Transformers: Architecture, Training & Usage

Binxu Wang

Apr. 18th, 2023

MLFS Tutorial

Part I
Natural Language Processing 101

Where Transformer and self-supervised learning first triumphed

Two Pillars of NLP

Representation

- How to represent language to machines?

Modelling

- How to model language statistically?

Simultaneously solved by DL models,
e.g. transformers

Models have different emphases

BERT

Openai-embedding

GPT

ChatGPT

Idea adapted from MIT 6.8610

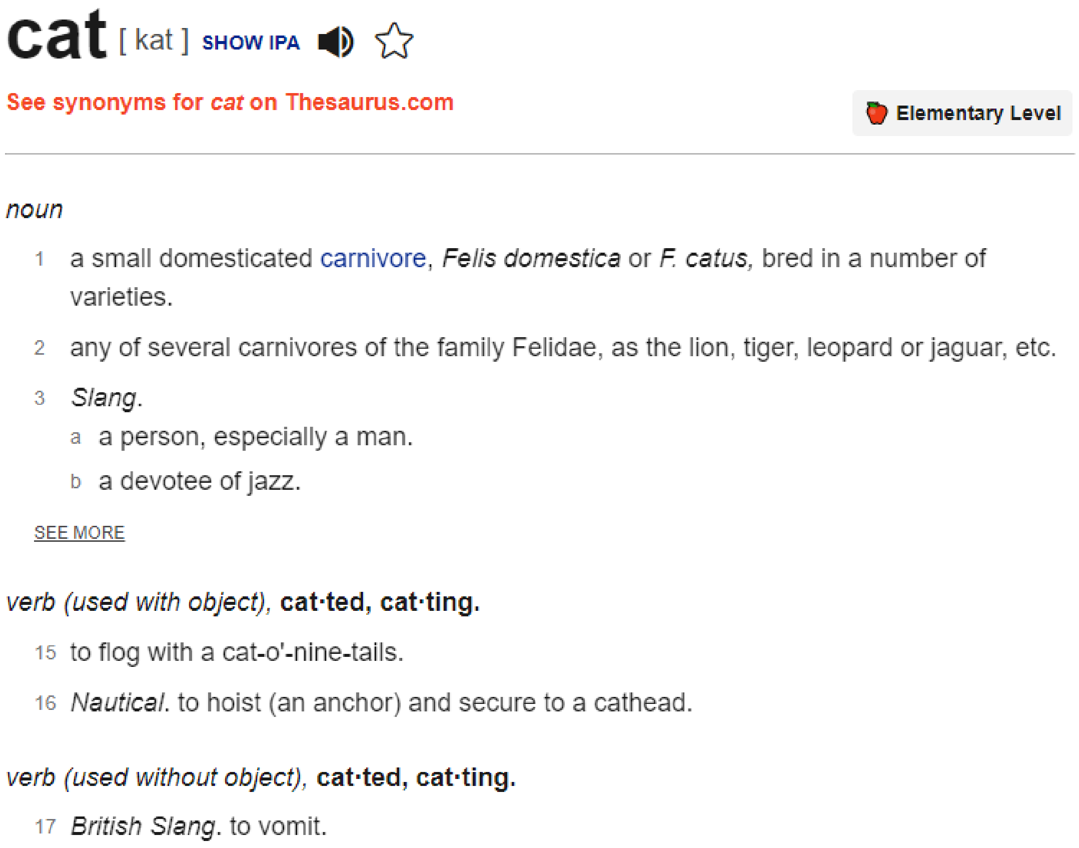
Representation: Words

- Unlike pixel, meaning of word are not explicitly in the characters.

- Word can be represented as number index
  - But indices are also meaningless.

- List of feature attributes (dictionary entry)
  - {Part of Speech, description, usage,…}
  - What are those features, how to design those?

Words in a sentence	I	love	cats	and	dogs	.
Token Index	328,	793,	3989,	537,	3255,	269



cat [kat] **SHOW IPA**

See synonyms for *cat* on Thesaurus.com

Elementary Level

noun

1 a small domesticated carnivore, *Felis domestica* or *F. catus*, bred in a number of varieties.

2 any of several carnivores of the family Felidae, as the lion, tiger, leopard or jaguar, etc.

3 *Slang*.
  a a person, especially a man.
  b a devotee of jazz.

SEE MORE

verb (used with object), **cat·ted, cat·ting.**

15 to flog with a cat-o'-nine-tails.

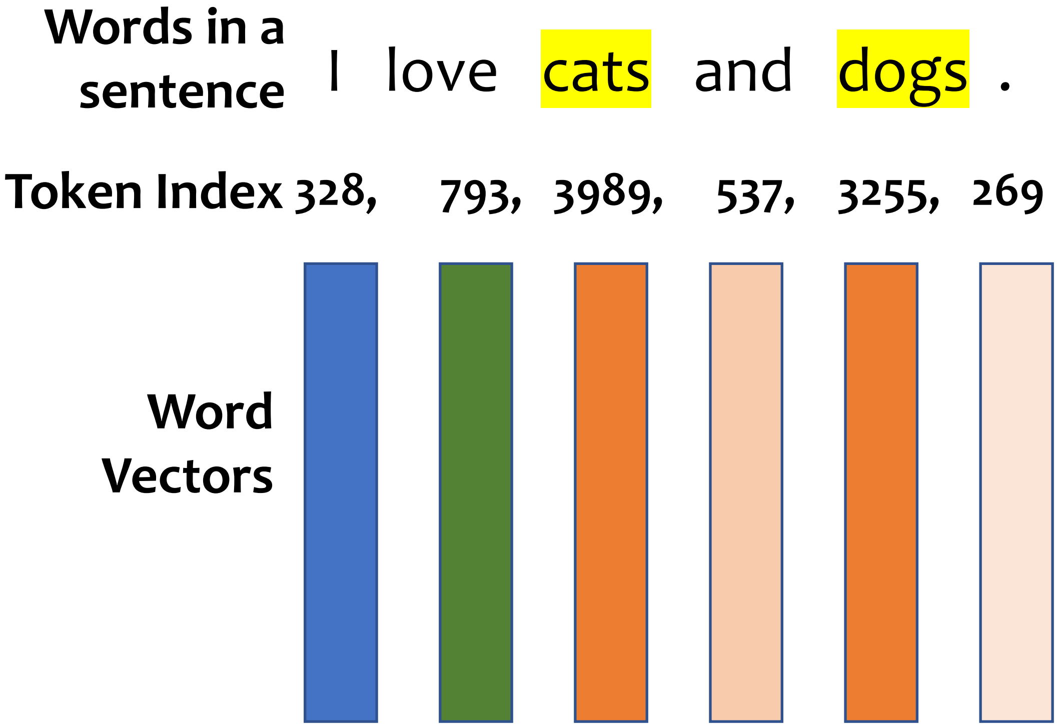
16 *Nautical*. to hoist (an anchor) and secure to a cathead.

verb (used without object), **cat·ted, cat·ting.**

17 *British Slang*. to vomit.

Representation: Word as Vectors

- Represent words in a vector space
  - Encode features in a distributed way.
  - Vector distance ⇒ similarity
  - Vector geometry ⇒ relation.
- Desirable features
  - Part of speech / usage (*Noun*)
  - Relation to other words (*cat – cats - kitten*)
  - Perceptual feature (for object.) (*cats are cute*)

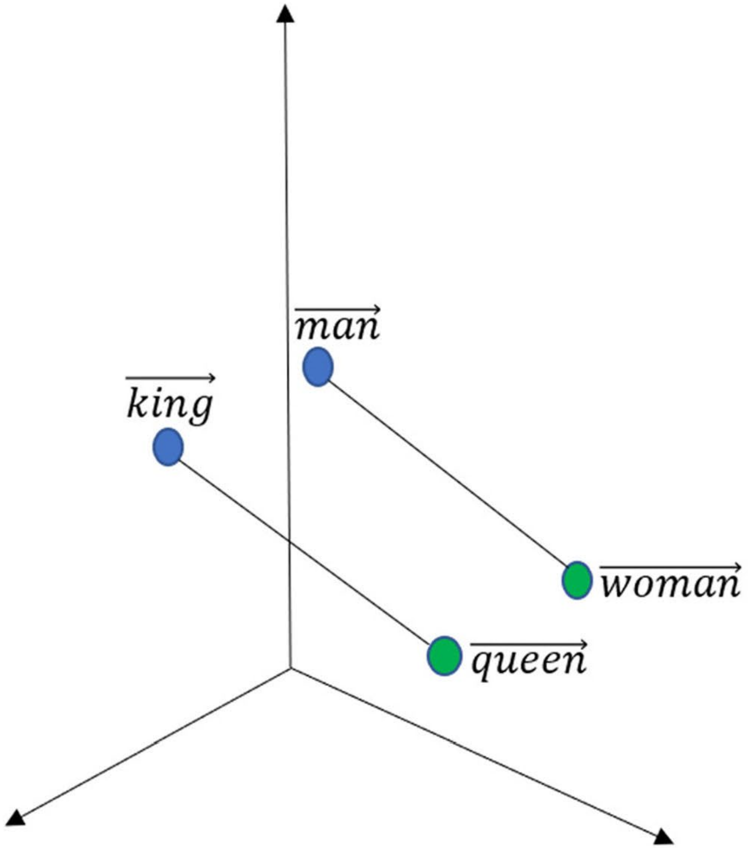


Words in a sentence I love cats and dogs .

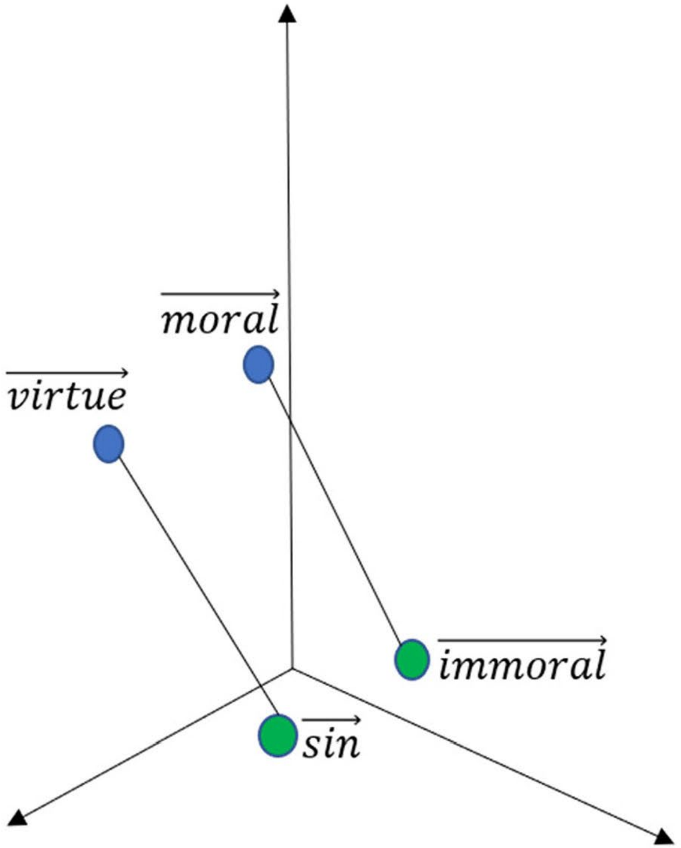
Token Index 328, 793, 3989, 537, 3255, 269

Word as Vectors:

Vector space geometry captures semantic relationship



a “gender” dimension

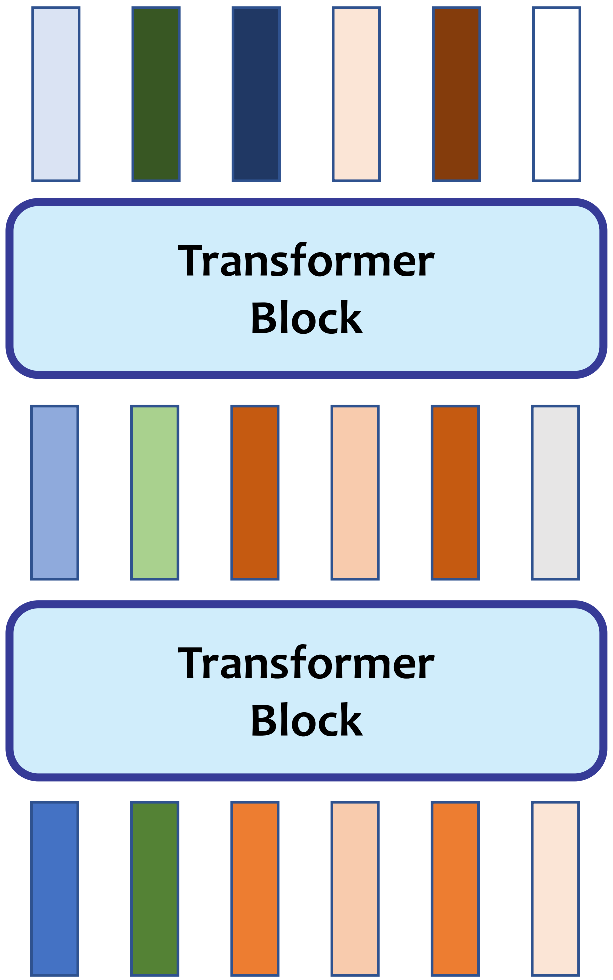


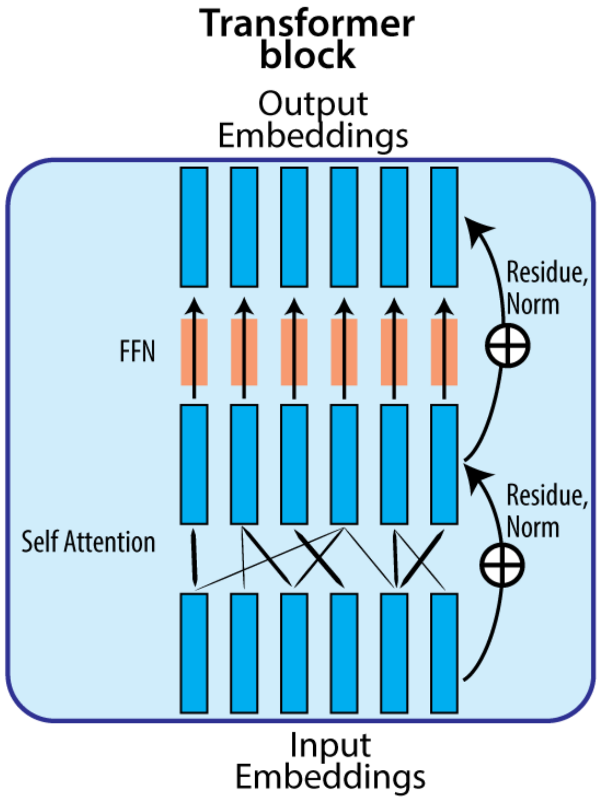
a “morality” dimension

Arseniev-Koehler, Alina. "Theoretical foundations and limits of word embeddings: what types of meaning can they capture?." *Sociological Methods & Research* (2021)

Word Vector in Context:
RNN / Transformers

- Meaning of word depends on context.
    - “I can buy a can of fish.”
- Word vectors should depend on context, not just word itself.
- Transformers / RNN let each word “absorb” influence from other words to be *contextualized*

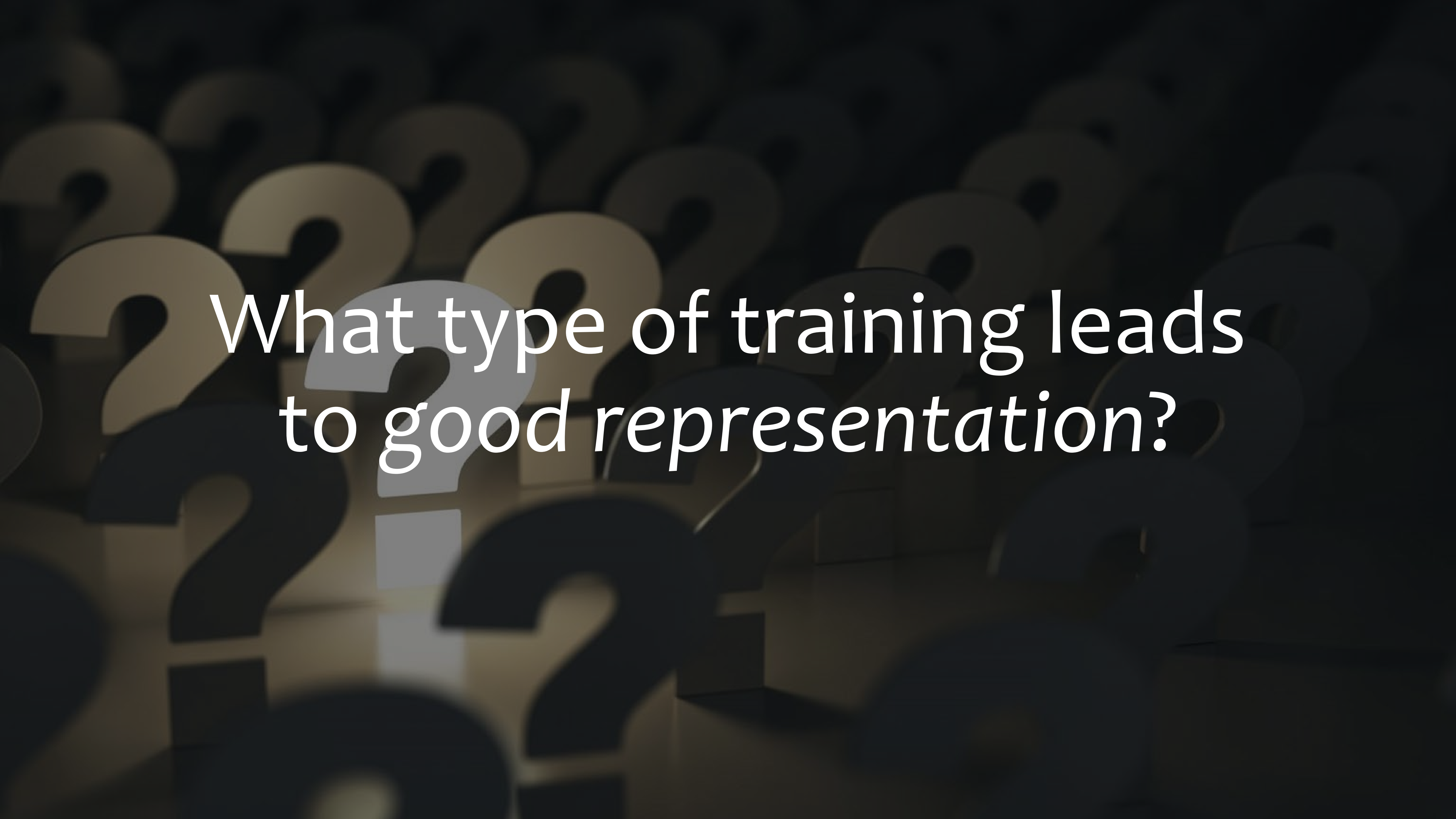




Word Representation can help …

- Text classification / understanding
  - Sentiment analysis (*happy or angry*)
  - Text similarity assessment
  - Information retrieval (*find most similar text*)
  - Machine translation

Slide adapted from MIT 6.8610



What type of training leads to *good representation*?

