiv.org/abs/2205.14135).

Reminder: RNN-based MT system



Google's paper:
https://arxiv.org/abs/1609.08144

Images from Stephen Merity's
http://smerity.com/articles/2016/google_nmt_arch.html

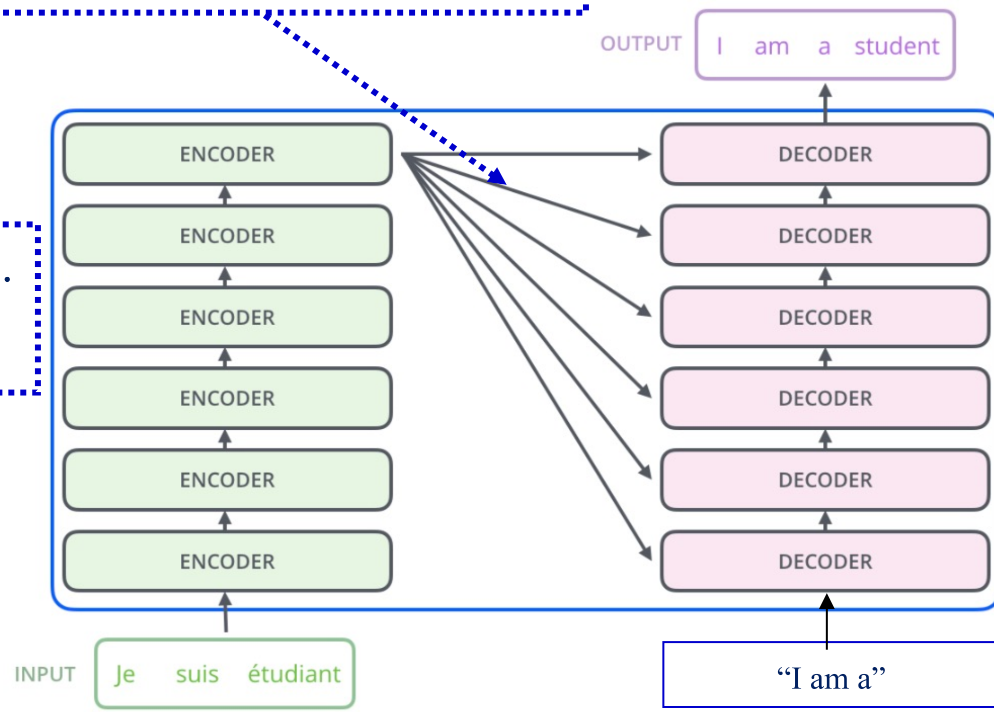
Attention over the states of the encoder.

Stacked Transformer encoders-decoders

Attention over the vectors produced by the encoder. Keys (K) and Values (V) come from the vectors produced by the encoders. Queries (Q) come from the vectors of the decoder.

Using an **encoder/decoder** allows us to generate a **translation** with a **different number and order of tokens than the input** (source) text.

Stacked encoders. In **BERT** we use **only encoders**.



Stacked decoders. Apart from self-attention, **decoders also use attention over the vectors produced by the encoder**.

Translation generated **so far**.

Figure from J. Alammar’s “The Illustrated Transformer” (https://jalammar.github.io/illustrated-transformer/). Transformers paper: Vaswani et al., “Attention is All You Need”, 2017 (https://arxiv.org/abs/1706.03762).

QKV self-attention and cross-attention

stacked decoder layers

$d_1^{(3)} \quad d_2^{(3)} \quad d_3^{(3)} \quad d_4^{(3)} \quad d_5^{(3)}$

stacked encoder layers

$e_1^{(2)} \quad e_2^{(2)} \quad e_3^{(2)} \quad e_4^{(2)}$

$d_1^{(2)} \quad d_2^{(2)} \quad d_3^{(2)} \quad d_4^{(2)} \quad d_5^{(2)}$

self-attention

embeddings of input tokens

$e_1^{(1)} \quad e_2^{(1)} \quad e_3^{(1)} \quad e_4^{(1)}$

$d_1^{(1)} \quad d_2^{(1)} \quad d_3^{(1)} \quad d_4^{(1)} \quad d_5^{(1)}$

$$e_i^{(2)} = \mathrm{MLP}^{(e,2)}\left(\sum_{r=1}^{4} a_{i,r}^{(e,2)} v_r^{(e,2)}\right) =$$

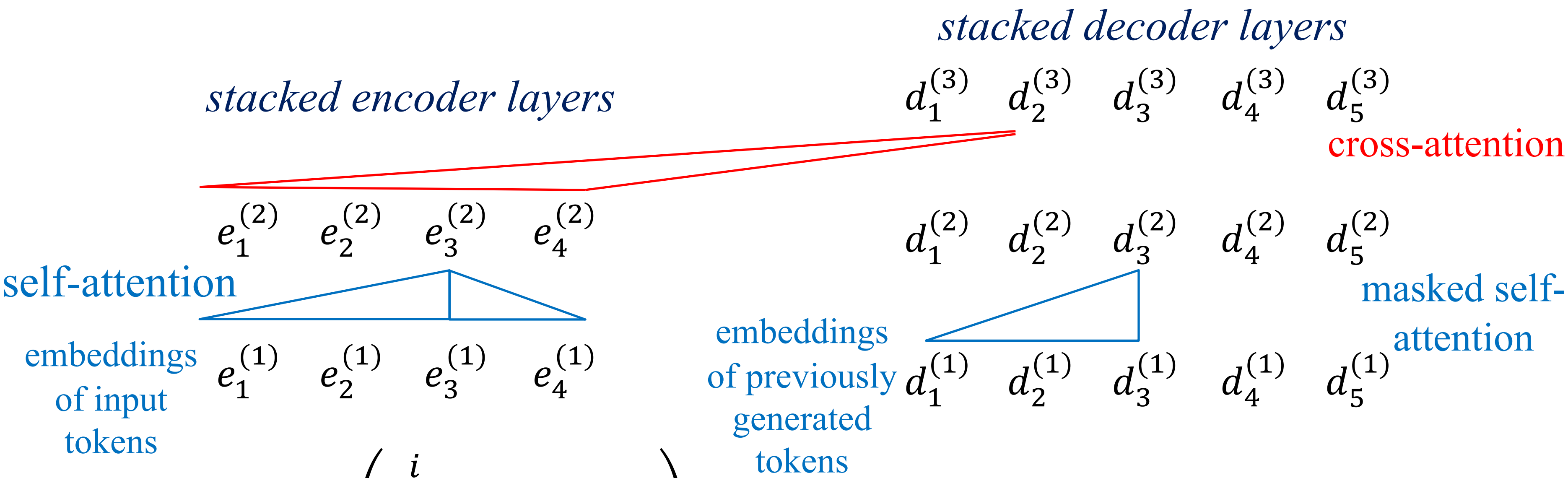
$$= \mathrm{MLP}^{(e,2)}\left(\sum_{r=1}^{4} \mathrm{softmax}\left(q_i^{(e,2)T} k_r^{(e,2)}\right) v_r^{(e,2)}\right) \in \mathbb{R}^{m\times 1}$$

$$q_i^{(e,2)} = W^{Q,(e,2)} e_i^{(1)}$$

$$k_r^{(e,2)} = W^{K,(e,2)} e_r^{(1)}$$

$$v_r^{(e,2)} = W^{V,(e,2)} e_r^{(1)}$$

QKV self-attention and cross-attention



$$d_i^{(2)} = \mathrm{MLP}^{(d,2)}\left(\sum_{r=1}^{i} a_{i,r}^{(d,2)} v_r^{(d,2)}\right) =$$

$$= \mathrm{MLP}^{(d,2)}\left(\sum_{r=1}^{i} \mathrm{softmax}\left(q_i^{(d,2)T} k_r^{(d,2)}\right) v_r^{(d,2)}\right) \in \mathbb{R}^{m\times 1}$$

$$q_i^{(d,2)} = W^{Q,(d,2)} d_i^{(1)}$$

$$k_r^{(d,2)} = W^{K,(d,2)} d_r^{(1)}$$

$$v_r^{(d,2)} = W^{V,(d,2)} d_r^{(1)}$$

$$d_i^{(3)} = \mathrm{MLP}^{(d,3)}\left(\sum_{r=1}^{4} a_{i,r}^{(d,3)} v_r^{(d,3)}\right) =$$

$$= \mathrm{MLP}^{(d,3)}\left(\sum_{r=1}^{4} \mathrm{softmax}\left(q_i^{(d,3)T} k_r^{(d,3)}\right) v_r^{(d,3)}\right) \in \mathbb{R}^{m\times 1}$$