Language model

- LM predicts the likelihood of any sequence of words, for a given *language corpus**.
  - $p(\text{"This is a fluffy dog."}) = p(w_1 w_2 w_3 w_4 w_5) = 0.132 \ldots$
  - $p(\text{"This are a purple flying dear."}) = p(w_1 w_2 w_3 w_4 w_5 w_6) = 0.0002 \ldots$

Note:
Each person / author has their individual LM. English ***marginalize*** *over those.*

LM of Shakespeare
LM of Trump

Language model

- Given such model, we can use conditional probability to
  - Guess missing words
    - "This __ a fluffy dog. “
    - $\arg\max p(w_2|w_1w_3w_4w_5) \rightarrow "is"$
  - Predict next words
    - "This is a fluffy ...
    - $\arg\max p(w_5|w_1w_2w_3w_4) \rightarrow "cat“$
  - Answer questions
    - "The Vatican locates in the city of ...
    - $\arg\max p(w_7|w_1w_2w_3w_4w_5w_6) \rightarrow "Rome“$

Remark:

By modelling statistics, language model implicitly learns grammar, semantics, common sense, factual knowledge…

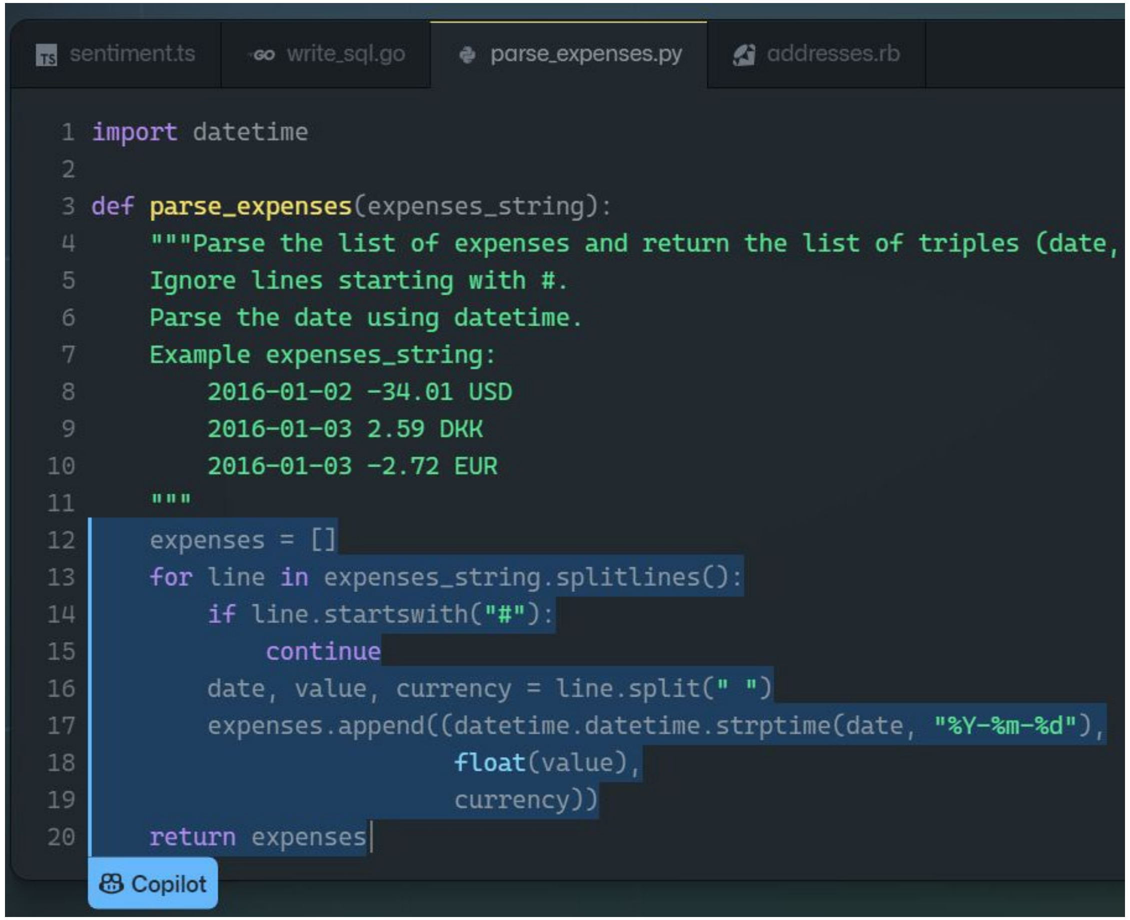
Language model can help …

- Text generation
  - Code completion
  - Chat bot
  - Text summary
  - Text correction
  - Machine translation

- Text classification
  - Detect spam
  - Authorship attribution
  - (*per likelihood comparison of two language models*)

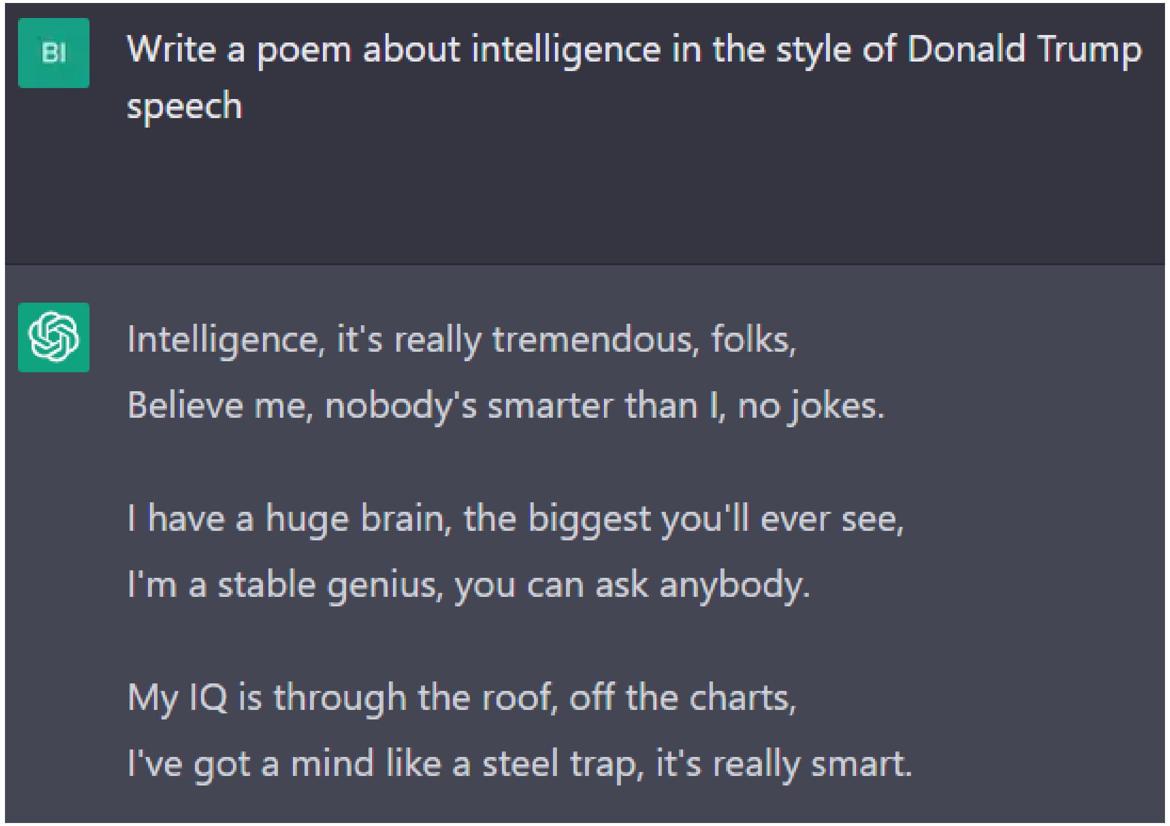
Slide adapted from MIT 6.8610

E.g. Github Copilot, ChatGPT



```python
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date,
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2016-01-02 -34.01 USD
        2016-01-03 2.59 DKK
        2016-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                         float(value),
                         currency))
    return expenses
```



Write a poem about intelligence in the style of Donald Trump speech

Intelligence, it's really tremendous, folks,
Believe me, nobody's smarter than I, no jokes.

I have a huge brain, the biggest you'll ever see,
I'm a stable genius, you can ask anybody.

My IQ is through the roof, off the charts,
I've got a mind like a steel trap, it's really smart.

Given a text collection, how to learn **language model**?

$$p(\text{"This is a fluffy dog."}) = p(w_1 w_2 w_3 w_4 w_5)$$

- *Joint density of a sequence is hard to model…*

- **Autoregressive LM:** Sequential factorization

$$p(w_1 w_2 w_3 w_4 w_5) = p(w_1)p(w_2|w_1)p(w_3|w_1 w_2)p(w_4|w_1 w_2 w_3)p(w_5|w_1 w_2 w_3 w_4)$$

  - Model **next word probability** given all previous words.

- *Classic example*: Bi-gram / N-gram model

$$p(w_1 w_2 w_3 w_4 w_5) \approx p(w_1)p(w_2|w_1)p(w_3|w_2)p(w_4|w_3)p(w_5|w_4)$$

  - Assume next word only depends on the last word

- **Masked LM:**

$$f(M|w_1 w_2 w_4 w_5) \propto p(w_1 w_2 M w_4 w_5)$$

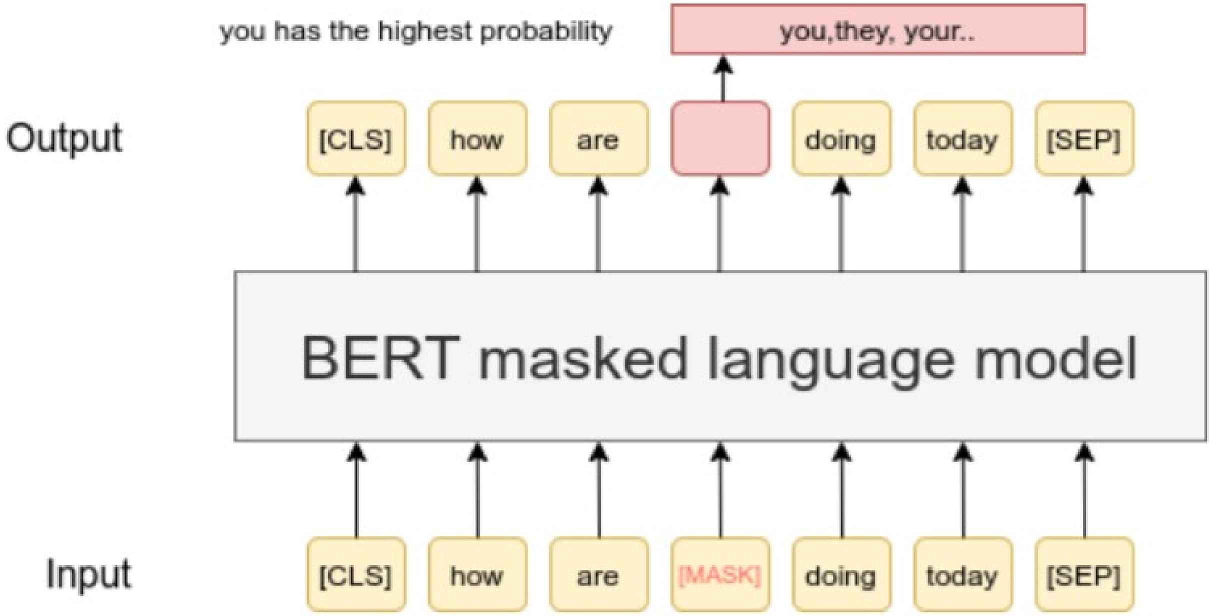
  - Model the probability of a missing word given all contexts.

Quiz:
How to *train* a bi-gram language model?

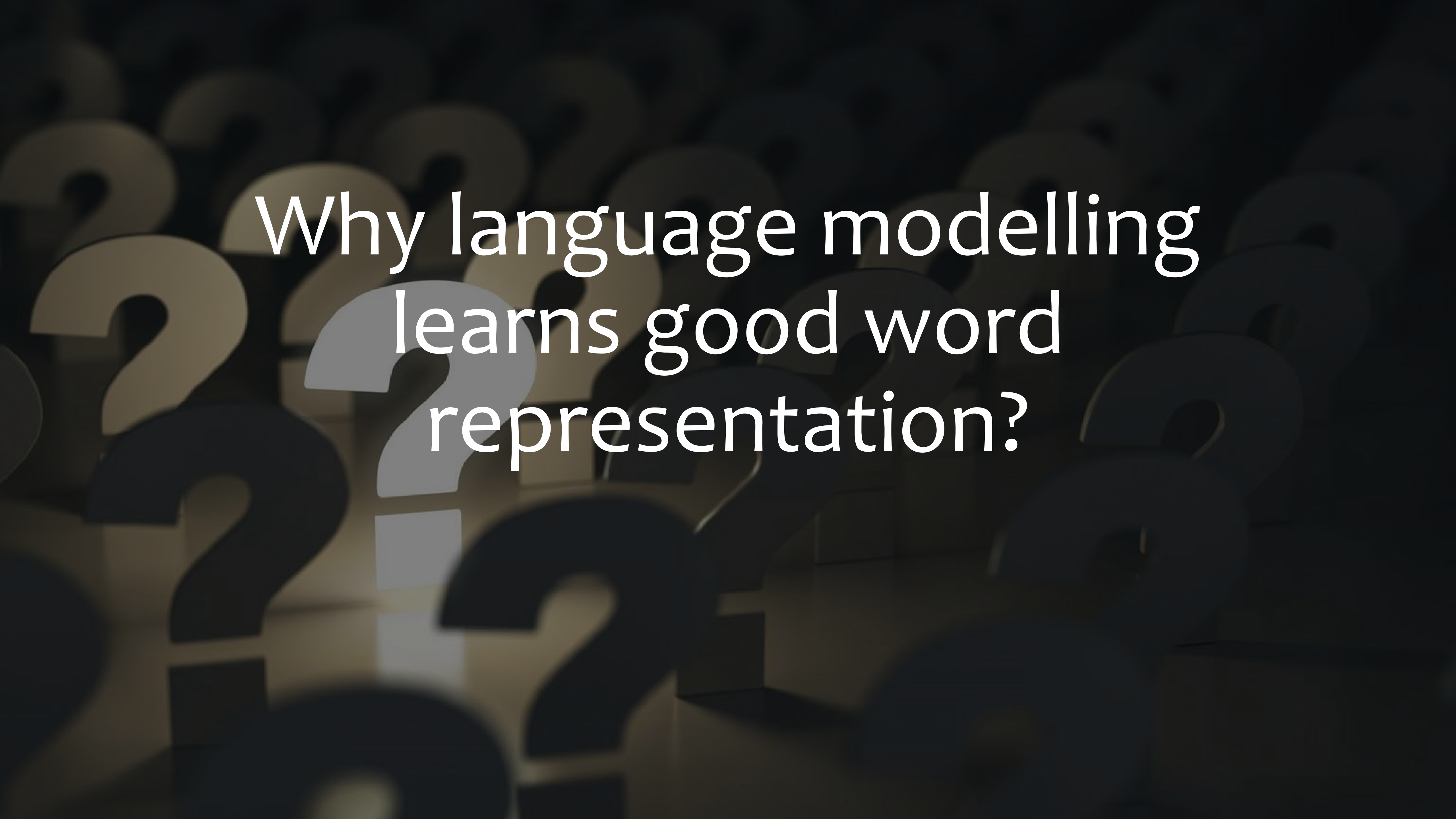
Learning Word Vectors via *Language Modelling*