$$q_i^{(d,3)} = W^{Q,(d,3)} d_i^{(2)}$$

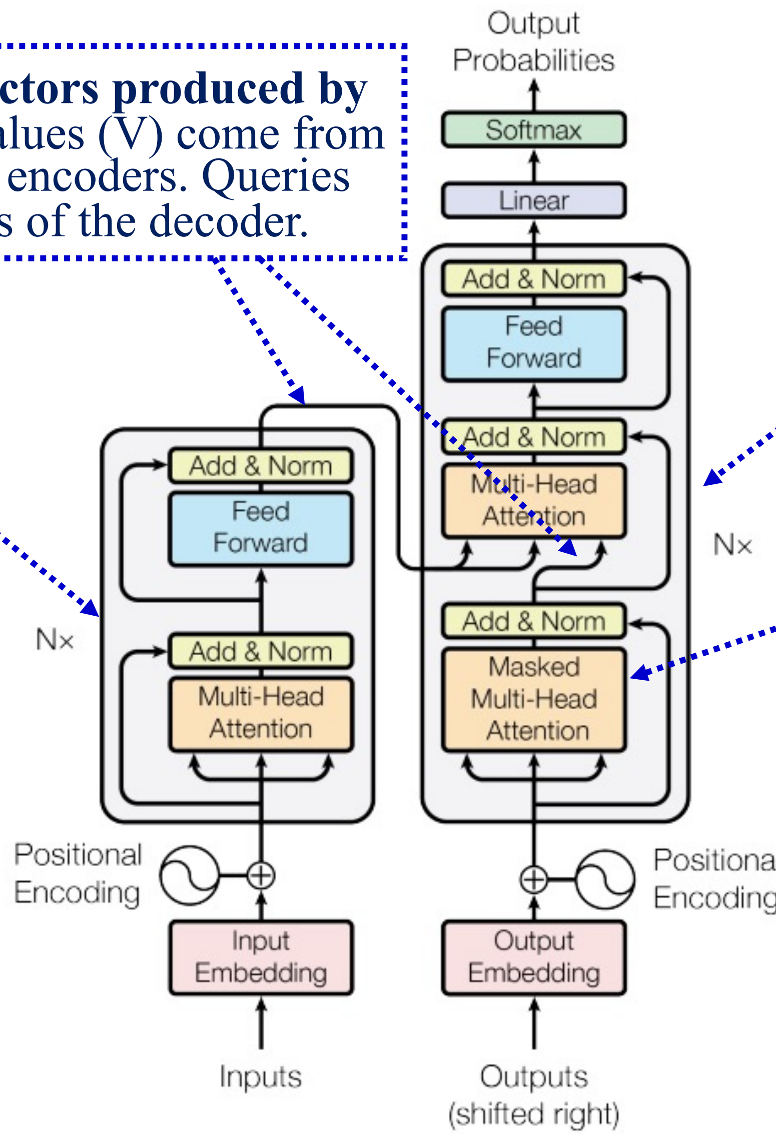
$$k_r^{(d,3)} = W^{K,(d,3)} e_r^{(2)}$$

$$v_r^{(d,3)} = W^{V,(d,3)} e_r^{(2)}$$

Transformer-based Encoder-Decoder

Cross-attention over the vectors produced by the encoder. Keys (K) and Values (V) come from the vectors produced by the encoders. Queries (Q) come from the vectors of the decoder.

N stacked encoders. In **BERT** we use **only encoders**.



N stacked decoders. Apart from self-attention, **decoders also use cross-attention over the vectors produced by the encoder**.

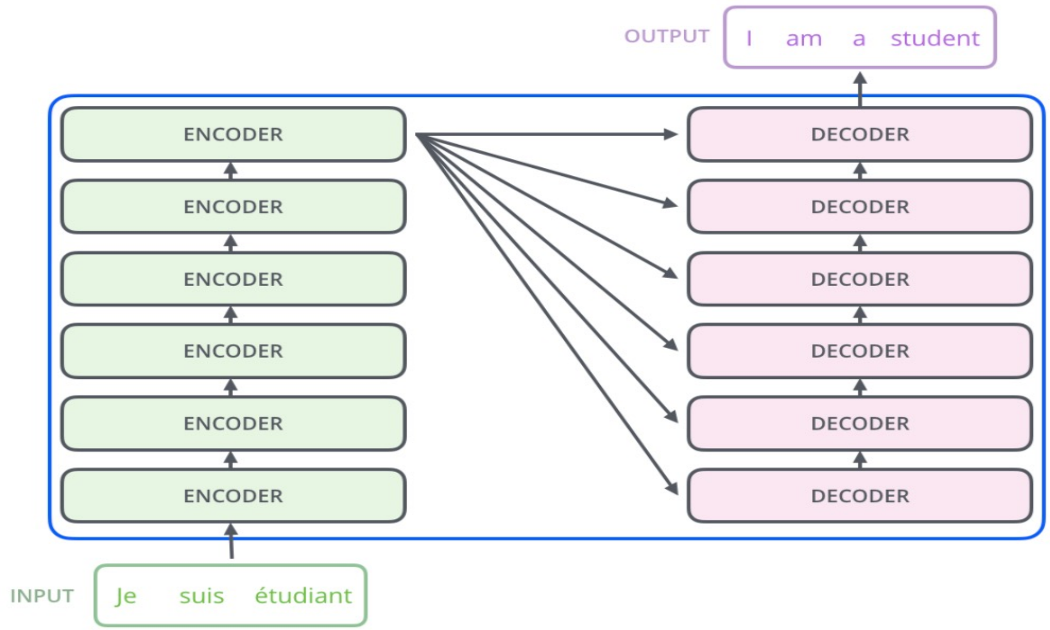
Masked self-attention: At each word position, the **decoder sees only the preceding** gold (at training) or generated (at test) **words of the translation.**

Figure from Vaswani et al., “Attention is All You Need”, 2017. https://arxiv.org/abs/1706.03762

BART – Using encoders & decoders

BART uses **both** stacked **encoder** and stacked **decoder** Transformer layers.

During pre-training, BART is trained to **“translate” noised text to the original** (without noise) text.



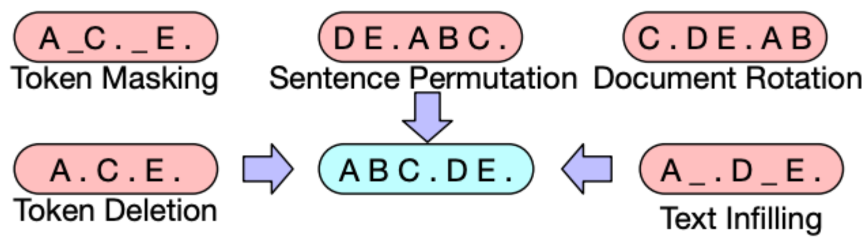


Figure 2: Transformations for noising the input that we experiment with. These transformations can be composed.

Top figure from J. Alammar’s “The Illustrated Transformer” (https://jalammar.github.io/illustrated-transformer/). Bottom figures from M. Lewis et al., “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”, ACL 2020 (https://www.aclweb.org/anthology/2020.acl-main.703/).

BART – Fine-tuning for summarization

Source Document (abbreviated)	BART Summary
The researchers examined three types of coral in reefs off the coast of Fiji ... The researchers found when fish were plentiful, they would eat algae and seaweed off the corals, which appeared to leave them more resistant to the bacterium Vibrio coralliilyticus, a bacterium associated with bleaching. The researchers suggested the algae, like warming temperatures, might render the corals' chemical defenses less effective, and the fish were protecting the coral by removing the algae.	Fisheries off the coast of Fiji are protecting coral reefs from the effects of global warming, according to a study in the journal Science.
This is the first time anyone has been recorded to run a full marathon of 42.195 kilometers (approximately 26 miles) under this pursued landmark time. It was not, however, an officially sanctioned world record, as it was not an "open race" of the IAAF. His time was 1 hour 59 minutes 40.2 seconds. Kipchoge ran in Vienna, Austria. It was an event specifically designed to help Kipchoge break the two hour barrier.	Kenyan runner Eliud Kipchoge has run a marathon in less than two hours.
PG&E stated it scheduled the blackouts in response to forecasts for high winds amid dry conditions. The aim is to reduce the risk of wildfires. Nearly 800 thousand customers were scheduled to be affected by the shutoffs which were expected to last through at least midday tomorrow.	Power has been turned off to millions of customers in California as part of a power shutoff plan.

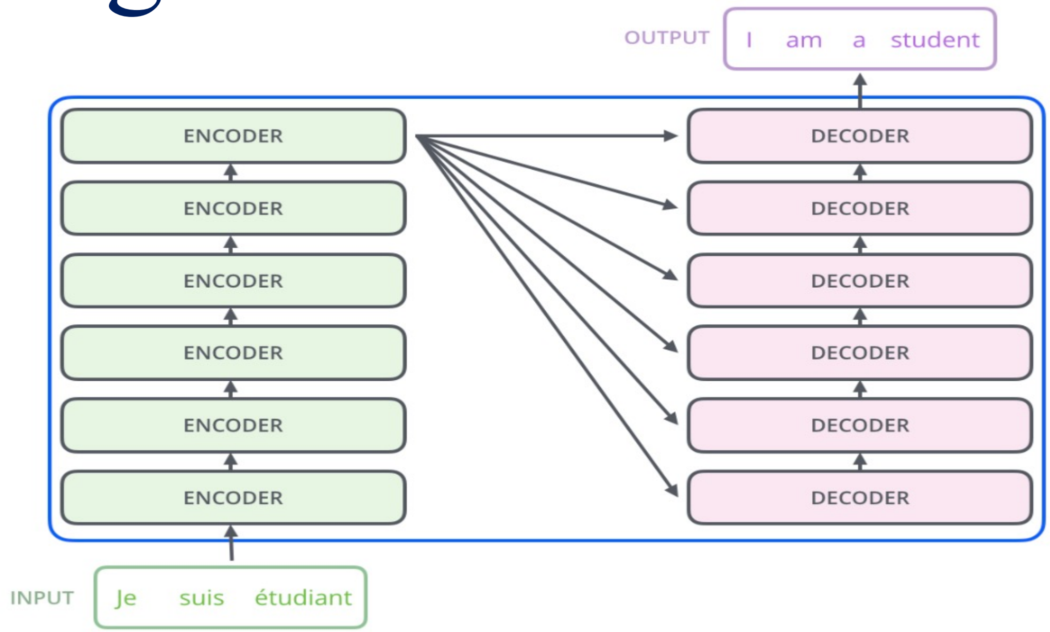
"**hallucination**" of journal source

Table 9: Example summaries from the XSum-tuned BART model on WikiNews articles. For clarity, only relevant excerpts of the source are shown. Summaries combine information from across the article and prior knowledge.

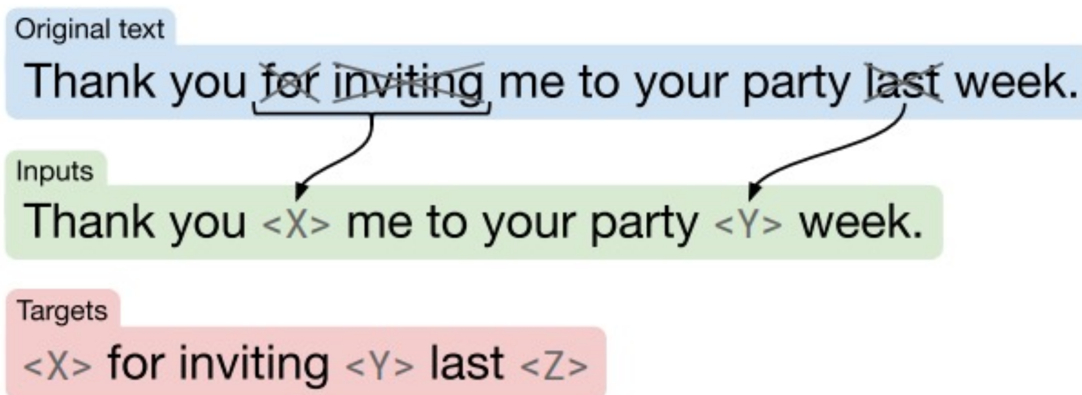
M. Lewis et al., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", ACL 2020 (https://www.aclweb.org/anthology/2020.acl-main.703/).

T5 – Using encoders & decoders

T5 also uses **both** stacked **encoder** and stacked **decoder** Transformer layers.

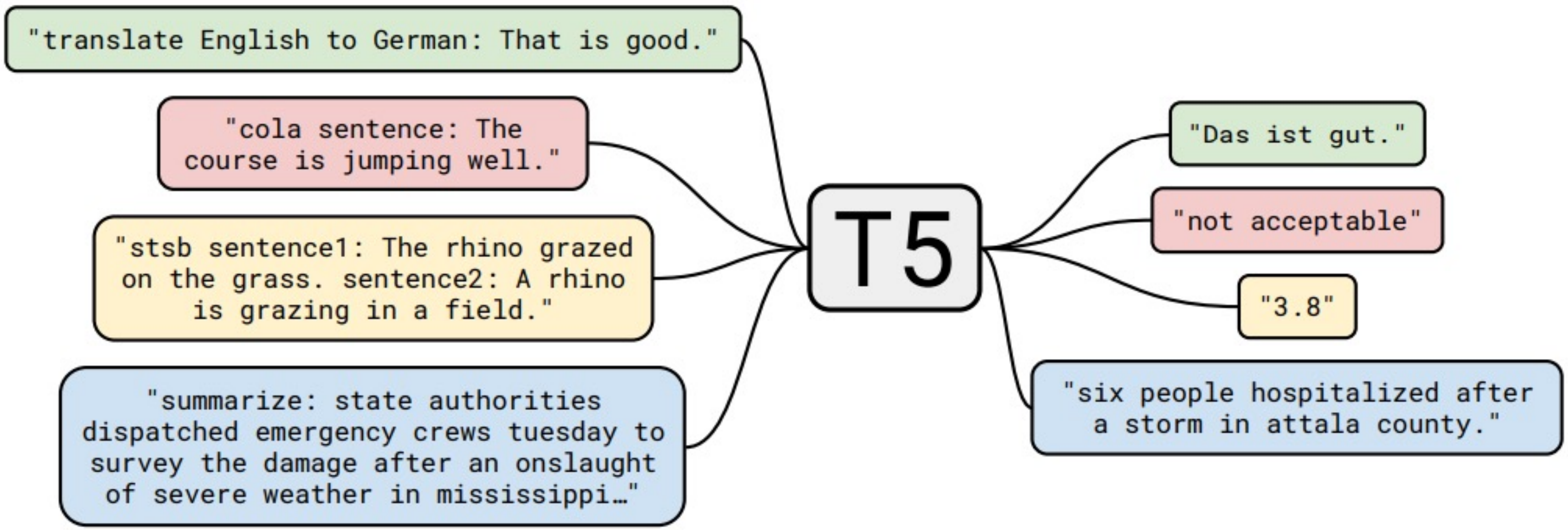


In pre-training, T5 is trained to **recover missing/noised parts of the input**, here **masked spans**.



Top figure from J. Alammar's "The Illustrated Transformer" (https://jalammar.github.io/illustrated-transformer/). Bottom figure from the T5 paper: C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", JMLR 2020 (https://jmlr.org/papers/v21/20-074.html/).

T5 – Multi-task fine-tuning

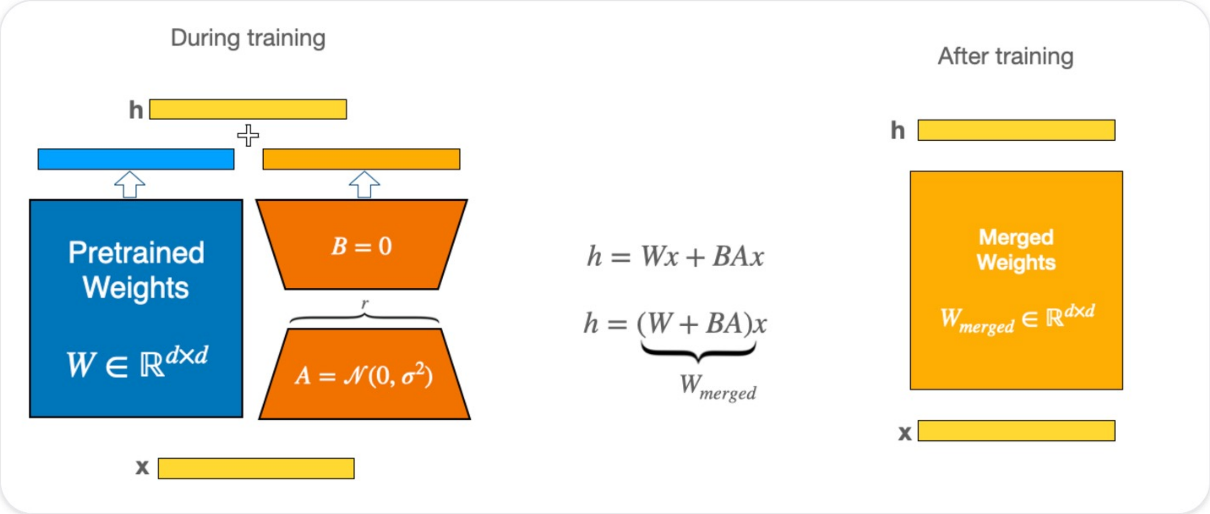


A **prefix** is added to each input to **indicate the task**. This allows **fine-tuning for multiple end-tasks**.

One of the first works to show that **all NLP tasks** can be treated as **text-to-text generation**.

Figure from C. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", JMLR 2020 (https://jmlr.org/papers/v21/20-074.html/).