BERT, GPT & CLIP

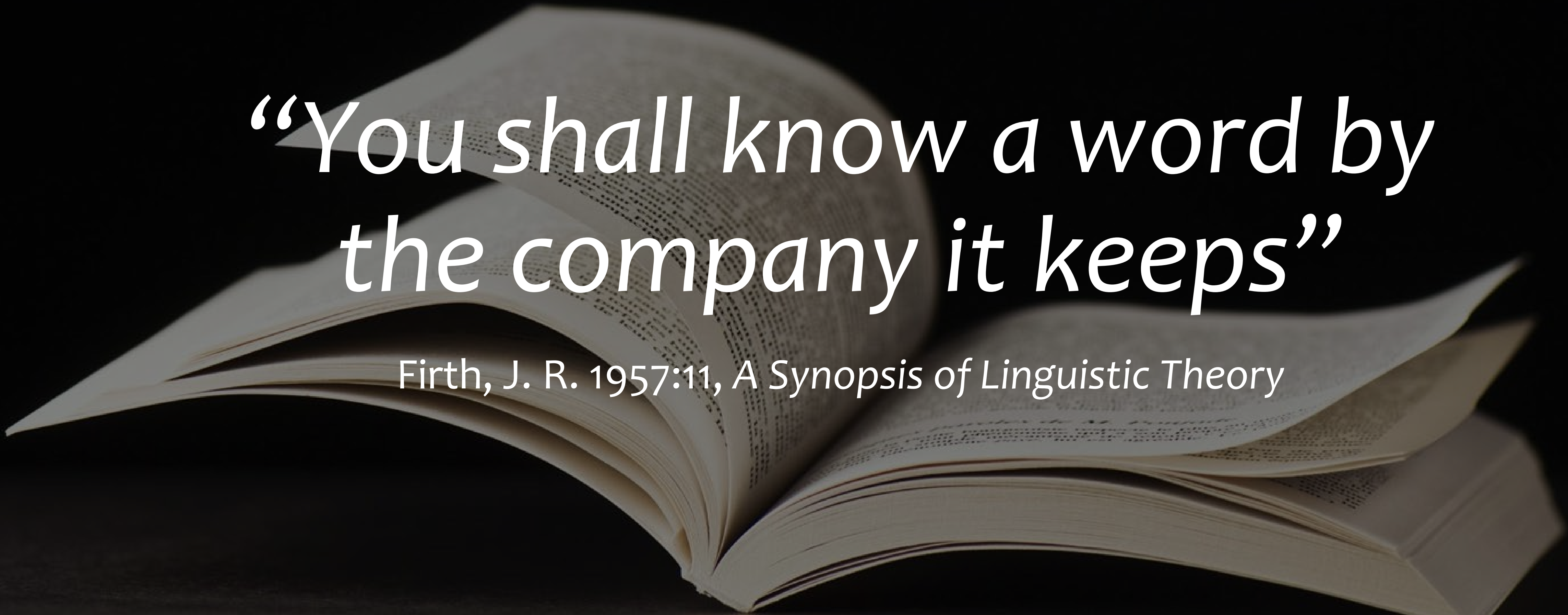
- *Self-supervised learning* of word representation
  - **Autoregressive language modelling:** predict next word (GPT)
  - **Masked language modelling:** predict missing words (BERT)
  - **Multimodal Contrastive Learning:** matching image and text. (CLIP)



MLM — Sentence-Transformers documentation (sbert.net)



Why language modelling learns good word representation?



“You shall know a word by the company it keeps”

Firth, J. R. 1957:11, *A Synopsis of Linguistic Theory*

Similar words have similar statistical effect on context

- "I love jogging, so __"
- "I like running, so __"

Similar text may follow …

- The *meaning* of a word **manifests** in its *effect on its companion…*
  - If two words (A, B) are interchangeable in all context

$$p(w_1 w_2 A w_4 w_5) = p(w_1 w_2 B w_4 w_5) \ \ \forall w_1, w_2, w_4, w_5$$

A and B have the same meaning.

  - So they should have similar word vector.

Interim Summary:

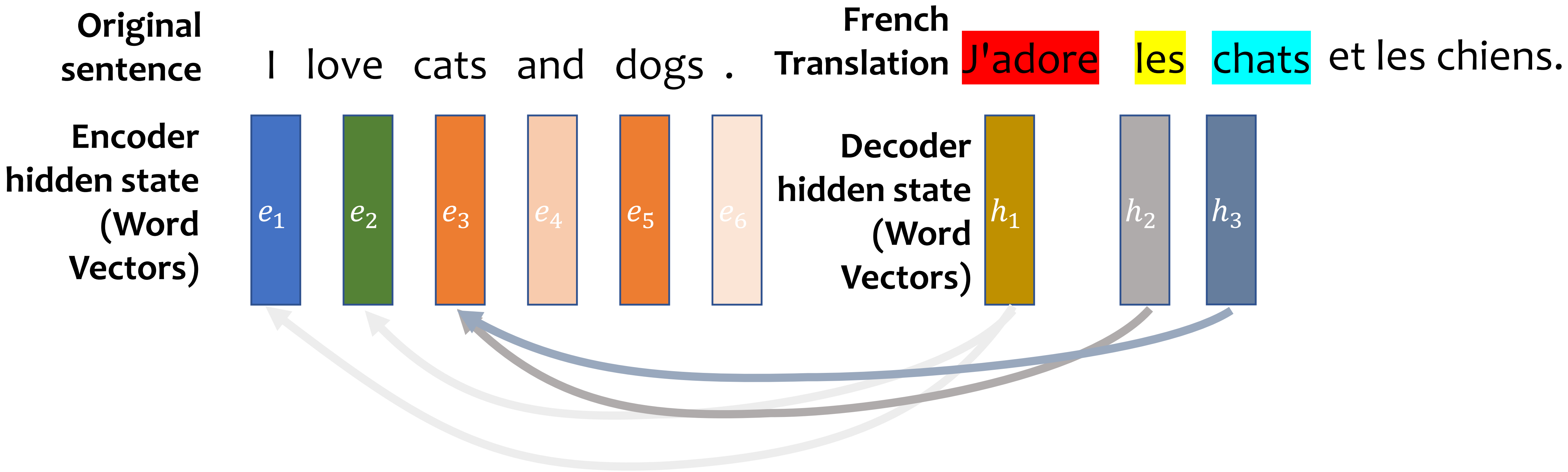
- **Final goal :**
  - A language model
  - A *good contextual representation* of words, sentence, paragraph.

- **Learning objective:**
  - Self-supervised learning (SSL), i.e. Language modelling
    - Predict missing words / next words
  - Multi-modal supervision.

- **Model architecture**
  - RNN (LSTM, GRU), Transformer, etc.

Part II
Attention & Transformer

Origin of Attention:
Machine Translation (Seq2Seq)


Original sentence
I love cats and dogs .
French Translation
J'adore les chats et les chiens.
Encoder hidden state (Word Vectors)
e_1
e_2
e_3
e_4
e_5
e_6
Decoder hidden state (Word Vectors)
h_1
h_2
h_3

- Use **Attention** to retrieve **relevant info** from a batch of vectors.

How to retrieve relevant information?

From dictionary to feature based attention.

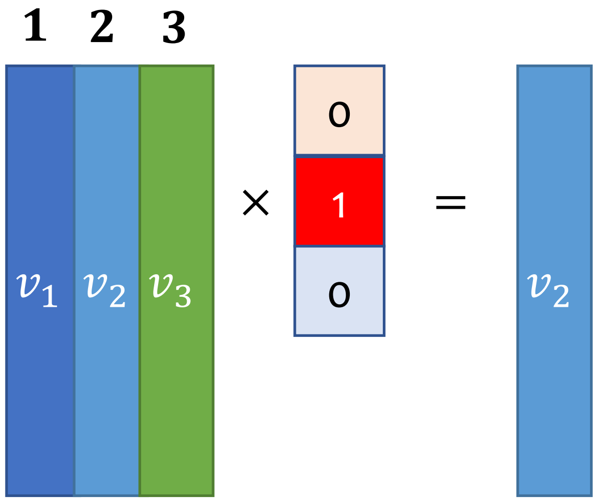
From Dictionary to Attention

Dictionary: Hard-indexing

- Dictionary maps keys to values
    - `dic = $\{1 : v_1, 2 : v_2, 3 : v_3\}$`
    - Keys 1,2,3
    - Values v_1, v_2, v_3

- Query retrieves information related to a key
    - `dic[2]` Query 2
    - Find 2 in keys
    - Get corresponding value.

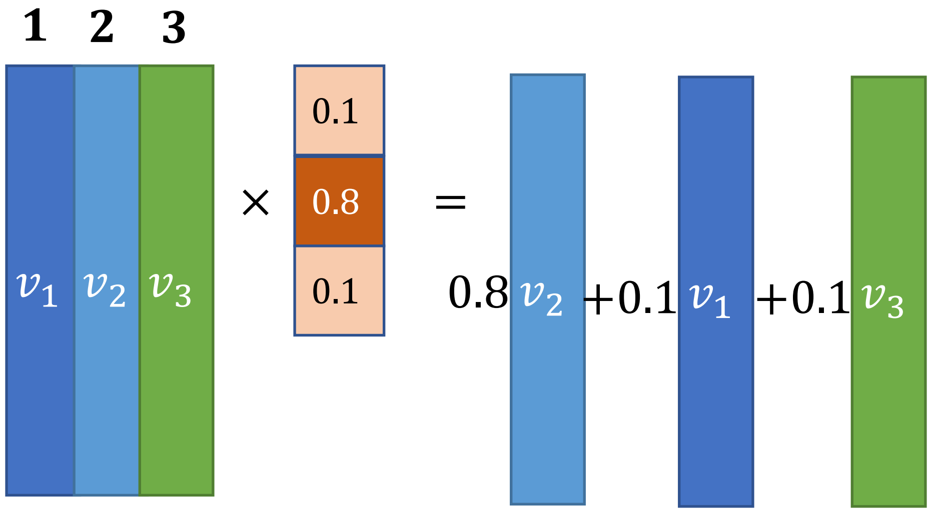
- Retrieving values as matrix vector product
    - $\boldsymbol{a}$ is one hot vector over the keys
    - Weighted sum the value vectors.



From Dictionary to Attention

Attention: Soft-indexing

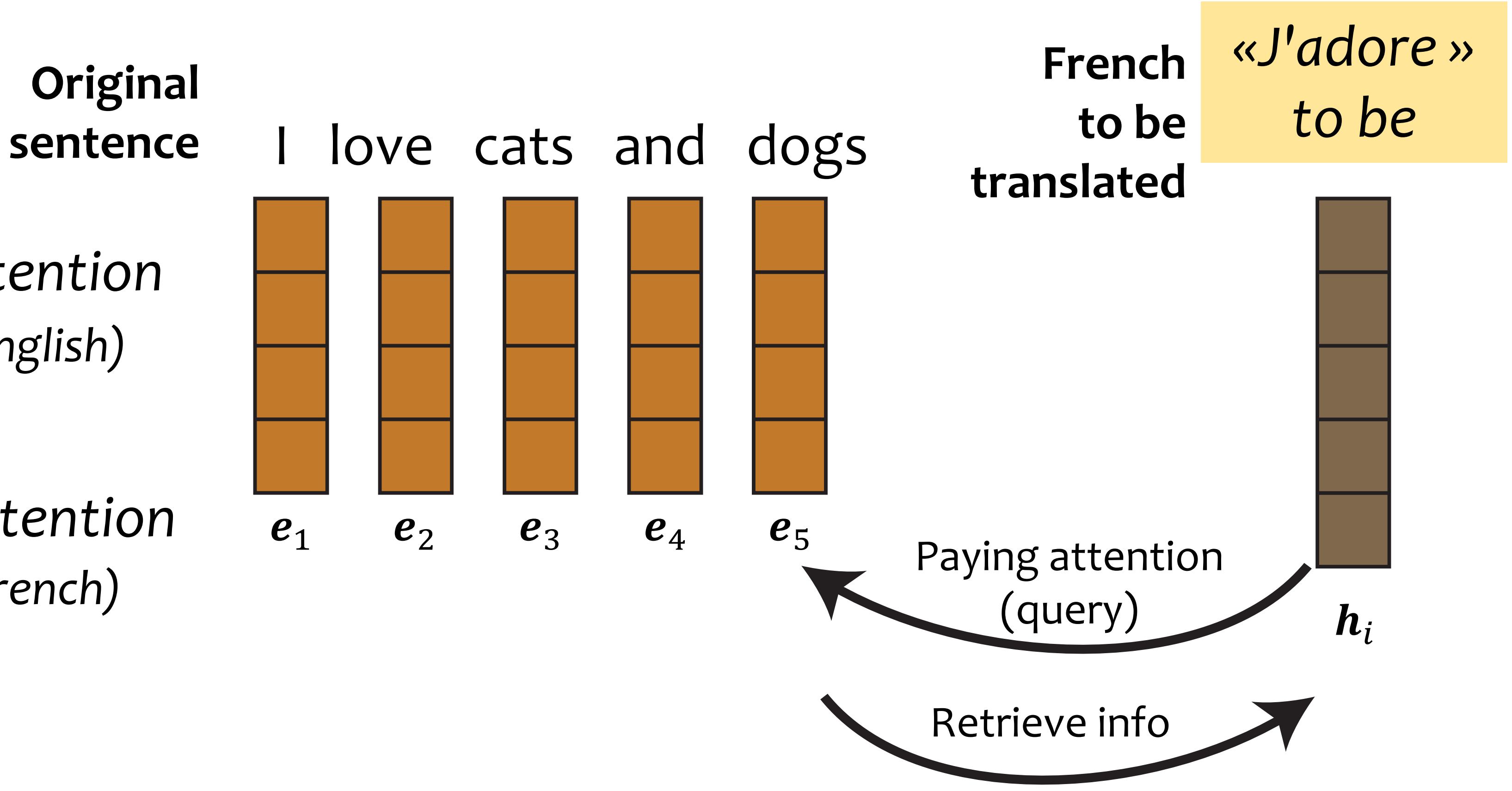
- Soft indexing
  - $\boldsymbol{a}$ as a distribution over the keys, represents how much information I want from this key.
  - Matrix vector product, weighted combines the values.
- How to compute a?
  - *Feature based attention:* based on similarity of query and key.



Set up

- $\boldsymbol{e}_j$ *vector of target of attention*
  - *(encoder hidden state, English)*

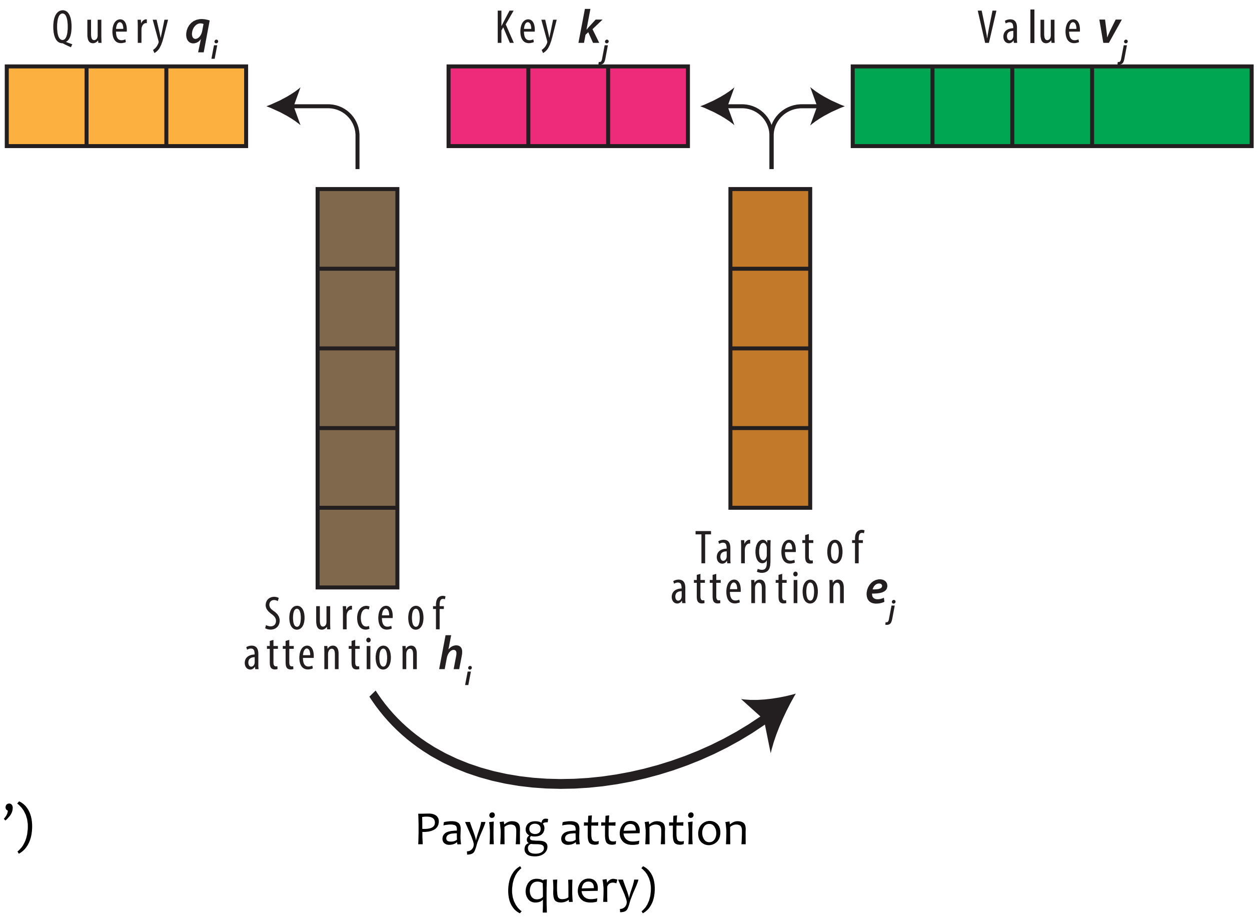
- $\boldsymbol{h}_i$ *vector of source of attention*
  - *(decoder hidden state, French)*



QKV attention

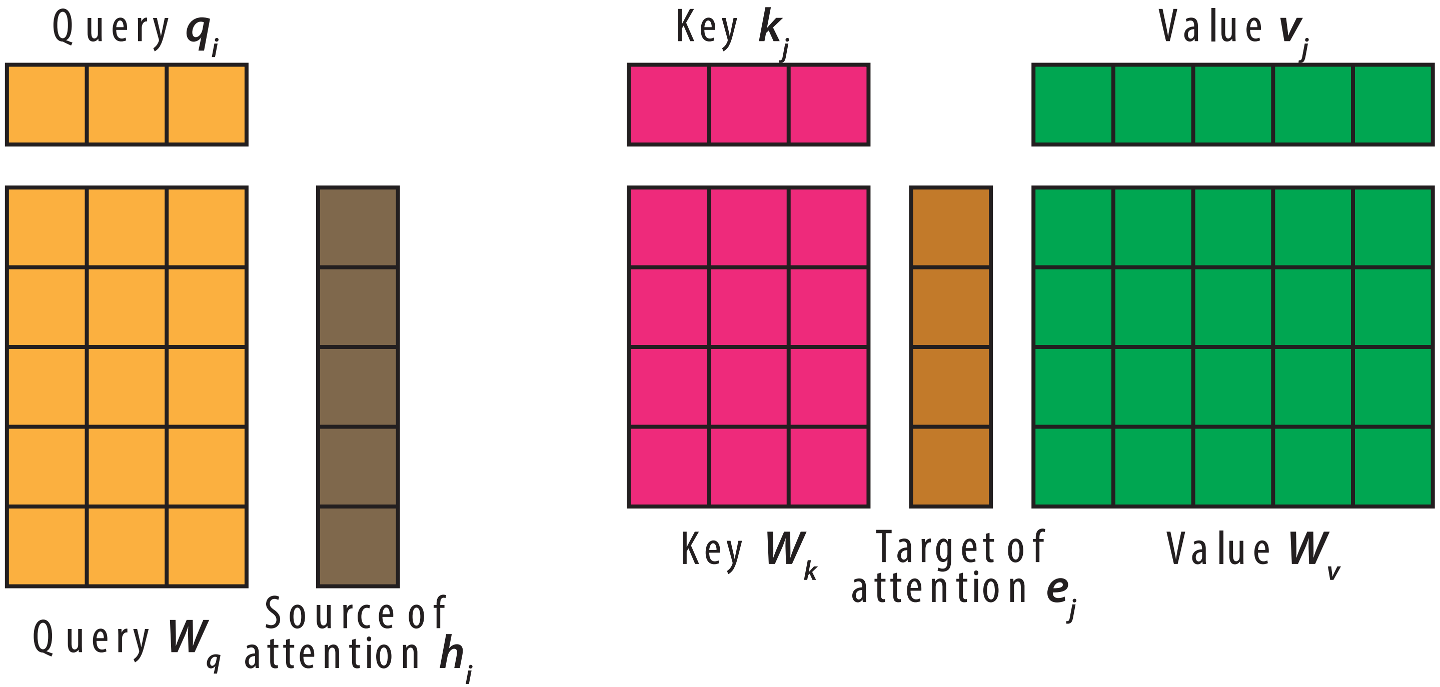
- $\boldsymbol{h}_i$ *vector of source of attention*
  - *(decoder hidden state, French)*
- $\boldsymbol{e}_j$ *vector of target of attention*
  - *(encoder hidden state, English)*

- **Query** : what source need
  - (*J'adore* : "I want subject pronoun & verb")
- **Key** : what the target provide
  - (*I* : "Here is the subject")
- **Value** : the information to be retrieved
  - (information related to *Je* or *J'*)



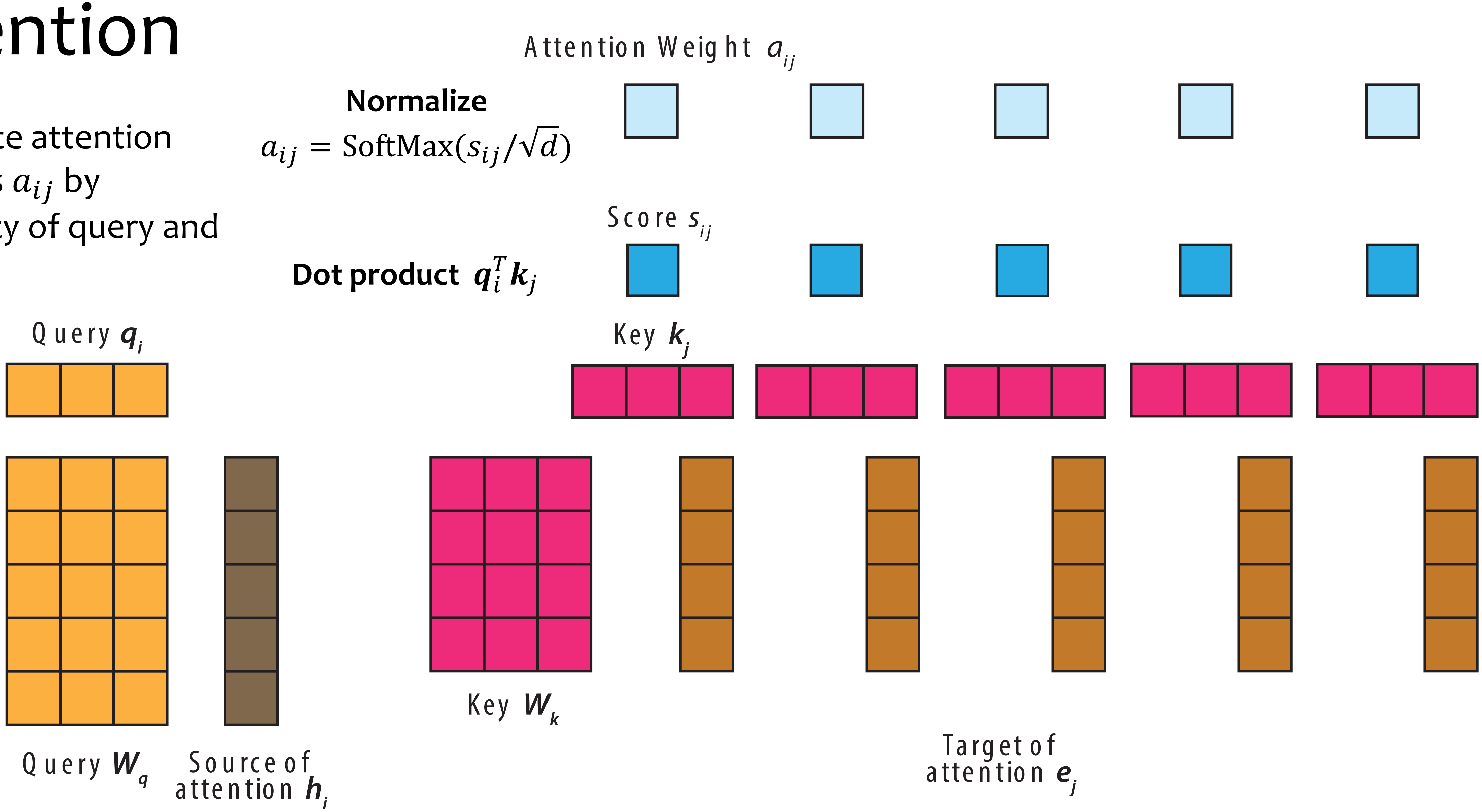
QKV attention

- QKV are linear projections of “word vector”
  - Query $q_i = W_q h_i$
  - Key $k_j = W_k e_j$
  - Value $v_j = W_v e_j$



QKV attention

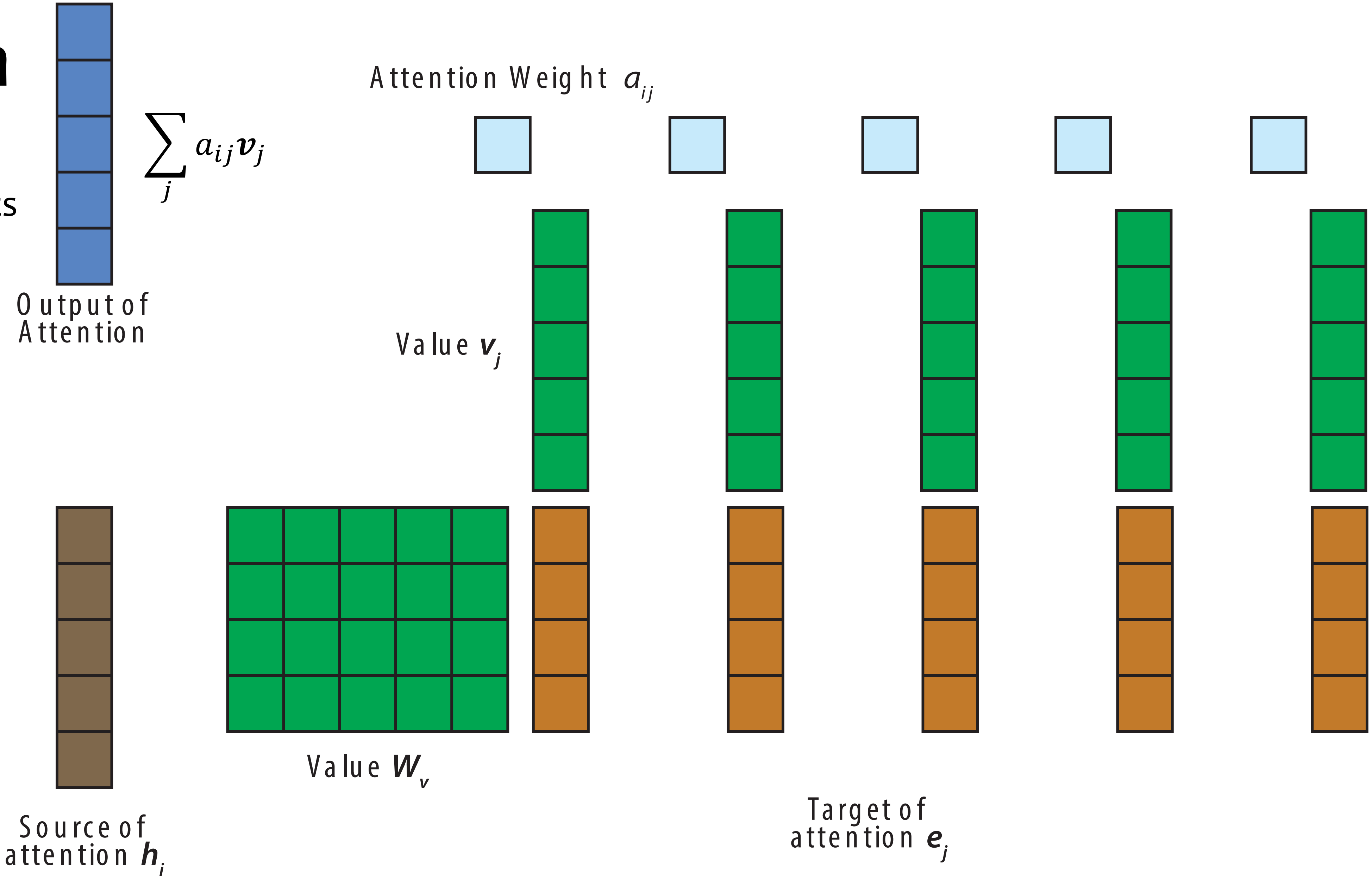
- Compute attention weights a_{ij} by similarity of query and key


Attention Weight a_{ij}
Normalize
$a_{ij} = \mathrm{SoftMax}(s_{ij}/\sqrt{d})$
Score s_{ij}
Dot product $\boldsymbol{q}_i^T \boldsymbol{k}_j$
Query $\boldsymbol{q}_i$
Key $\boldsymbol{k}_j$
Query $\boldsymbol{W}_q$
Source of attention $\boldsymbol{h}_i$
Key $\boldsymbol{W}_k$
Target of attention $\boldsymbol{e}_j$

QKV attention

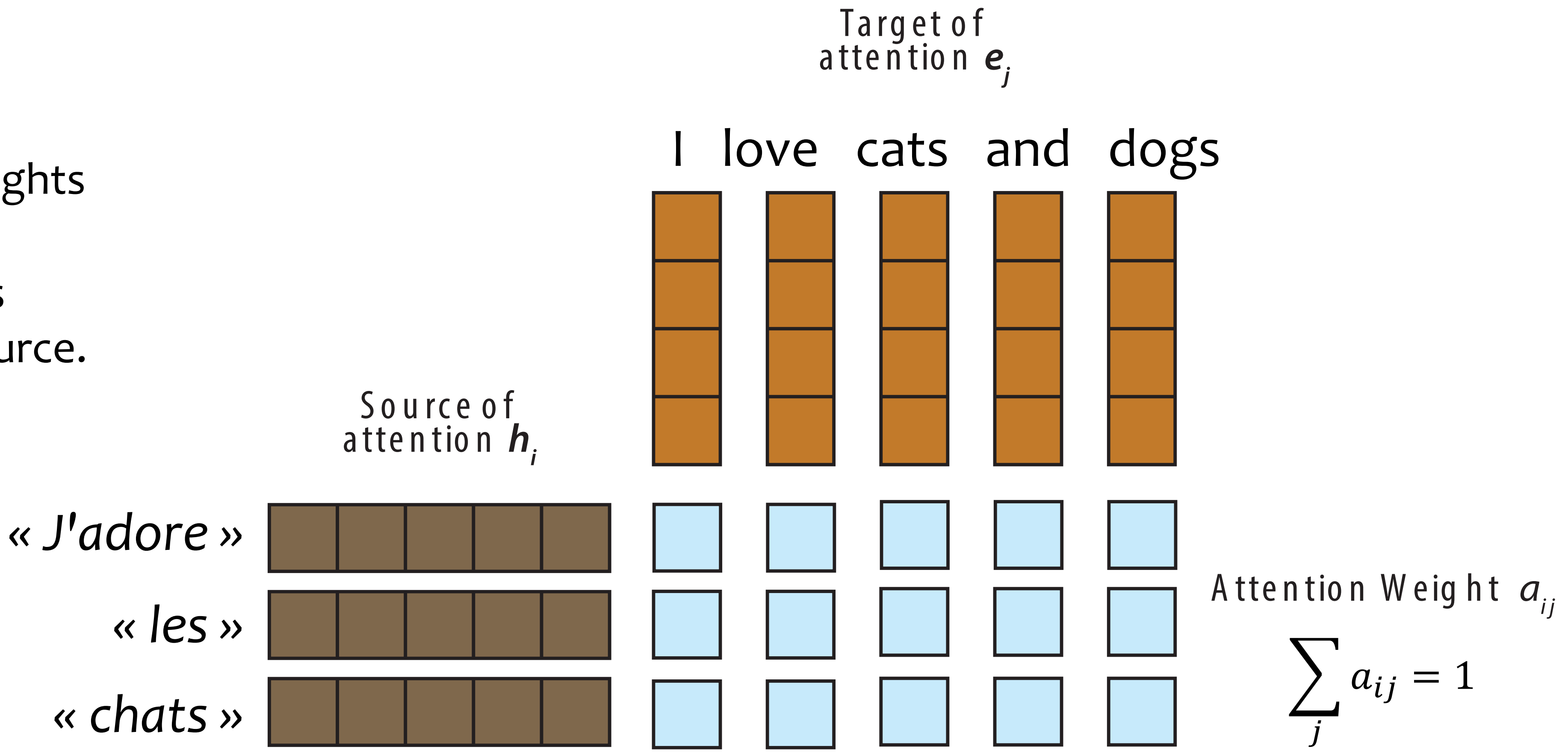
- Use attention weights a_{ij} to weighted sum values $\boldsymbol{v}_j$