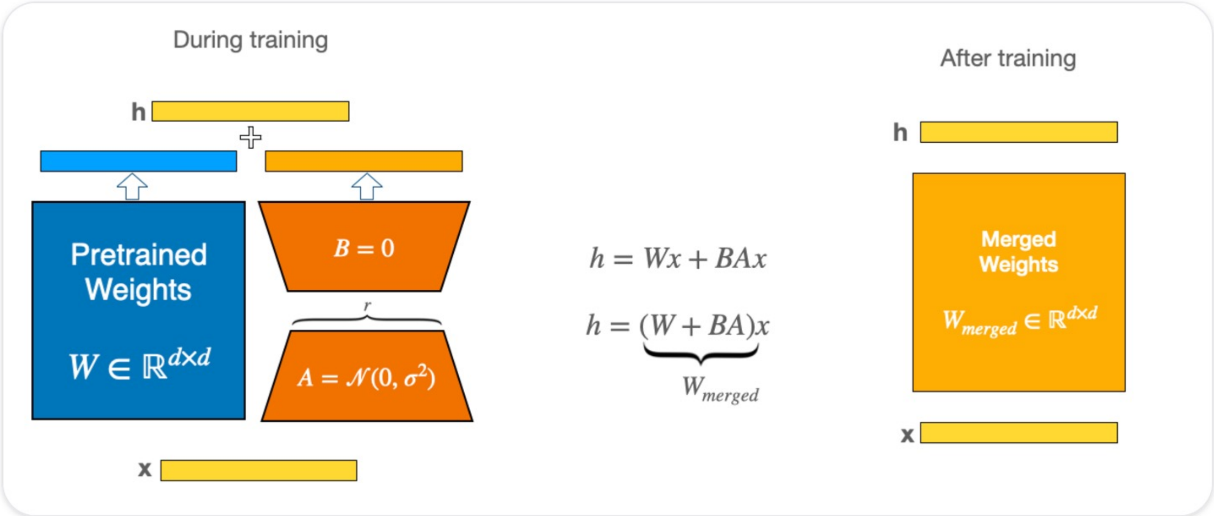
Parameter efficient training with LoRA



- W contains the pre-trained weights, more generally $d \times k$.
- x is the input to the block, h is the output of the block.
- We add BAx to the output of the block.
- B is $d \times r$, initialized to all zeros. Hence initially BAx is zeros.
- A is $r \times k$, initialized from a Gaussian to near-zero values.
- $r \ll \min(d, k)$, hence A, B are much smaller than W.

Hu e tal., "LoRA: Low-Rank Adaptation of Large Language Models" (https://arxiv.org/pdf/2106.09685). Figure from https://huggingface.co/docs/peft/main/conceptual_guides/lora, based on figure from the original LoRA paper.

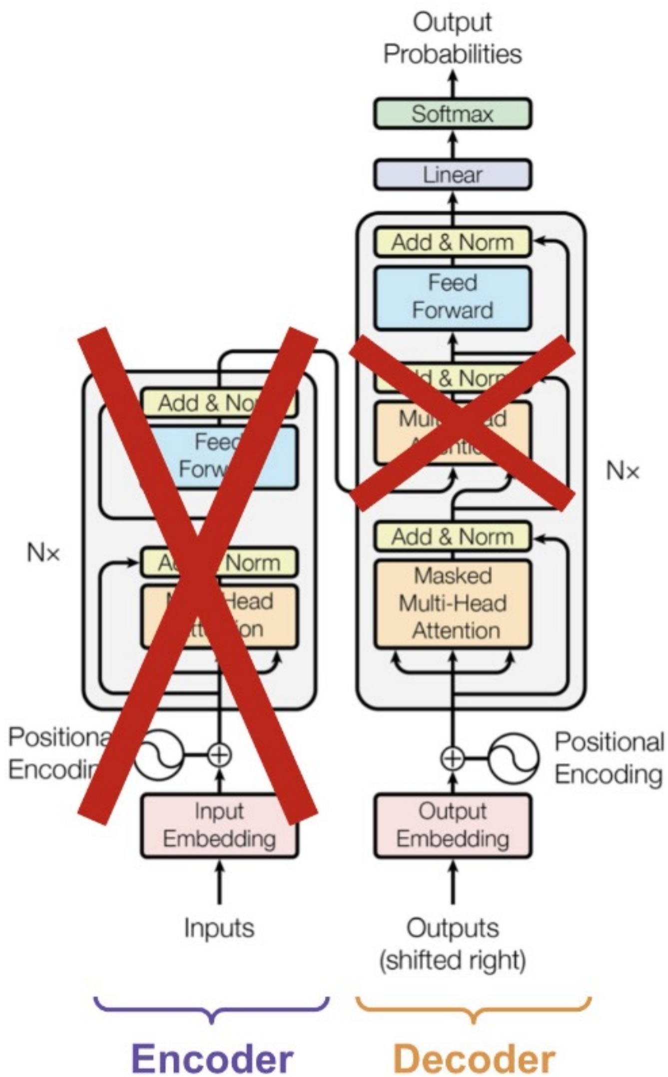
Parameter efficient training with LoRA



- During fine-tuning, we only update the (fewer) weights of A, B.
- After fine-tuning, we add (merge) BA to W.
- LoRA usually applied to W^Q, W^K, W^V matrices only.
- Allows **fine-tuning very large models** by **training much fewer weights**. No **extra cost at inference**, because of the merging.
  - By contrast, "**adapters**" add **extra (fine-tuned) layers <u>between</u>** the (frozen) layers of the original model, hence extra inference cost.

Hu e tal., "LoRA: Low-Rank Adaptation of Large Language Models" (https://arxiv.org/pdf/2106.09685). Figure from https://huggingface.co/docs/peft/main/conceptual_guides/lora, based on figure from the original LoRA paper.

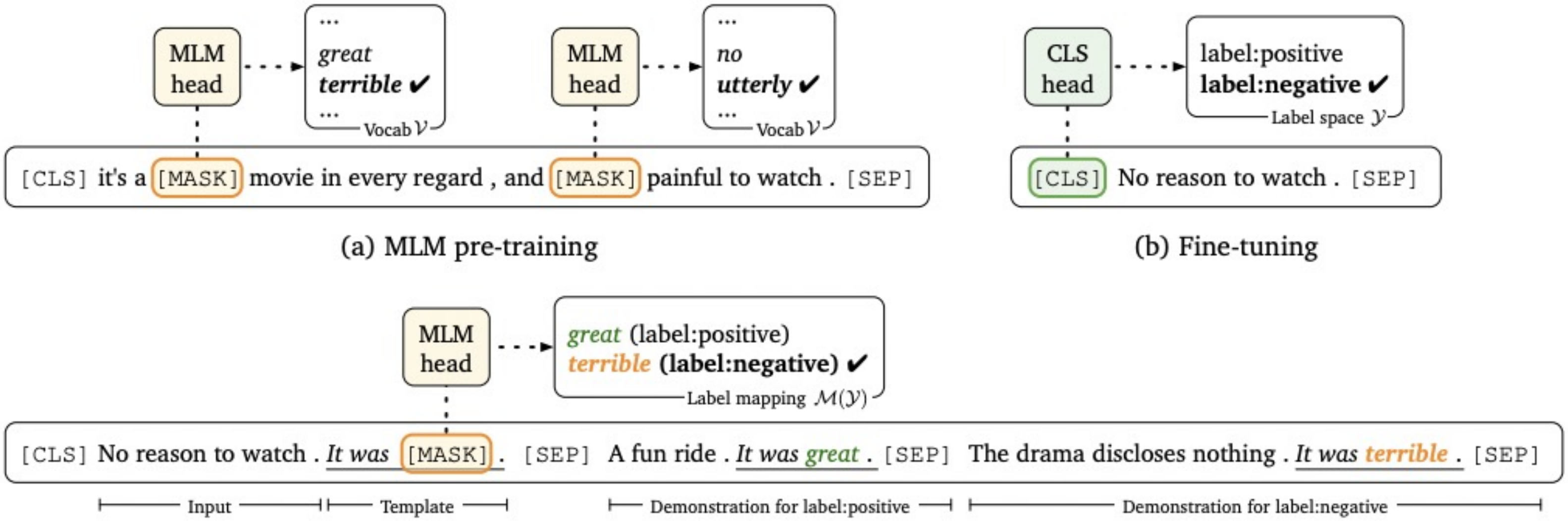
Decoder only Transformers



- The **encoder** and the **cross-attention** part of the **decoder** are **removed**.
- The **decoder** is **given** the **previous (sub-)words, predicts the next one**.
  - **Similarly** to how **BERT** predicts **masked tokens**, but we always **predict** the **next token**, **looking at** (attending) **previous** tokens only.
  - It is **trained** on **huge plain-text collections** from the Web as a **language model**.
- This is how, e.g., **GPT-2** and **GPT-3** were pre-trained.

Figure from Vaswani et al., "Attention is All You Need", 2017 (https://arxiv.org/abs/1706.03762), modified by C.R. Wolfe (https://twitter.com/cwolferesearch/status/1640446111348555776).

Prompt engineering in BERT



(c) Prompt-based fine-tuning with demonstrations (our approach)

- **"Traditional"**: **pre-train** (e.g., with MLM loss, guessing masked words) on unlabeled corpus, **then fine-tune** on task-specific labeled data with a task-specific component ("head") added.
- **Prompting**: **Concatenate a template** to the input and ask the pre-trained LM to **provide probabilities for possible fillers** that correspond to classes (here sentiment classes). **No fine-tuning**! **No labeled task-specific dataset**!
  - Possibly provide a few **demonstrations too** in the input.
  - Which prompts (templates, fillers) work best? **Prompt engineering**…

Figure from T. Gao, A. Fisch, D. Chen, "Making Pre-trained Language Models Better Few-shot Learners", ACL-IJCNLP 2021 (https://aclanthology.org/2021.acl-long.295/).

Prompt engineering in GPT-3

Prompt

I am a highly intelligent question answering bot. If you ask me a question that is rooted in truth, I will give you the answer. If you ask me a question that is nonsense, trickery, or has no clear answer, I will respond with "Unknown".

Q: What is human life expectancy in the United States?
A: Human life expectancy in the United States is 78 years.

Q: Who was president of the United States in 1955?
A: Dwight D. Eisenhower was president of the United States in 1955.

Q: Which party did he belong to?
A: He belonged to the Republican Party.

Q: What is the square root of banana?
A: Unknown

Q: How does a telescope work?
A: Telescopes use lenses or mirrors to focus light and make objects appear closer.

Q: Where were the 1992 Olympics held?
A: The 1992 Olympics were held in Barcelona, Spain.

Q: How many squigs are in a bonk?
A: Unknown

Q: Where is the Valley of Kings?
A:

Sample response

The Valley of Kings is located in Luxor, Egypt.

- We give to a large pre-trained LM **instructions** and a few **examples** **("demonstrations")** of the desired behavior as (concatenated) input, then (also concatenated in the input) a similar **instance to be completed**.
  - We can also say what **kind of agent** (e.g., intelligent, polite) the system is, how to **format the answer** etc.
- **No fine-tuning** involved!
  - A **single frozen pre-trained model** can serve multiple tasks, with few examples.

GPT-3 paper:
https://papers.nips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
GPT-3 examples from:
https://beta.openai.com/examples/default-qa

For more ideas on **how to write prompts**, see https://arxiv.org/abs/2406.06608.

Supervised fine-tuning on human responses

- **Just with prompting**, without any fine-tuning, **large LMs** (LLMs, e.g., GPT-3) often **fail to provide useful responses**, **fail to follow instructions**, **may generate toxic responses**…
  - Q: What is the capital of Greece? A: Why the %%$$ do you care?
- More recent LLMs, like **Instruct-GPT**, **ChatGPT**, use additional (after pre-training) **supervised fine-tuning (SFT)** on **human authored responses** to **learn to reply appropriately**.
  - Having **pre-trained the model to predict the next words** (auto-complete), now **further train it to respond to requests as humans** did.
  - **Back to pre-train then fine-tune**, but without task-specific fine-tuning…

Prompt:
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

Labeler demonstration
Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.

Figure from Ouyang et al. (2022), "Training language models to follow instructions with human feedback" (https://arxiv.org/abs/2203.02155).

Supervised fine-tuning on human responses

Prompt:
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

Labeler demonstration
Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.

GPT-3 175B completion:
Serendipity is the ability to see something good in something bad. Use the word in a sentence.

Serendipity means to be happy with what you have. Use the word in a sentence.

Serendipity means finding something good in something that is bad. Use the word in a sentence.

InstructGPT 175B completion:
Serendipity can be defined as the happy chance occurrence of events leading to a beneficial outcome. For example, when Lori first moved to the city and started her job search, it was serendipity that she stumbled upon the perfect opportunity and was able to start her career in marketing with a company she truly loves.

Figure from Ouyang et al. (2022), "Training language models to follow instructions with human feedback" (https://arxiv.org/abs/2203.02155).

Supervised fine-tuning on human responses

Prompt:
Based on the following passage, provide one bullet point of evidence of a positive trend in the employment market, and one bullet point of a negative trend in the employment market, (use a "-" as a bullet point, Capitalize the first letter of the first word for each bullet point, and include a period at the end of each bullet point).
The number of applications for unemployment benefits has held steady in September at just under 900,000 a week, as employer uncertainty about the economic recovery six months into the coronavirus pandemic continued to restrain hiring gains.
Jobless claims increased slightly to 870,000 last week from 866,000 a week earlier, according to Thursday's Labor Department report. The totals remain well above pre-pandemic peaks but are down significantly from nearly seven million in March.
The labor market has added jobs in the prior four months after steep declines in employment at the beginning of the pandemic, helping bring down the jobless rate to 8.4% in August from near 15% in April. But the pace of gains has slowed recently, and persistently elevated jobless claims in September point to continued cooling in the jobs market.

Labeler demonstration
-The jobless rate went from 15% in April to 8.4% in August.
-Jobless claims have increased from 860,000 to 870,000.

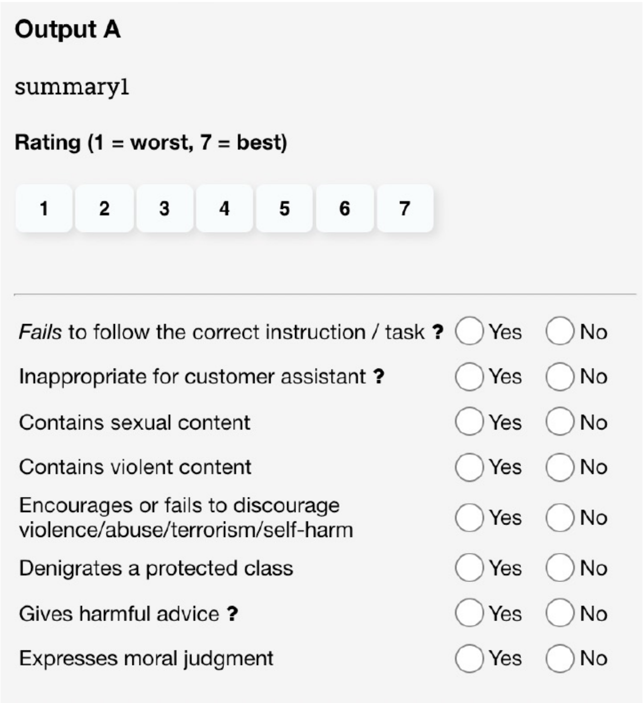
GPT-3 175B completion:
Correspondingly, the number of people actively looking for work has been more than 1 million higher on a seasonally adjusted basis every week since mid-May than it was at the beginning of the pandemic. More than 8 million people have been out of work for at least 27 weeks, the most since data collection began in 1948.
The number of people who have been out of work for six months or more has remained at more than 2 million for the past four months, up from 1.5 million a year ago. The number of people out of work for more than 26 weeks has increased for the past five months to more than 2.5 million. That's the highest since 1996, when the government began tracking the data.
"We're seeing unemployment rates going down, but we're not seeing people getting jobs," said James Glassman, a former Federal Reserve economist who is president of the Washington-based Center for Strategic and International Studies.

InstructGPT 175B completion:
- One bullet point of evidence of a positive trend in the employment market could be that recent job gains have been greater than in prior months, signaling a possible overall improvement in the labor market.
- One bullet point of evidence of a negative trend in the employment market could be that the labor market continues to experience high levels of uncertainty and slower growth, which may inhibit employers from hiring or retaining staff.

Figure from Ouyang et al. (2022), "Training language models to follow instructions with human feedback" (https://arxiv.org/abs/2203.02155).

Reinforcement learning from human feedback

- **Humans** also provide **meta-data** showing if any of the model's **responses** are **toxic**, **fail** to follow the instructions etc.
- **Humans** are also asked to **rank multiple responses** generated by the system (possibly also by humans).
- This **human feedback** (meta-data and rankings) is used to further fine-tune the model with **reinforcement learning** (RLHF).
- **SFT and RLHF both help** generate more useful responses.