- The output of attention should contain more useful information, and it's added to the original $\boldsymbol{h}_i$

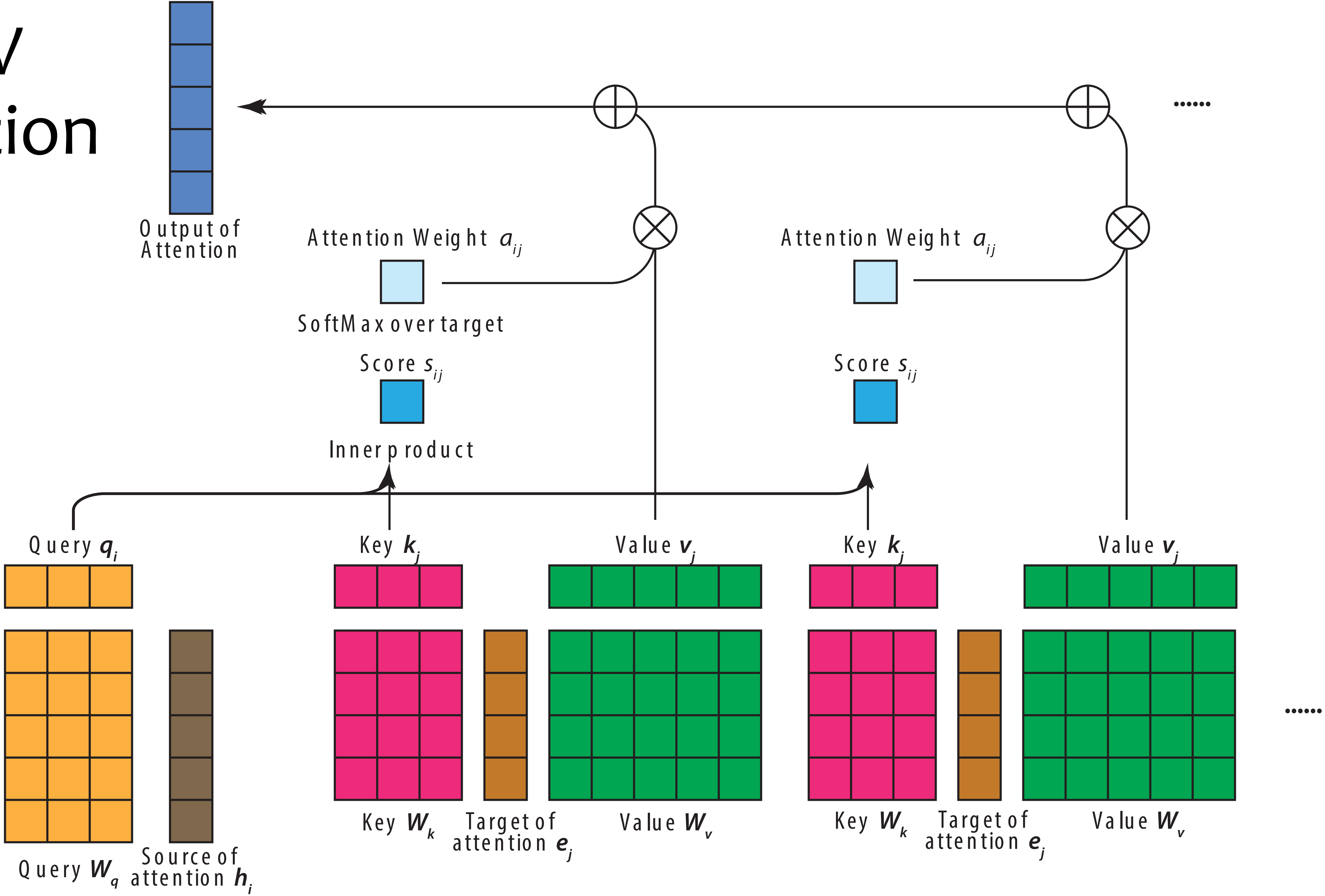

Attention Weight a_{ij}
$\sum_j a_{ij}\boldsymbol{v}_j$
Output of Attention
Value $\boldsymbol{v}_j$
Value $\boldsymbol{W}_v$
Source of attention $\boldsymbol{h}_i$
Target of attention $\boldsymbol{e}_j$

Attention matrix

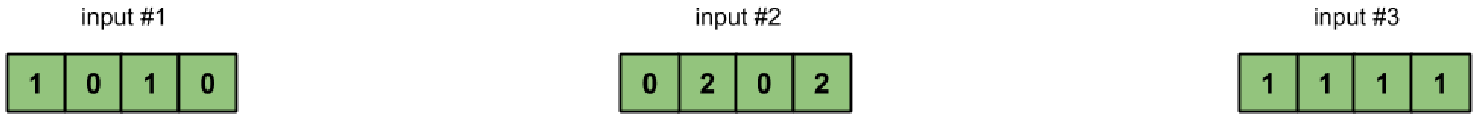
- The attention weights a_{ij} describes the information flows from target to source.


Target of attention $\boldsymbol{e}_j$
I
love
cats
and
dogs
Source of attention $\boldsymbol{h}_i$
« J'adore »
« les »
« chats »
Attention Weight a_{ij}
$\sum_j a_{ij} = 1$

QKV attention



Self-attention



input #1

1	0	1	0

input #2

0	2	0	2

input #3

1	1	1	1

Summary: Attention mechanism

- Compute linear projection $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$
- Compute the inner product (similarity) of key $\boldsymbol{k}$ and query $\boldsymbol{q}$
- SoftMax the normalized score as attention distribution.

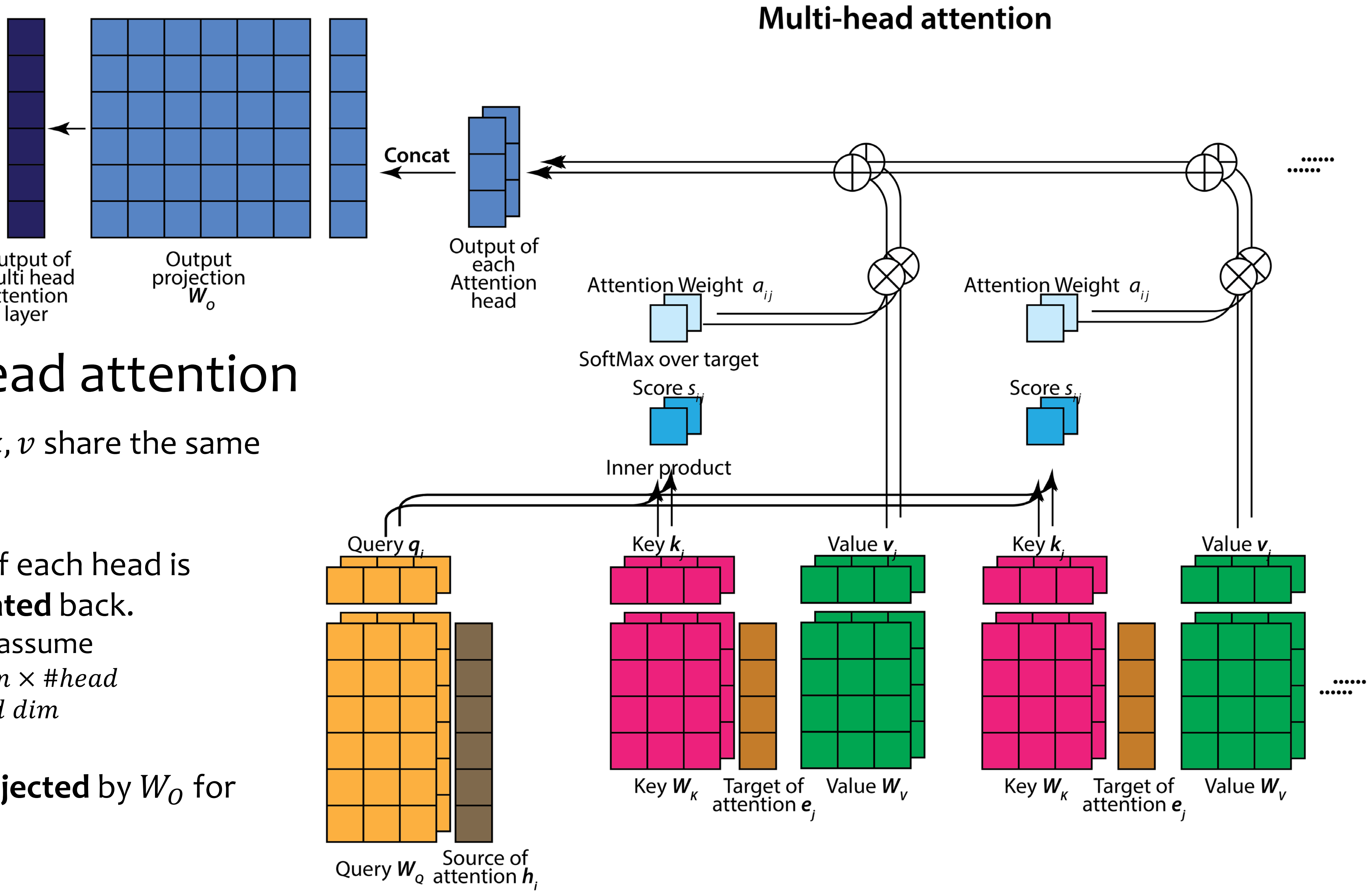
$$a_{ij} = \mathrm{SoftMax}\left(\frac{\boldsymbol{k}_j^T \boldsymbol{q}_i}{\sqrt{len(\boldsymbol{q}_i)}}\right), \sum_j a_{ij} = 1$$

- Use attention distribution to weighted average values $\boldsymbol{v}$.

$$\boldsymbol{z}_i = \sum_j a_{ij} \boldsymbol{v}_j$$

- Note:
  - Learnable weights: linear matrix W_q, W_k, W_v

Aka, "Scaled dot product attention"



Multi-head attention

- Often, q, k, v share the same $head\ dim$
- Outputs of each head is **concatenated** back.
  - Often assume $head\ dim \times \#head = embed\ dim$
- Linear **projected** by W_O for mixing.

Summary: Multi-head attention

- Each head performs a scaled dot-product attention.
- Output of each attention head $\boldsymbol{z}_i$ (weighted sum of value) is concatenated and then linear projected.

- Learnable parameter
  - W_Q, W_K, W_V
  - An output projection matrix W_O (maybe bias b_O)

Note for setting the dimension

- General principle:
  - Vectorized operation is much faster than loops.
  - When weight matrices have the same shape, it's easier to batch them together and vectorize matrix multiplication.

- Some common practice / norm
  - $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ vectors share the same $head\ dim$.

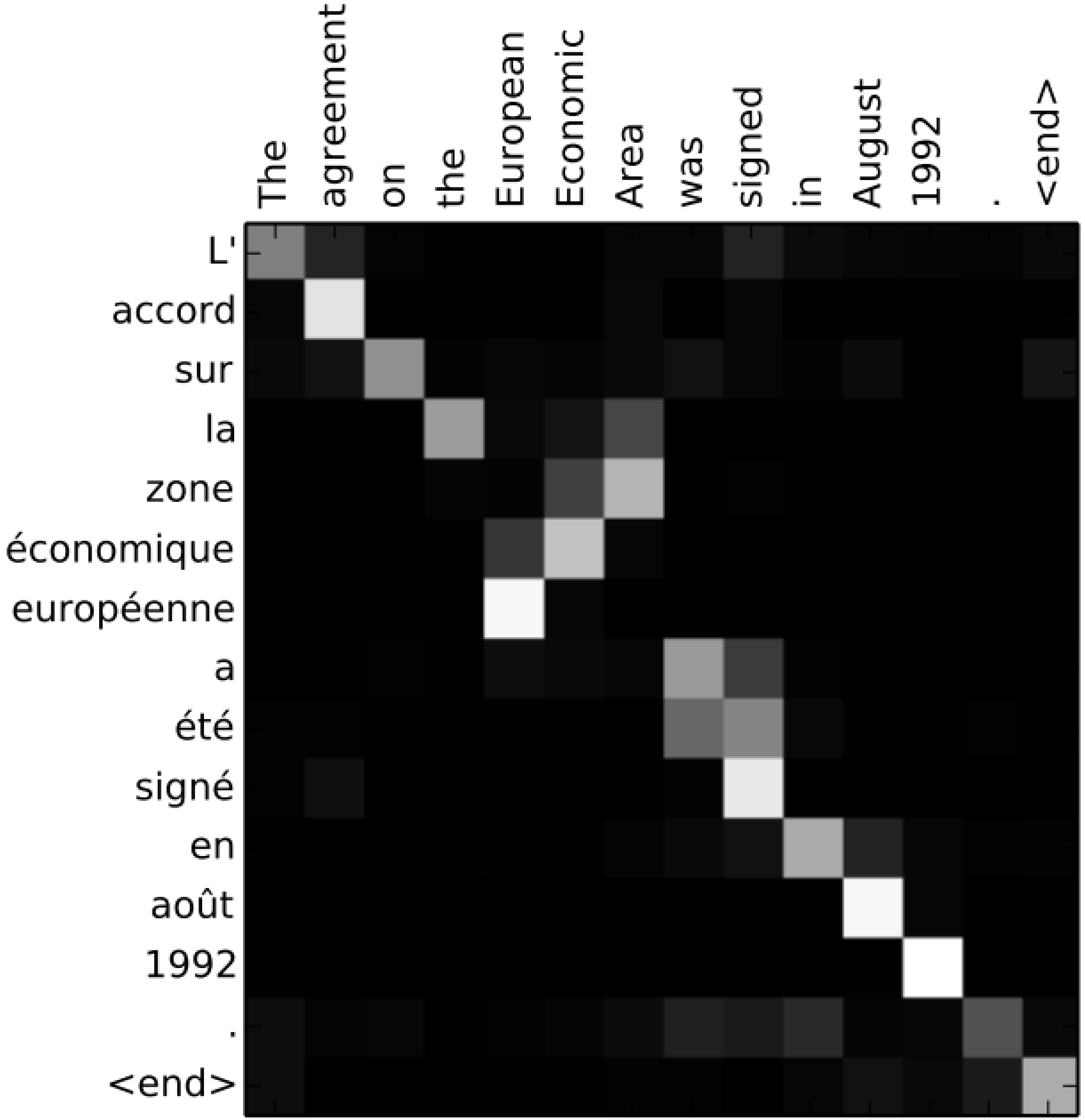
  - When computing $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ all the heads are concat in one axis, then splitted.
    - So, W_Q, W_K, W_V have the same shape $[embed\ dim, head\ dim \times \#head\ cnt]$.

  - Mandate $head\ dim \times \#head\ cnt = embed\ dim$.
    - ⇒ Concat head output has the same shape as the embedding
    - ⇒ input and output of multi-head attention are the same shape.

  - By such, usually W_Q, W_K, W_V, W_O are square matrices of shape $[embed\ dim, embed\ dim]$.

Visualizing attention matrix a_{ij} after training

- *Learning to pay Attention*
- *French 2 English*
    - *"la zone economique europeenne" →*
      "the European Economic Area"
    - *"a ete signe" →*
      "was signed"

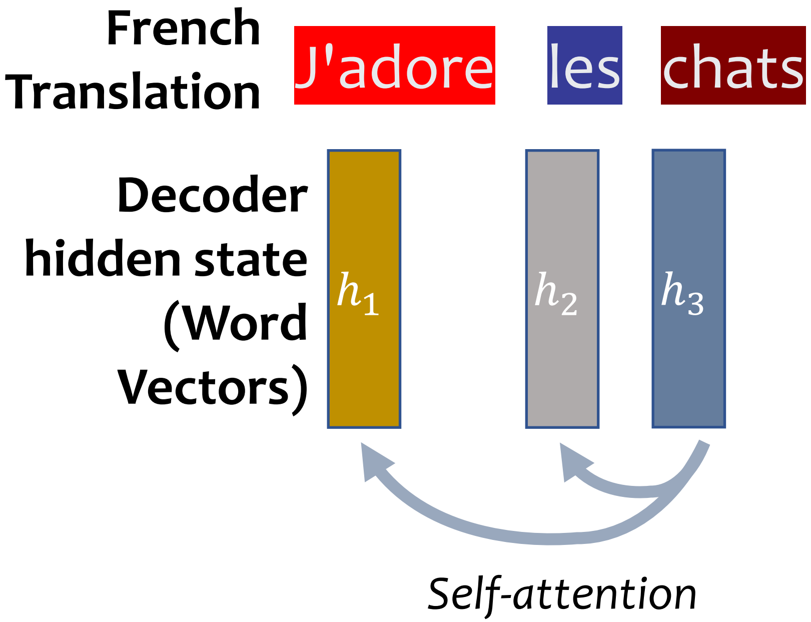


https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Cross vs Self Attention

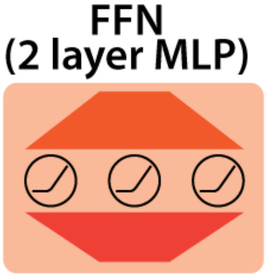
- Cross Attention
  - Source and target are different bunch of vectors (can have different length, number)
  - E.g. words in one language pay attention to words in **another**.