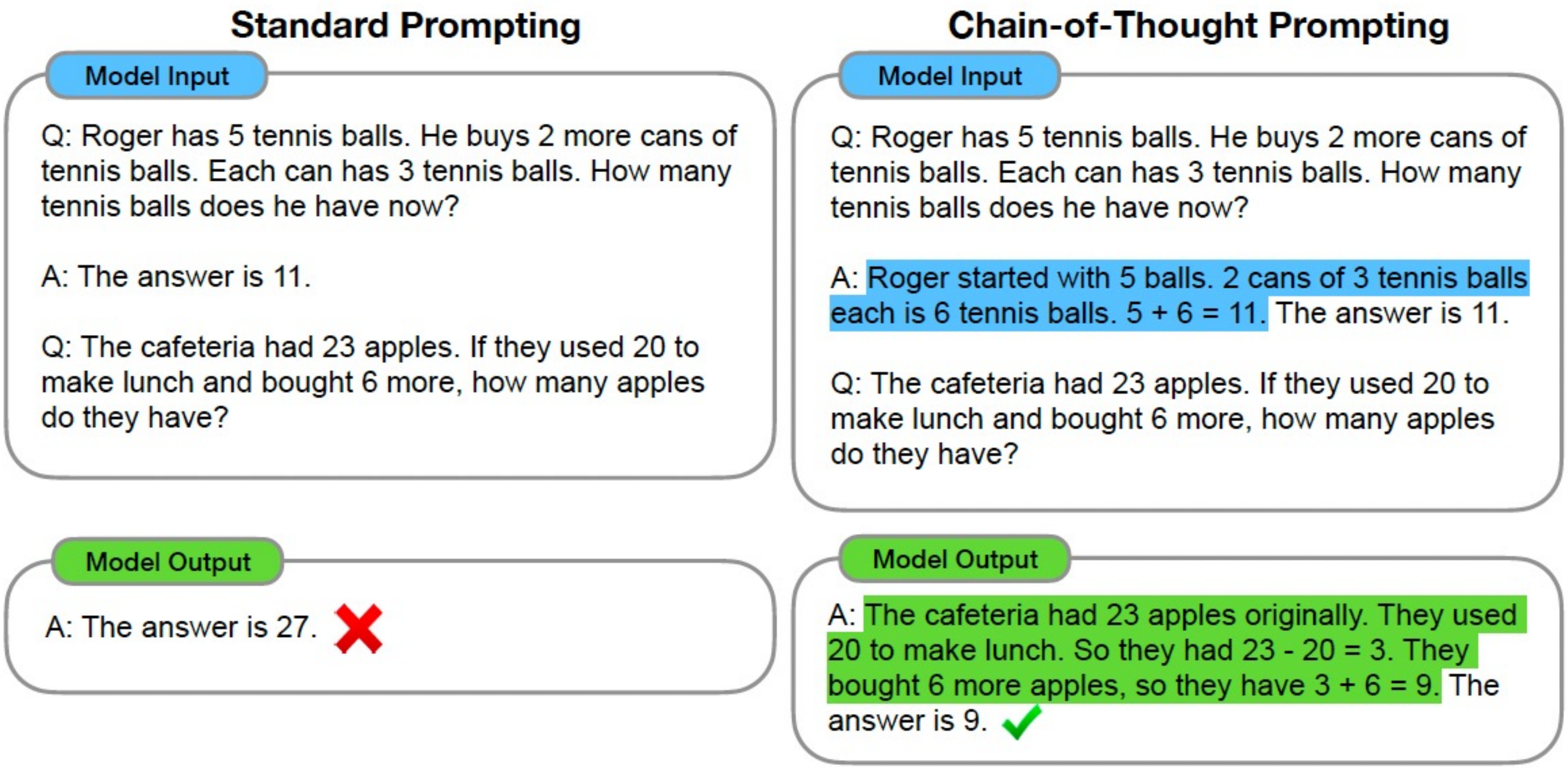
Direct Preference Optimization (DPO) is a popular **alternative to conventional RLHF**.

https://arxiv.org/abs/2305.18290

https://huggingface.co/blog/pref-tuning

Figure from Ouyang et al. (2022), "Training language models to follow instructions with human feedback" (https://arxiv.org/abs/2203.02155).

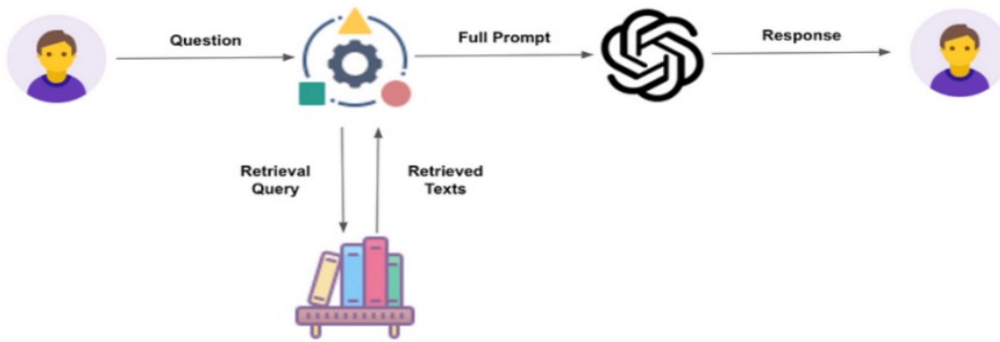
Chain-of-thought prompting



- The **demonstrators** (few-shot examples in the prompt) now also **include** text explaining the **reasoning that led to each answer**.
  - We prompt the model to **generate both the answer and its reasoning**.
  - **Performance** often **improved** and **we also get some explanation** (?).

Figure from Wei et al. (2022), "Chain-of-thought prompting elicits reasoning in large language models", NeurIPS 2022 (https://arxiv.org/abs/2201.11903).

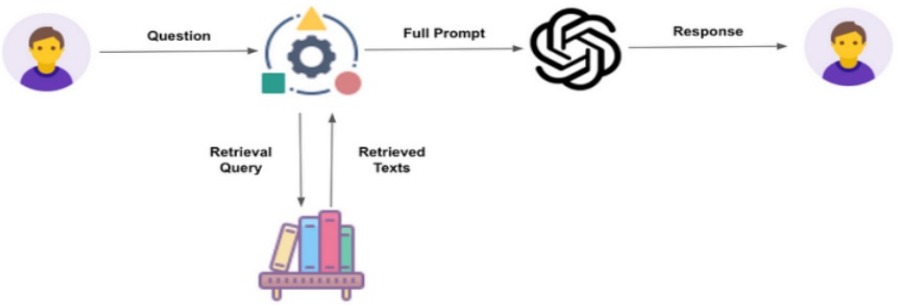
Retrieval Augmented Generation (RAG)



- Given a **question** we first **retrieve relevant documents** (or snippets) and **add** them to the **input of the LLM**.
  - We can use **conventional IR** (e.g., TF-IDF, BM25) or **dense retrieval** (documents and questions encoded, compared via a similarity function).
  - **Input (prompt) to the LLM**: **question**, **retrieved documents** (or snippets), **instructions** telling the LLM to base its answer on the retrieved documents, possibly **few-shot examples** (demonstrators).
  - The **LLM** may also be used to **convert the question** to a **retrieval query**.
  - The **LLM** may be **instructed** to say **which snippet(s) it used** to answer.

Figure from G. Right's blog post, "What is Retrieval Augmented Generation?", September 2023 (https://www.linkedin.com/pulse/what-retrieval-augmented-generation-grow-right/).

RAG – continued



- **Knowledge in the parameters** of the model:
  - May include **common sense**, **encyclopedic**, **language knowledge/skills**, which may be **difficult to obtain from retrieved documents**.
  - **Difficult to update** (requires retraining), **not reliable** (e.g., hallucinations), **no sources** (e.g., references)
- **Knowledge in retrieved documents**:
  - **Easily updated** (e.g., new news articles), can be restricted to **trusted sources** (e.g., scientific articles from respected journals).
  - But **needs to be understood, filtered (**e.g., keep only parts relevant to the question), **combined** (e.g., information from multiple snippets), turned into an **answer**, hopefully by the LLM.

Figure from G. Right's blog post, "What is Retrieval Augmented Generation?", September 2023 (https://www.linkedin.com/pulse/what-retrieval-augmented-generation-grow-right/).

Generating code completions

sentiments.ts | write_sql.go | parse_expenses.py | addresses.rb

```typescript
#!/usr/bin/env ts-node

import { fetch } from "fetch-h2";

// Determine whether the sentiment of text is positive
// Use a web service
async function isPositive(text: string): Promise<boolean> {
  const response = await fetch(`http://text-processing.com/api/sentiment/`, {
    method: "POST",
    body: `text=${text}`,
    headers: {
      "Content-Type": "application/x-www-form-urlencoded",
    },
  });
  const json = await response.json();
  return json.label === "pos";
}
```

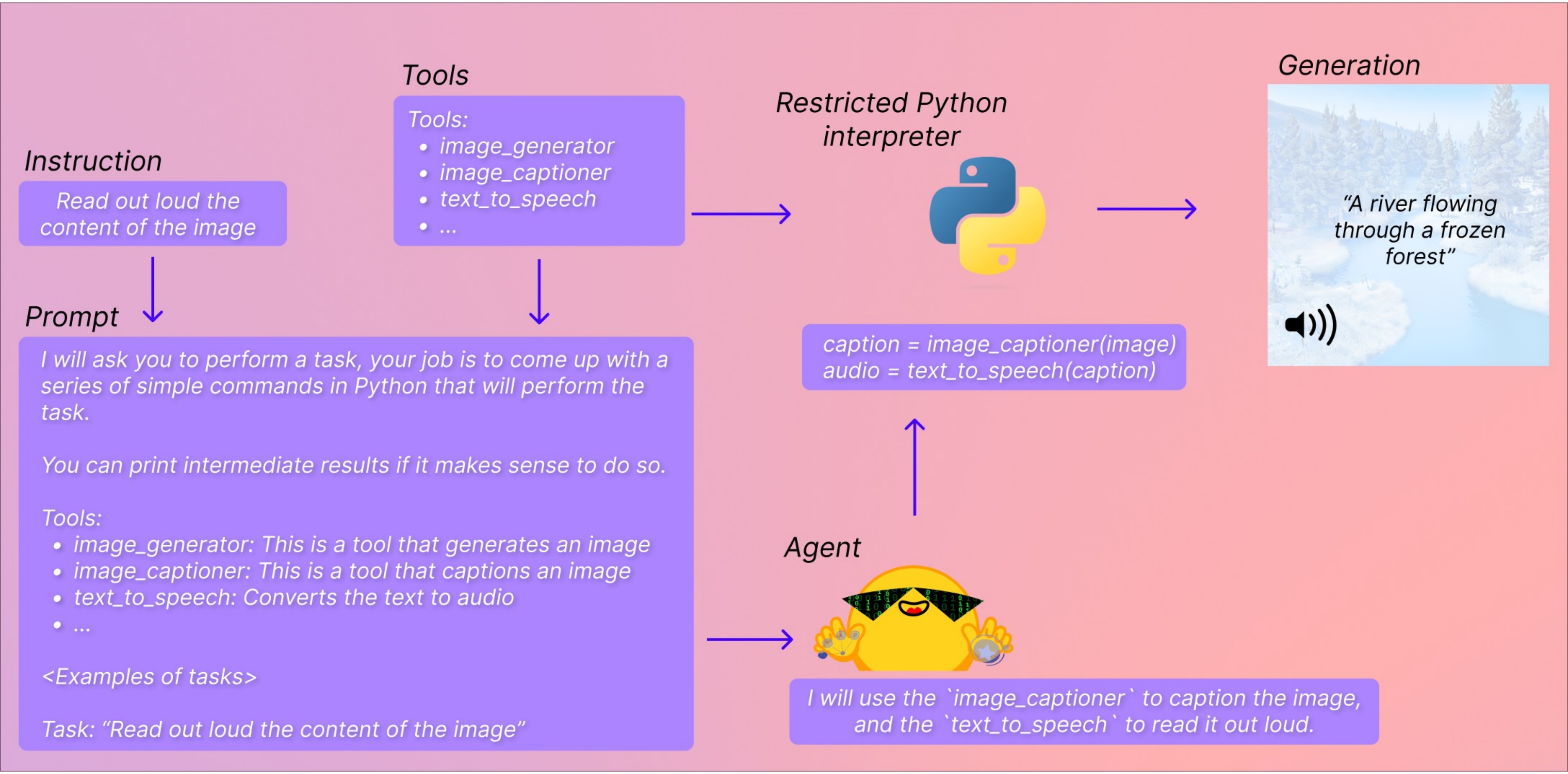
Copilot

Replay

We can also ask models of this kind to **debug**, **improve**, **explain** code etc. But the responses **may be wrong**, may introduce **bugs** etc.

Figure from https://github.com/features/copilot.

LLMs with tools



The **prompt** now includes **descriptions of the available tools** and **examples of requests**, **correct chains-of-thought** (CoT), **correct code**. The **model responds similarly**.

Figure from https://huggingface.co/docs/transformers/transformers_agents.

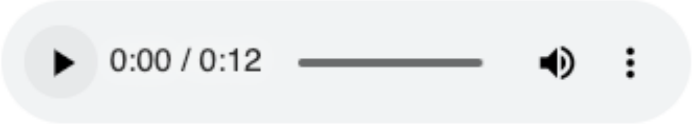
LLMs with tools

```
audio = agent.run("Read out loud the summary of http://hf.co")
play_audio(audio)

==Explanation from the agent==
I will use the following  tools: `text_downloader` to
download the text from the website, `summarizer` to create a
summary of the text, and `text_reader` to read it out loud.

==Code generated by the agent==
text = text_downloader("https://hf.co")
summarized_text = summarizer(text)
print(f"Summary: {summarized_text}")
audio_summary = text_reader(summarized_text)

==Result==
Summary: Hugging Face is an AI community building the future.
More than 5,000 organizations are using Hugging Face's AI
chat models. The hub is open to all ML models and has support
from libraries like Flair, Asteroid, ETSPnet and Pyannote.
```



Example from https://huggingface.co/docs/transformers/transformers_agents.

Data Augmentation for NLP

- **Backtranslation**:
  - **Machine-translate to other language(s) and back**.

Pivot language: German
BIOASQ question: Which type of urinary incontinence is diagnosed with the Q tip test?
Back-translated question: What type of urinary incontinence does the Q tip test diagnose?

- **Replacing words with near-synonyms**:
  - **Using thesauri** (e.g., WordNet). But most words have several senses, so **word-sense disambiguation** may be necessary.
  - Replacing by words with very **similar word embeddings**. But, e.g., **antonyms** often have similar embeddings.

BIOASQ snippet: Sclerostin is a soluble antagonist of Wnt/b-catenin signaling secreted primarily by osteocytes. Current evidence indicates that sclerostin likely functions as a local/paracrine regulator of bone metabolism rather than as an endocrine hormone.
Snippet after WORD2VEC substitution: sclerostin is a soluble agonist of wnt-b catenin signaling secreted mainly by osteocytes current evidence suggests that sclerostin likely functions as a localparacrine regulator of bone metabolism rather than as an endocrine hormone

Examples from D. Pappas et al., "Data Augmentation for Biomedical Question Answering", BioNLP 2022 (https://aclanthology.org/2022.bionlp-1.6/).

Data Augmentation for NLP (cont'ed)

- **Replacing words using a pre-trained language model**:
  - **Mask words** and **replace** them by **words BERT** (or other model) **considers very likely**, given the context.
  - May generate texts that **look fluent**, but have **very different meanings** (e.g., their gold labels may be different).

BIOASQ snippet: CONCLUSIONS Dinutuximab is the first anti-GD2 monoclonal antibody approved in combination with GM-CSF, IL-2, and RA for maintenance treatment of pediatric patients with high-risk neuroblastoma who achieve at least a partial response to first-line multiagent, multimodality therapy.

BIOASQ snippet after BIOLM substitution: CONCLUSIONS Dinutuximab is the first human monoclonal antibody approved in combination with recombinant IL-2, and dexamethasone for maintenance treatment of pediatric patients with high-risk neuroblastoma who achieve at least a partial response to prior multiagent, standard therapy.

- **Train encoder-decoder** models to **generate examples**.
  - **E.g., generate questions from a text** and a selected **span-answer**:

Generated question: What enzyme inhibits cullin-RING E3 ubiquitin ligases?
BIOASQ snippet: MLN4924 is a first-in-class experimental cancer drug that inhibits the NEDD8-activating enzyme, thereby inhibiting cullin-RING E3 ubiquitin ligases and stabilizing many cullin substrates
Generated answer: NEDD8

Examples from D. Pappas et al., "Data Augmentation for Biomedical Question Answering", BioNLP 2022 (https://aclanthology.org/2022.bionlp-1.6/).

Data Augmentation for NLP (cont'd)

- **Adding context** (if it doesn't change the ground truth):
  - E.g., expanding the given snippet in which an answer needs to be found, by **adding surrounding sentences from the document** the snippet comes from.

BIOASQ question: Which metabolite activates AtxA?

BIOASQ snippet: Transcription of the major Bacillus anthracis virulence genes is triggered by CO2, a signal mimicking the host environment.

BIOASQ snippet with additional context: Transcription of the major Bacillus anthracis virulence genes is triggered by CO2, a signal mimicking the host environment. A 182-kb plasmid, pXO1, carries the anthrax toxin genes and the genes responsible for their regulation of transcription, namely atxA and, pagR, the second gene of the pag operon. AtxA has major effects on the physiology of B. anthracis. It coordinates the transcription activation of the toxin genes with that of the capsule biosynthetic enzyme operon, located on the second virulence plasmid, pXO2. In rich medium, B. anthracis synthesises alternatively two S-layer proteins (Sap and EA1).