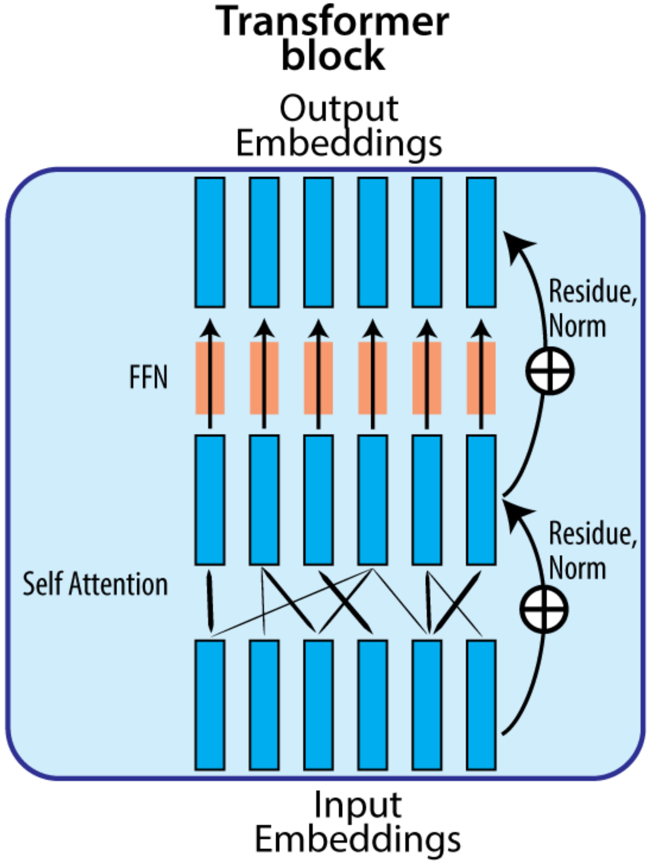
- Self Attention ($e_i = h_i$)
  - Source and target are the same bunch of vectors, with same length, number.



From Attention to **TransformerBlock**

- A **transformer block** has
  - Self Attention
    - information exchange ***between tokens***
  - Feed forward network
    - Information transform ***within tokens***
    - E.g. a multi-layer perceptron with 1 hidden layer
    - GeLU activation is commonly used.
  - Normalization (Layer normalization)
- A transformer model contains N x **transformer block**



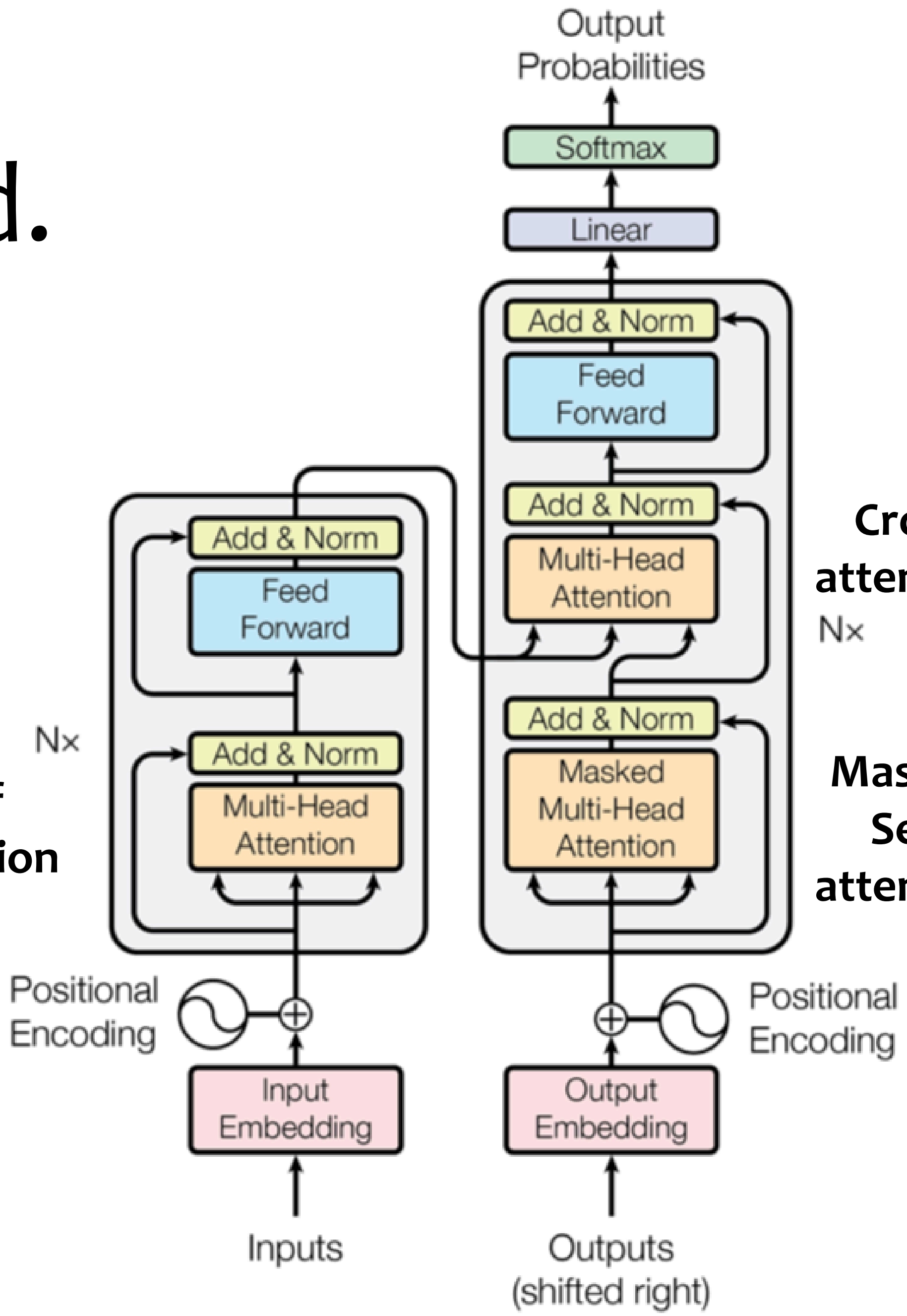


Transformer is all you need.

- **Position encoding**
  - Sinusoidal function
  - Learnt from scratch (**position embedding**)

- **Encoder:** process input in parallel
  - Self Attention + FFN

- **Decoder:** generate output autoregressively
  - Masked Self Attention
  - Cross attention (fetch into)
  - FFN



Self attention

Cross attention

Masked Self attention

Figure 1: The Transformer - model architecture.

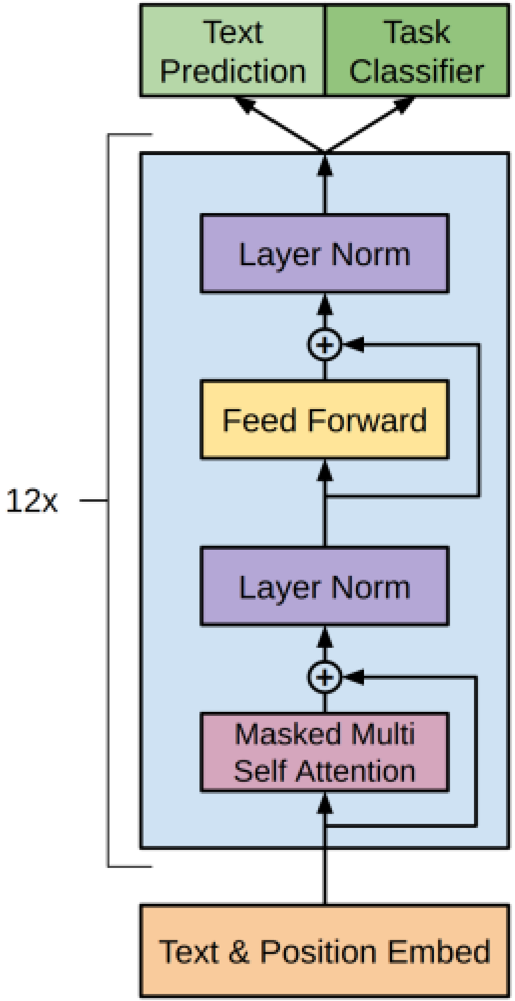
https://arxiv.org/abs/1706.03762

GPT architecture

- Learned text & position embedding

- Transformer block with causal masked self-attention

- *Optional cross attention* module for adding conditional info.



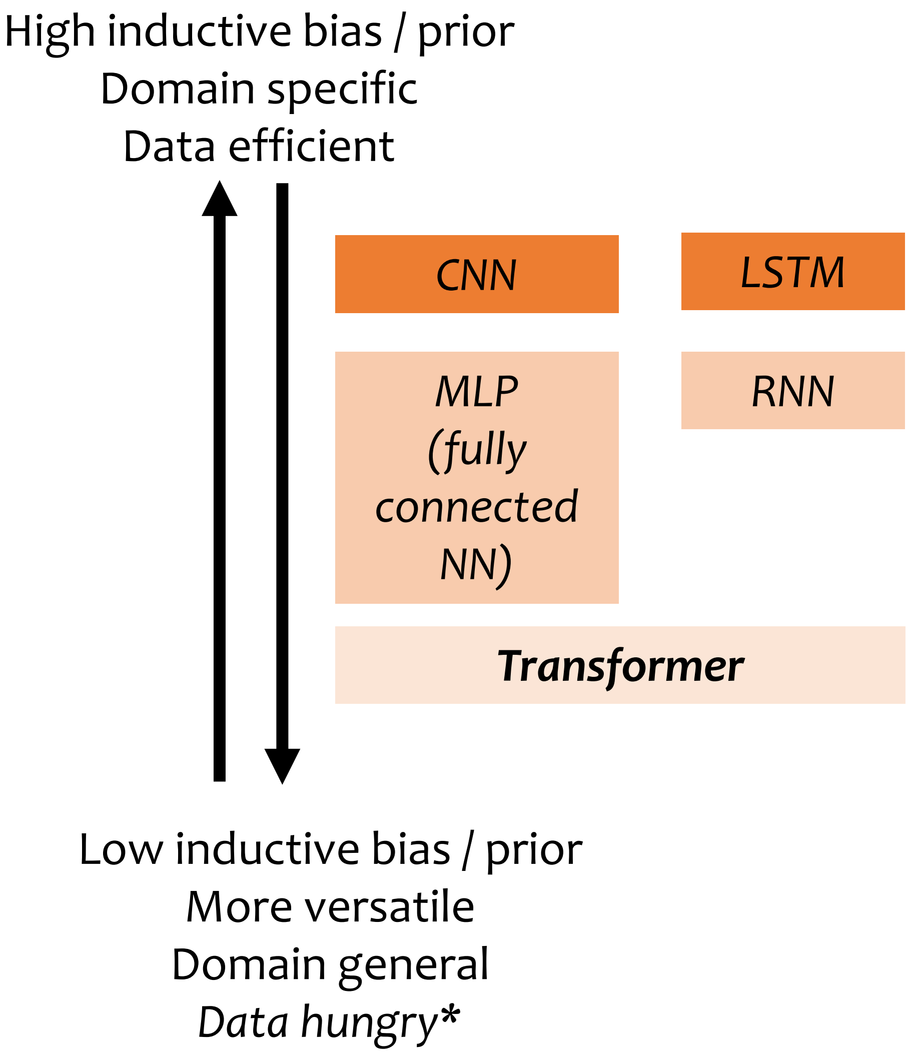
GPT1
Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I., 2018. Improving language understanding by generative pre-training.

Interim Summary: Attention

- Attention allows **dynamic routing** of information.
  - **Cross attention** : Fetching *relevant information* from one domain to the other
  - **Self attention :** Mixing information in one domain
- ***TransformerBlock*** = Composition of Self / Cross-Attention + MLP + Normalization
- Transformer = input embeddings + sequence of ***TransformerBlocks***

Remark on Transformer Architecture

- Transformer is extremely versatile.
  - potentially more general than MLP, weights are *dynamically generated*.
- Low **inductive bias**
  - Can learn everything from data
  - At large data limit, low bias is better than man-made bias.
  - On the flip side, it can suffer at small dataset.



High inductive bias / prior
Domain specific
Data efficient

CNN

LSTM

MLP (fully connected NN)

RNN

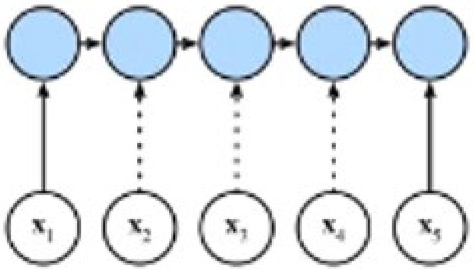
Transformer

Low inductive bias / prior
More versatile
Domain general
*Data hungry**

Remark on Transformer vs RNN

RNN/LSTM

- **Sequential:** Time complexity $\sim T$
- Hidden states may *forget* about past words
- Space complexity $O(1)$
  - Scale to long sequence trivially



Transformer

- **Parallel:** Time complexity ~ 1
- Constant access to all the past / words
- Space complexity $\sim O(T^2)$
  - Limit by memory & position embedding when scale to long sequence

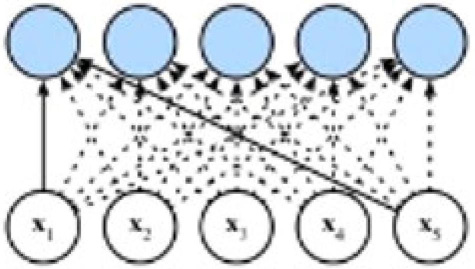


Figure from Neuromatch 2022

Part III
Training language models

Given these models, how to train them?

Learning Word Vectors via *Language Modelling*

BERT, GPT & CLIP

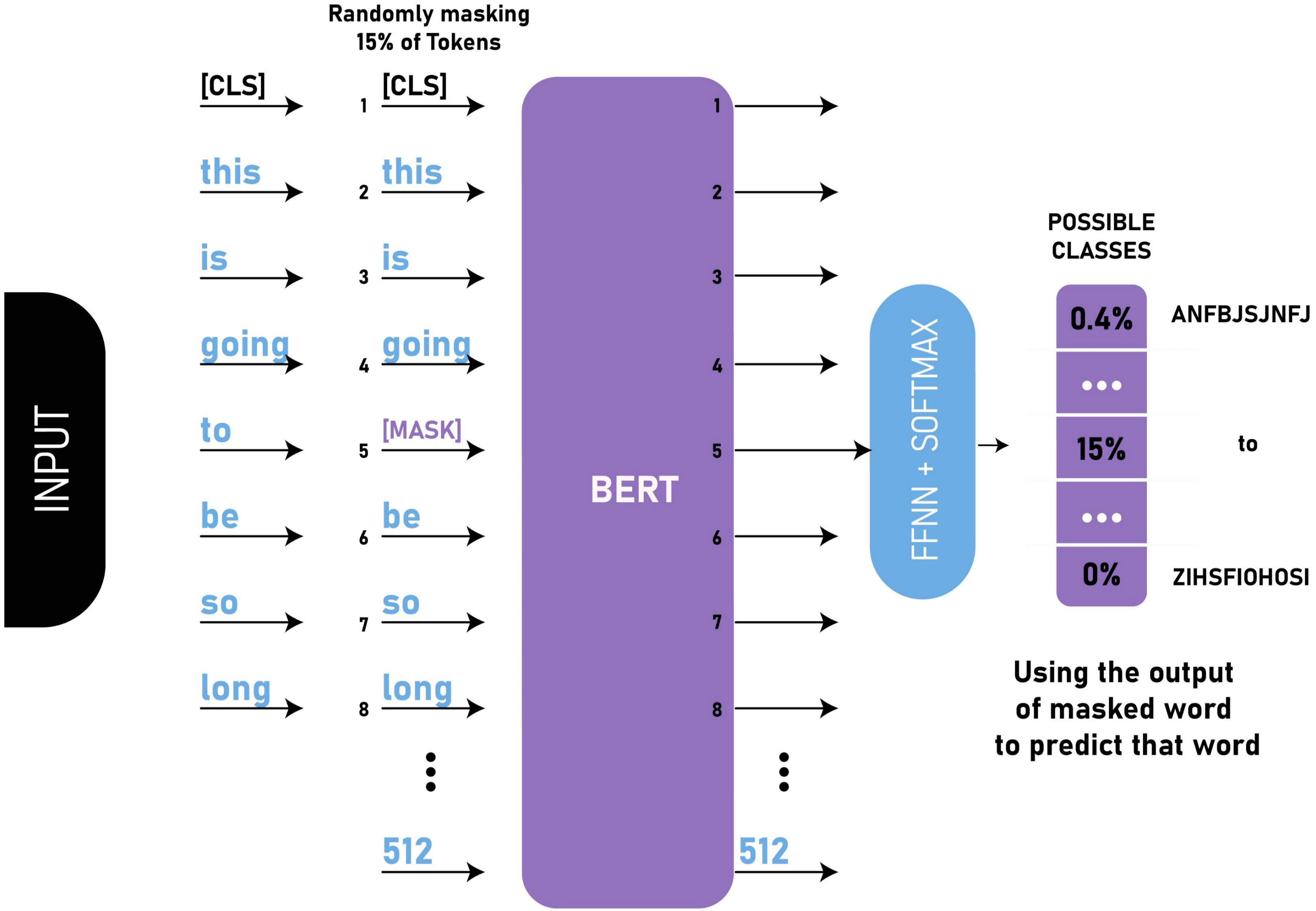
- *Self-supervised learning* of word representation
  - **Autoregressive language modelling:** predict next word (GPT)
  - **Masked language modelling:** predict missing words (BERT)
  - **Multimodal Contrastive Learning:** matching image and text. (CLIP)