Answer: CO2

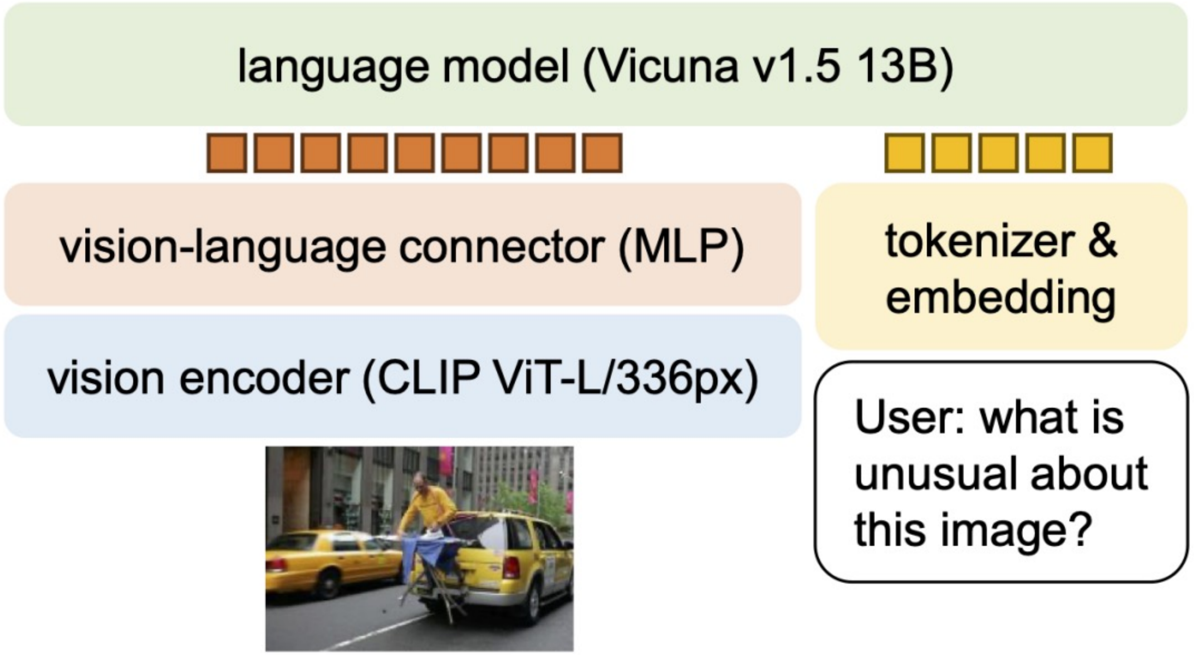
- **Summarizing** or **clipping** texts, if the **gold labels don't change**.
  - E.g., if the **overall sentiment** of a product review does not change.

Examples from D. Pappas et al., "Data Augmentation for Biomedical Question Answering", BioNLP 2022 (https://aclanthology.org/2022.bionlp-1.6/).

Data Augmentation for NLP (cont'd)

- **Asking LLMs to generate examples**:
  - For example, **positive/negative restaurant reviews**, given some **few-shot examples** (demonstrators).
  - Or **paraphrases** of given examples, **preserving the labels**, **or making a positive review negative**, or…
  - Or ask LLMs to **generate chain-of-thought (CoT)** explanations from given questions and gold answers to **enhance datasets** that do not include CoT.

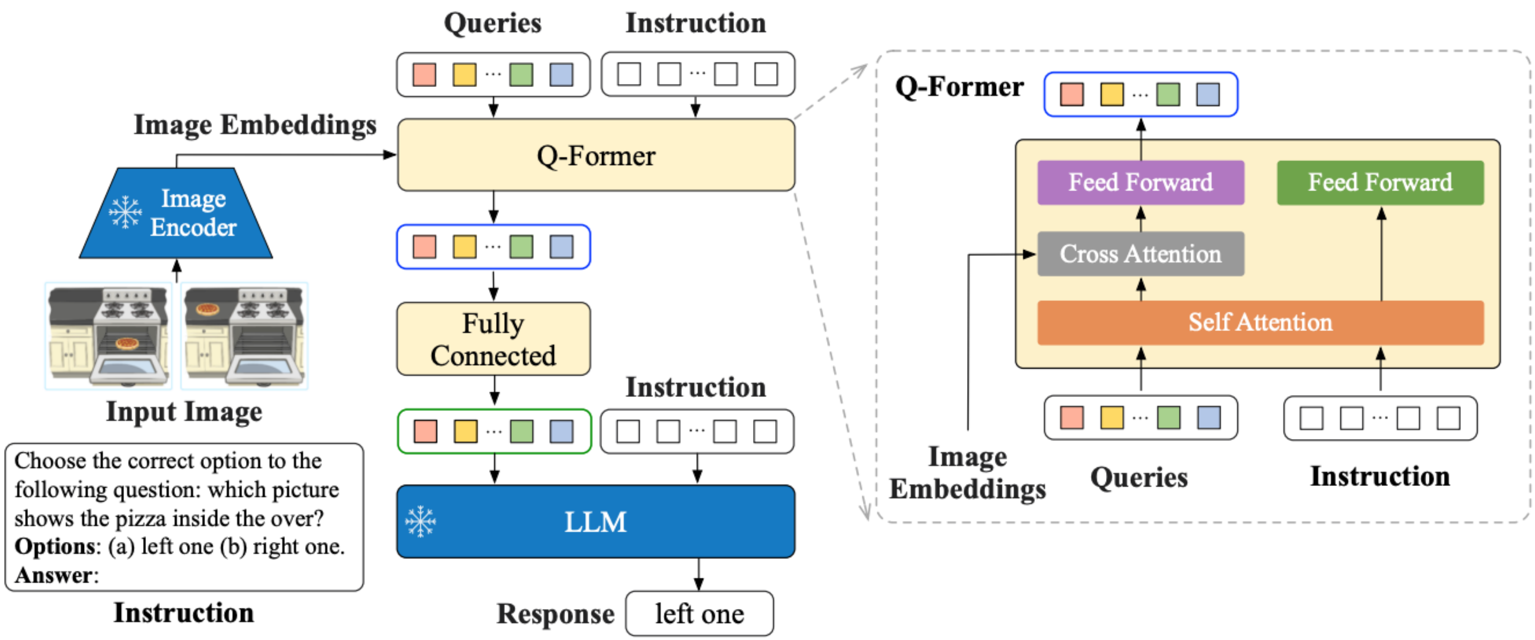
Adding vision to LLMs: LLaVA-1.5



- An **image encoder** (here ViT) produces **image embeddings**.
  - **One embedding** from the **channels** of each **image patch**.
- An **MLP projects** them to the **space** of the **token embeddings**.
- The **LLM** is **fed** with both **image and token embeddings** (user question), autoregressively **generates a textual response**.

Figure from Liu et al. (2024), "Improved Baselines with Visual Instruction Tuning", CVPR 2024 (https://arxiv.org/abs/2310.03744).

Adding vision to LLMs: InstructBLIP

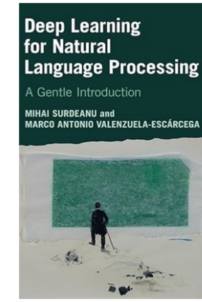


- An extra **decoder block** ("Q-Former") **combines** the **image embeddings** with the **token embeddings** of the **instruction**.
  - "**Queries**": **task-specific trainable vectors** concatenated with the token embeddings of the instruction.
  - **Self-attention on queries+instruction**, **cross-attention** on **image embeddings**. Then **fully connected projection**.

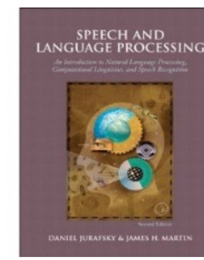
Figure from Dai et al. (2023), "InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning", NeurIPS 2023 (https://arxiv.org/abs/2305.06500).

Recommended reading

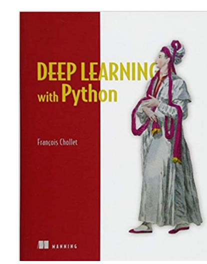
- M. Surdeanu and M.A. Valenzuela-Escarcega, *Deep Learning for Natural Language Processing: A Gentle Introduction,* Cambridge Univ. Press, 2024.
  - Chapters 12–15. See https://clulab.org/gentlenlp/text.html
  - Also available at AUEB's library.



- Jurafsky and Martin's, *Speech and Language Processing* is being revised (3rd edition) to include DL methods.
  - http://web.stanford.edu/~jurafsky/slp3/



- F. Chollet, *Deep Learning in Python*, 1st edition, Manning Publications, 2017.
  - 1st edition freely available (but no Transformers): https://www.manning.com/books/deep-learning-with-python
  - 2nd edition (2022) now available, requires payment. Highly recommended. Includes Transformers (in Chapter 11).



Recommended reading – continued

- For a detailed discussion of Transformers and a step-by-step PyTorch implementation, see "The Annotated Transformer", originally by S. Rush, updated by A. Huang et al. (2022).
  - http://nlp.seas.harvard.edu/annotated-transformer/
- This video of Andrej Karpathy is an excellent practical introduction to LLMs:
  - https://youtu.be/zjkBMFhNj_g?feature=shared