MLM — Sentence-Transformers documentation (sbert.net)

Masked word prediction / Cloze task:

BERT = Bidirectional Encoder Representations from Transformers

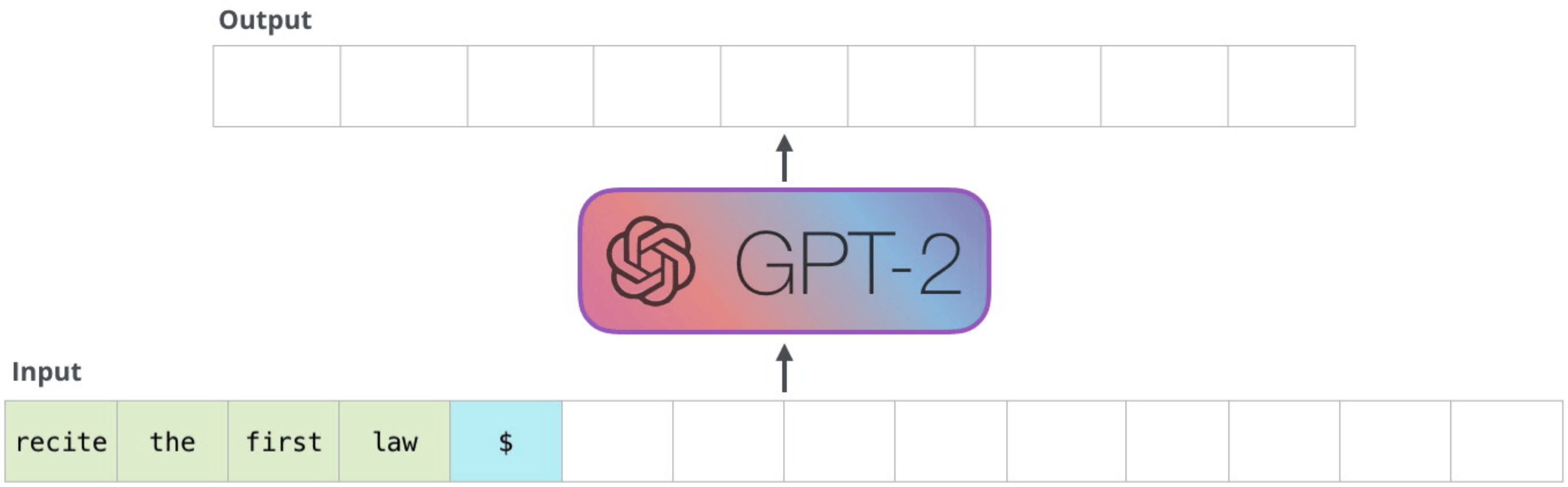
- Mask out 15% tokens and predict those [MASK].
- **Bidirectional**: Predict the mask token from context on both sides .
- Transformer **Encoder,** all-to-all self-attention
- Suitable for text **representation**



Next word prediction:
GPT = Generative Pre-trained Transformer

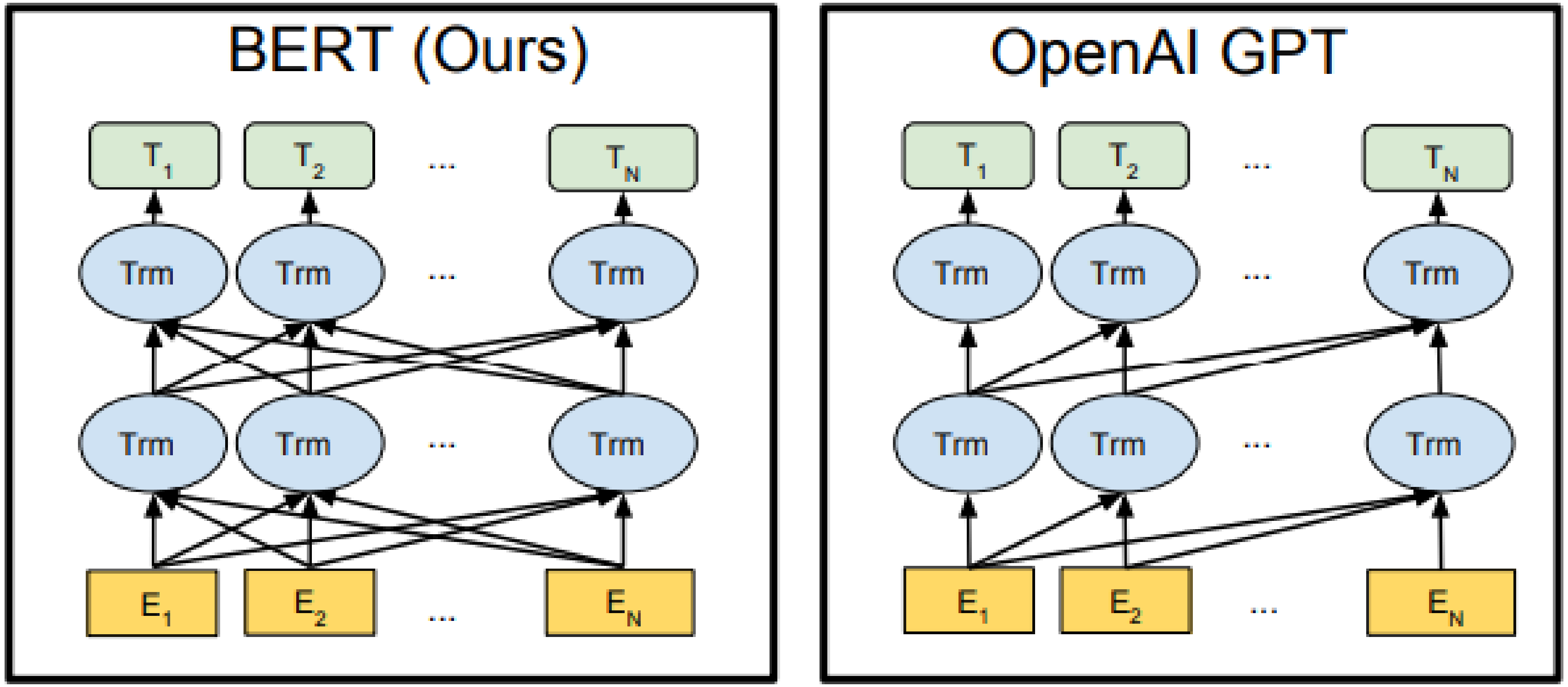
- **Pre-trained:** Each word predict the next word considering all previous words
  - Enforcing causal attention.
  - Enabling text generation

- **Generative:** Autoregressive model, suitable for text generation.



The Illustrated GPT-2

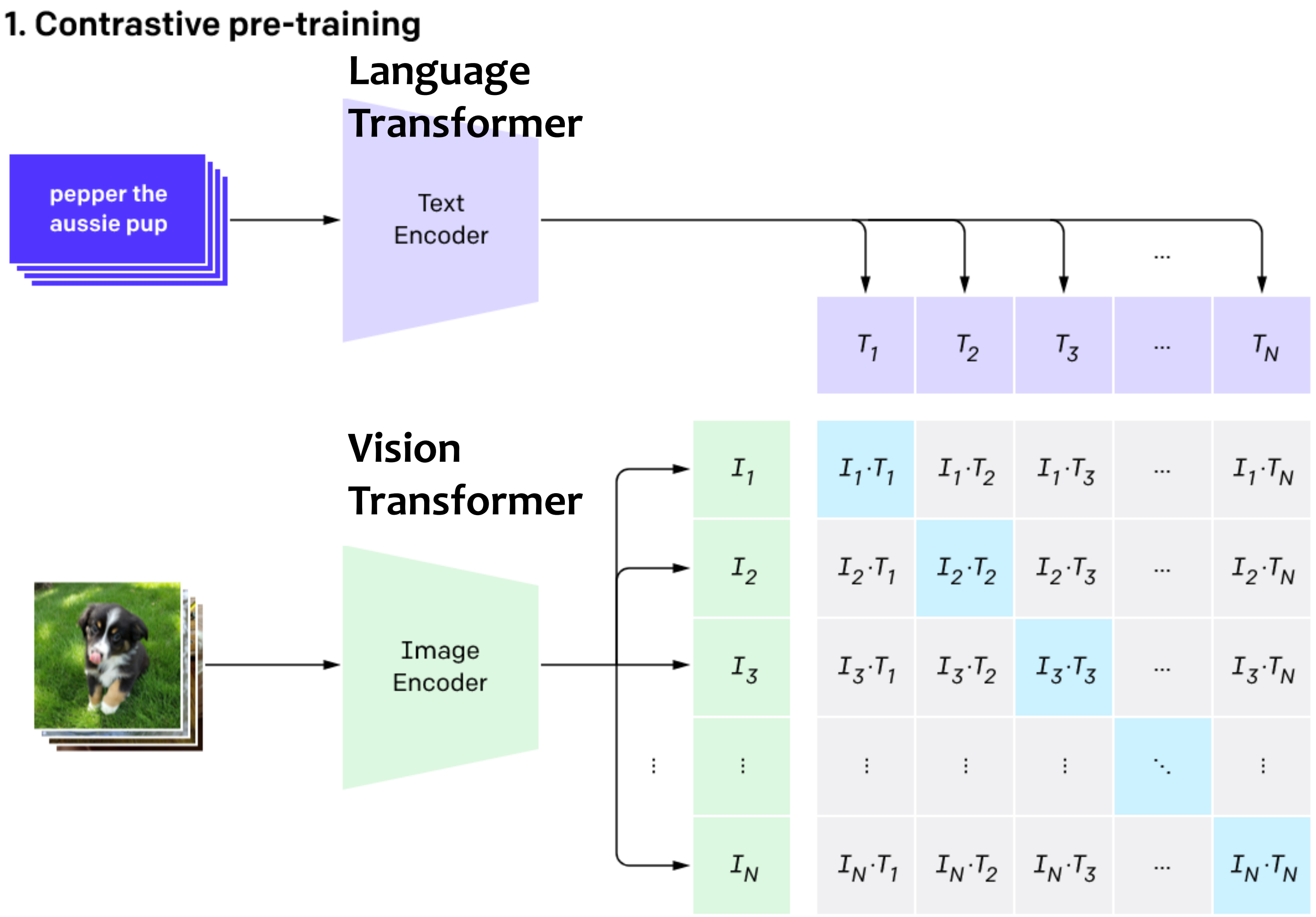
Comparing BERT and GPT



BERT Paper

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Multimodal representation learning:
CLIP = Contrastive Language-Image Pre-Training


1. Contrastive pre-training
Language Transformer
pepper the aussie pup
Text Encoder
Vision Transformer
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
⋮ ⋮ ⋮ ⋱ ⋮
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

- Learn a joint encoding space for text caption and image
- Contrastive learning
  - Maximize representation similarity between paired image and caption.
  - Minimize representation similarity of other pairs

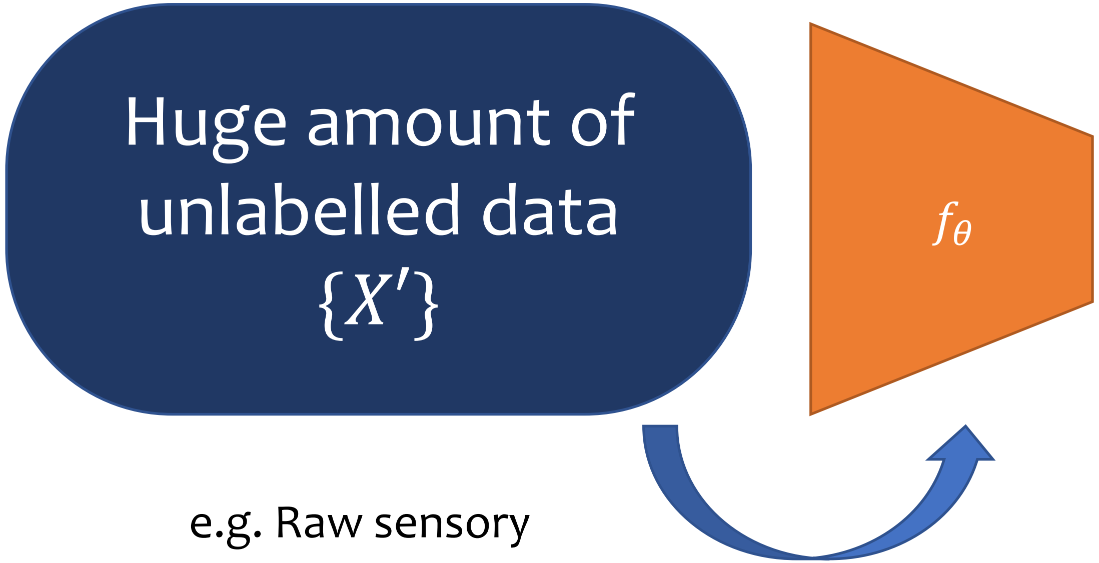
https://github.com/openai/CLIP

Beyond Supervised Learning

How to learn without labelling, generally

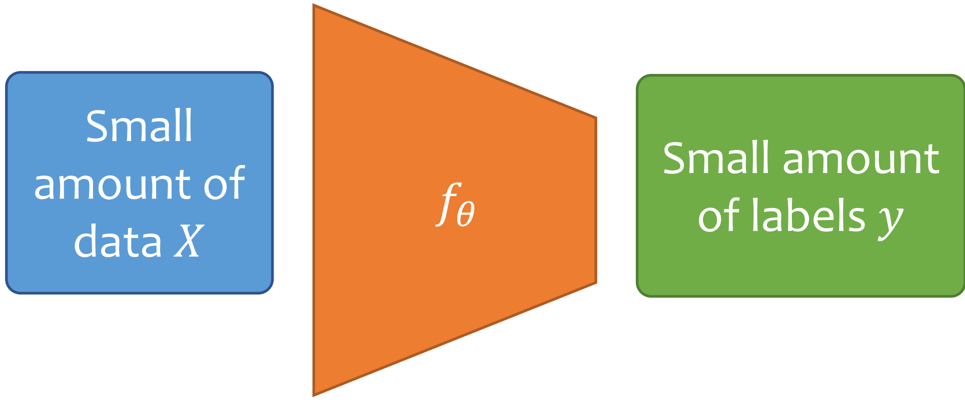
Self-supervised learning paradigm

Self-supervised pre-training



e.g. Raw sensory experience

Supervised fine-tuning

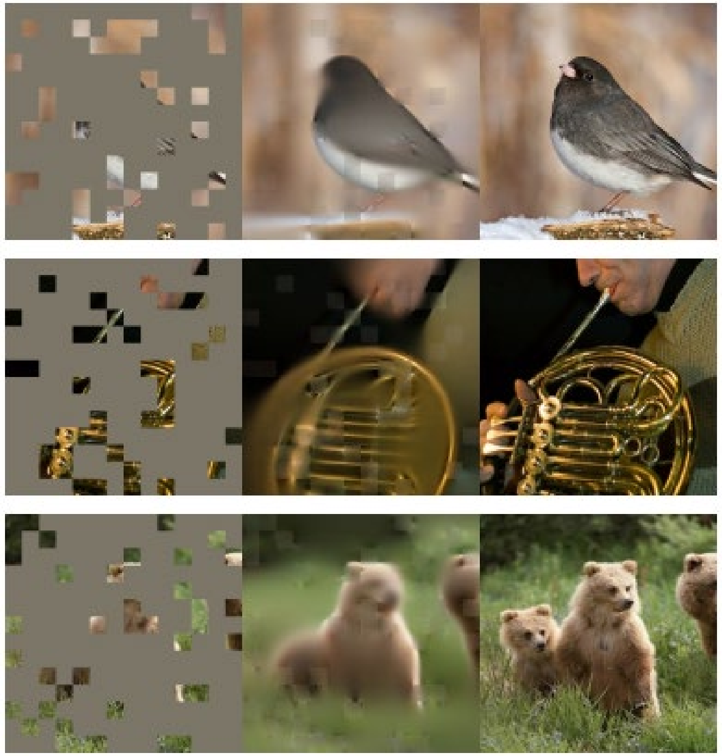


e.g. Semantic label

- Objective of pretraining
  - Learn ***good representation*** , *s.t.*
    - downstream tasks can be easily solved on these representations. (e.g. linear decodable)
    - with fewer labeled samples

No supervision?
Create supervised tasks and solve them.

- Self-supervised learning
  - Mask out some part of data and learning by predicting
  - Learning by comparing similarity.



Self-prediction
(Masked Autoencoder, BERT, GPT)

Contrastive learning
(SimCLR, CLIP)

Part IV
Using language models after pretraining

Finetuning , Prompting, Head, Adapter

Two expectations for AI

Machine Translation

Code completion

Song writing

Email writing

Text summary

Chat bot

Doing math

……

Specialist

- Pro:
  - Solve one task far better than anyone.
- Con:
  - Perform poorly on any other tasks …

Generalist

- Pro
  - Flexibly solve many tasks with little modifications.
- Con
  - May not perform as well as specialist in one task.

Hungyi-Lee https://www.youtube.com/watch?v=F58vJcGgjt0&t=4s

Two major ways of using Language Models

Fine-tuning

- Gradient descent on weights to optimize performance on *one task.*

- What to fine-tune?
  - *Full network*
  - *Readout heads*
  - *Adapters*

Parameter efficient fine-tuning

Change the model "itself"

Prompting

- Design special prompt to cue / condition the network into specific mode to solve any tasks.

- No parameter change. one model to rule them all.

Change the way to use it.

Hungyi-Lee
https://www.youtube.com/watch?v=F58vJcGgjt0&t=4s

A historical note on Finetuning vs Prompting

Fine-tuning

- Popularized on **BERT**
  - Provide representation for solving downstream tasks.
  - Smaller scale
  - Cannot "*speak*"

Prompting

- Popularized on **GPT3/4**
  - Can speak.
  - Can be prompted into solving tasks.

Change the model itself

Change the way to use it.

Hungyi-Lee
https://www.youtube.com/watch?v=F58vJcGgjto&t=4s

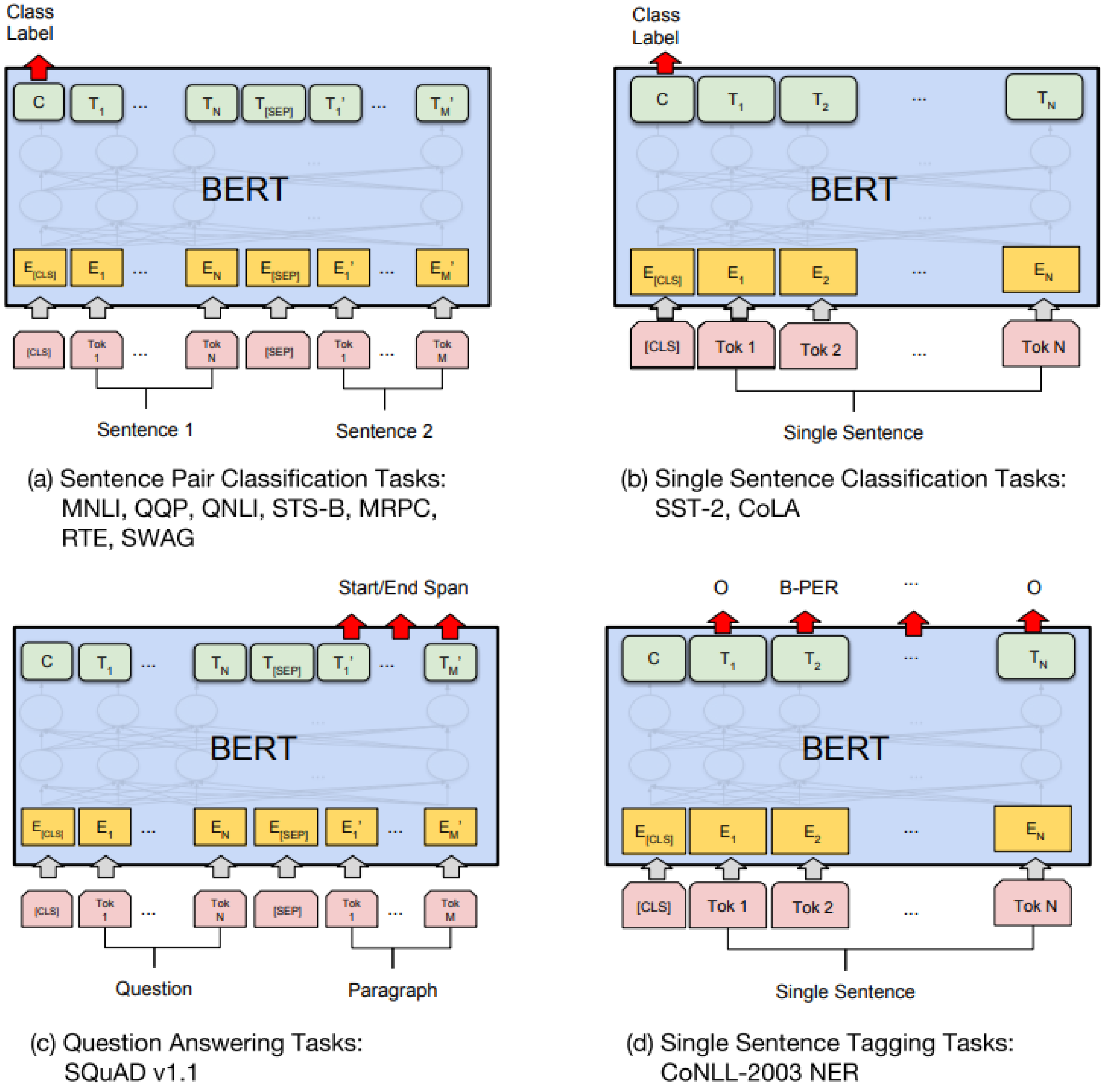
“Finetuned” ironman suits

Large Foundation model

Finetuned models for special purposes

Finetuning
Readout Heads

- BERT as example.
    - Classify sentence / pair
    - Q&A: Find the relevant text span in a paragraph.



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

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

(c) Question Answering Tasks: SQuAD v1.1

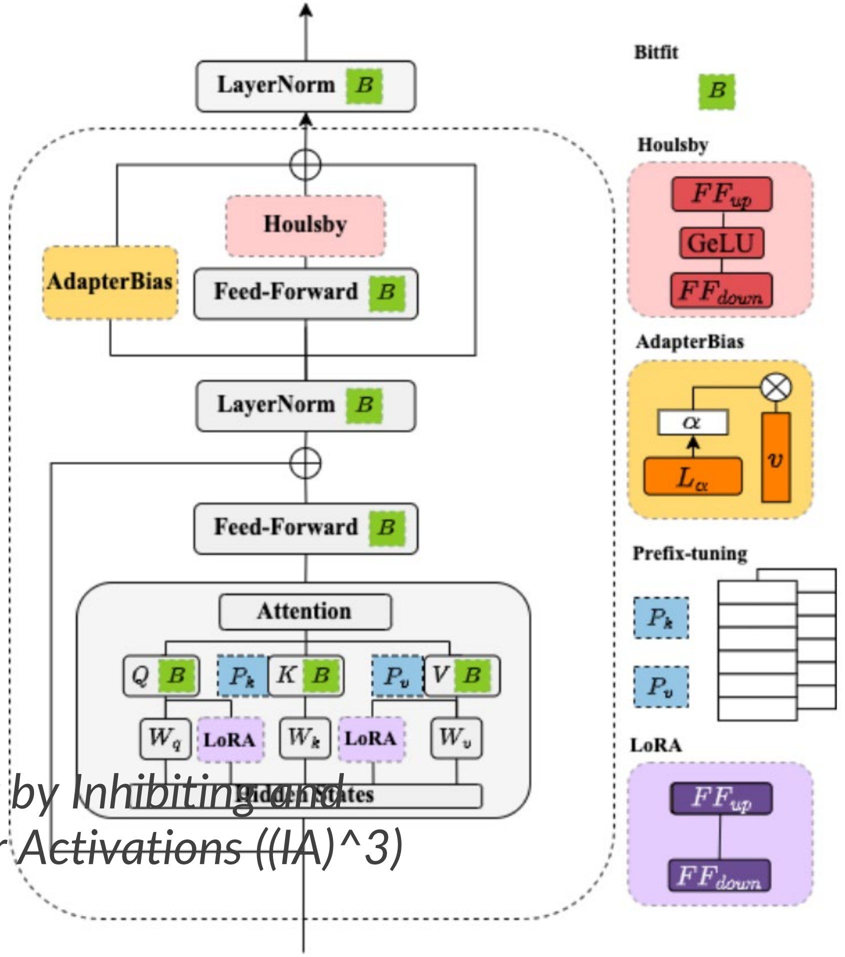
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 4: Illustrations of Fine-tuning BERT on Different Tasks.

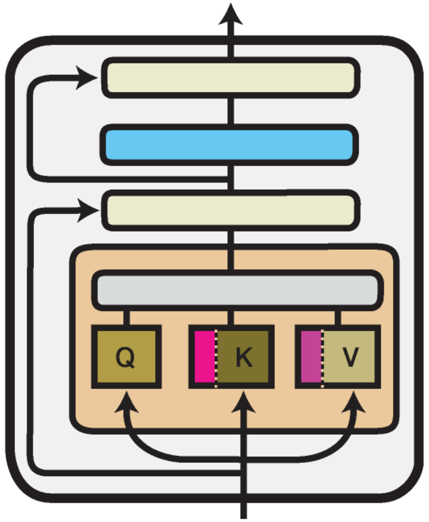
Finetuning Adapter

Light weight addon for LLM

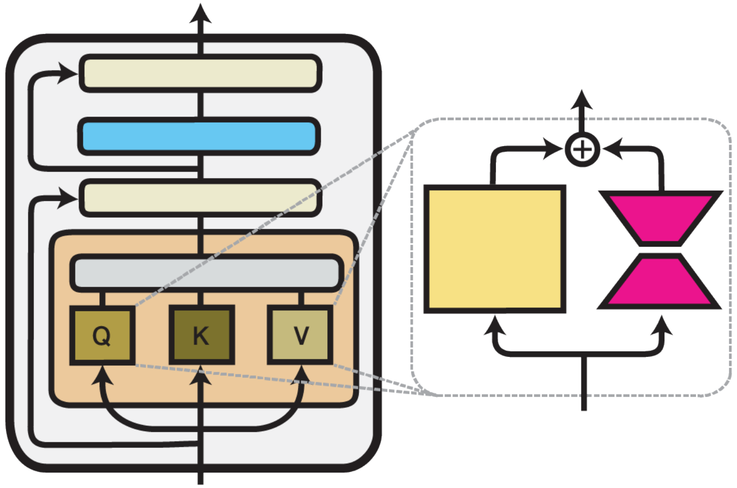
- *Insert tiny modules* into LLM and only finetune those parameters.
- Easier to save.



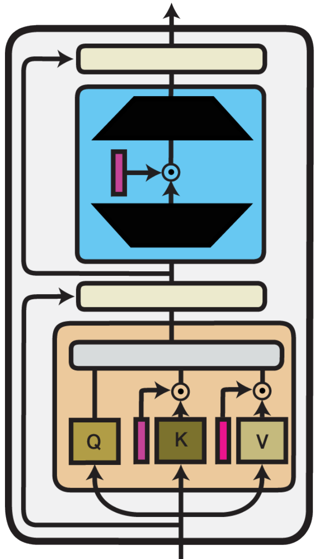
Prefix Tuning



Low-Rank Adaptation (LoRA)



Infused Adapter by Inhibiting and Amplifying Inner Activations ((IA)^3)



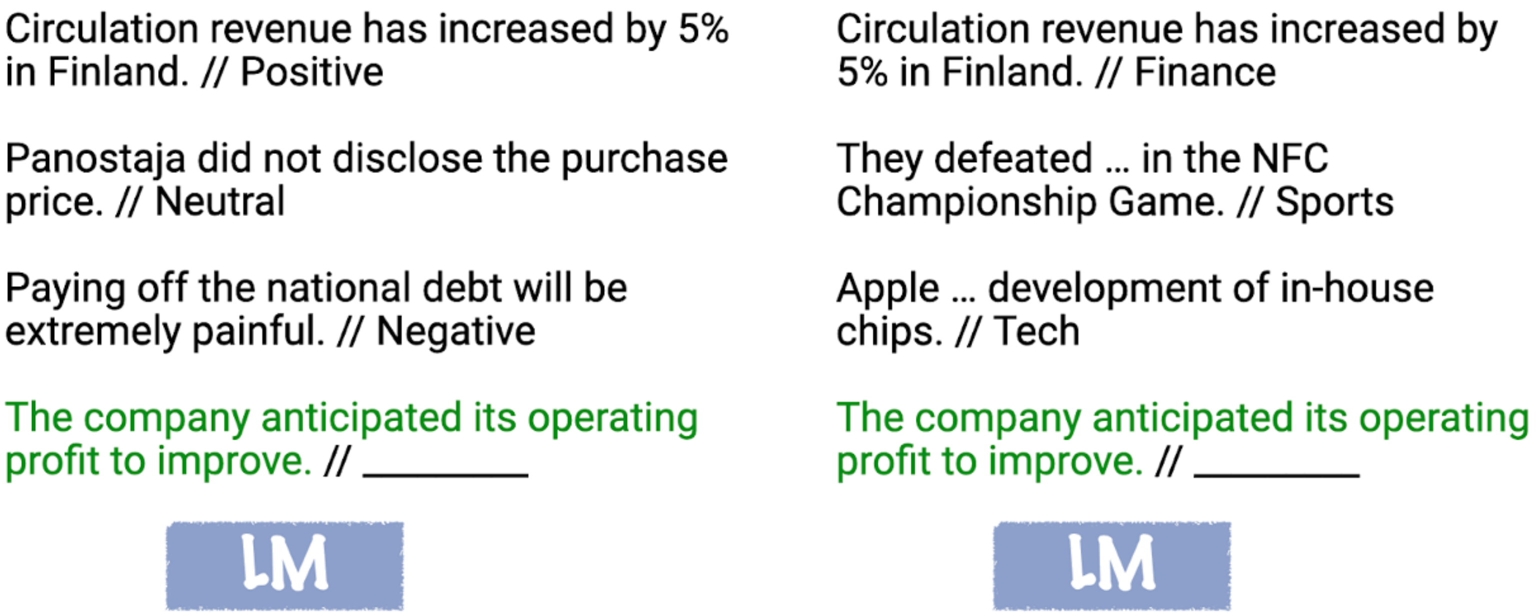
https://arxiv.org/abs/2210.06175

Adapter Methods — adapter-transformers documentation

Prompting:

let the model **learn** a task **in context.**

- Give some examples as demonstration and let the model follow it to solve new problems.



Circulation revenue has increased by 5% in Finland. // Positive

Panostaja did not disclose the purchase price. // Neutral

Paying off the national debt will be extremely painful. // Negative

The company anticipated its operating profit to improve. // ________

Circulation revenue has increased by 5% in Finland. // Finance

They defeated … in the NFC Championship Game. // Sports

Apple … development of in-house chips. // Tech

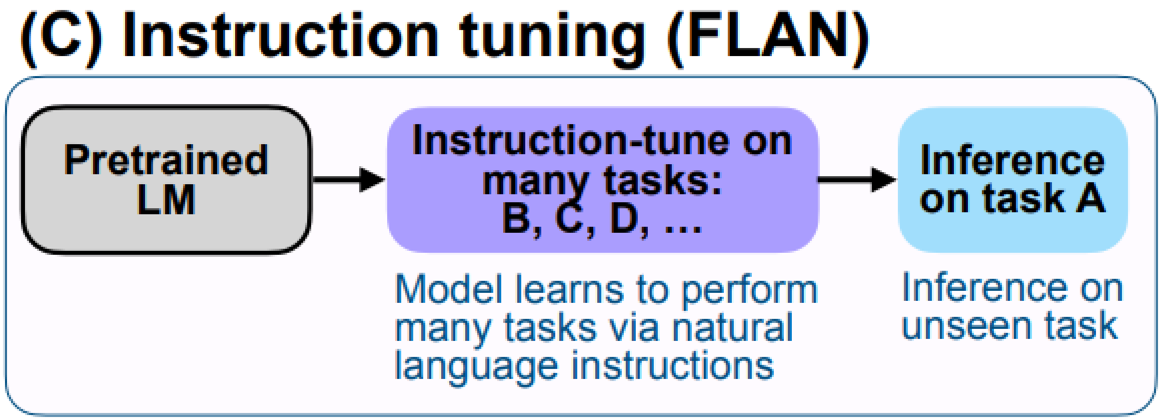
The company anticipated its operating profit to improve. // ________

http://ai.stanford.edu/blog/understanding-incontext/

Finetuing + Prompt
Instruction Tuning

- Finetuned a model to answer questions, not just continue text.

- Can generalize this ability to unseen language tasks.

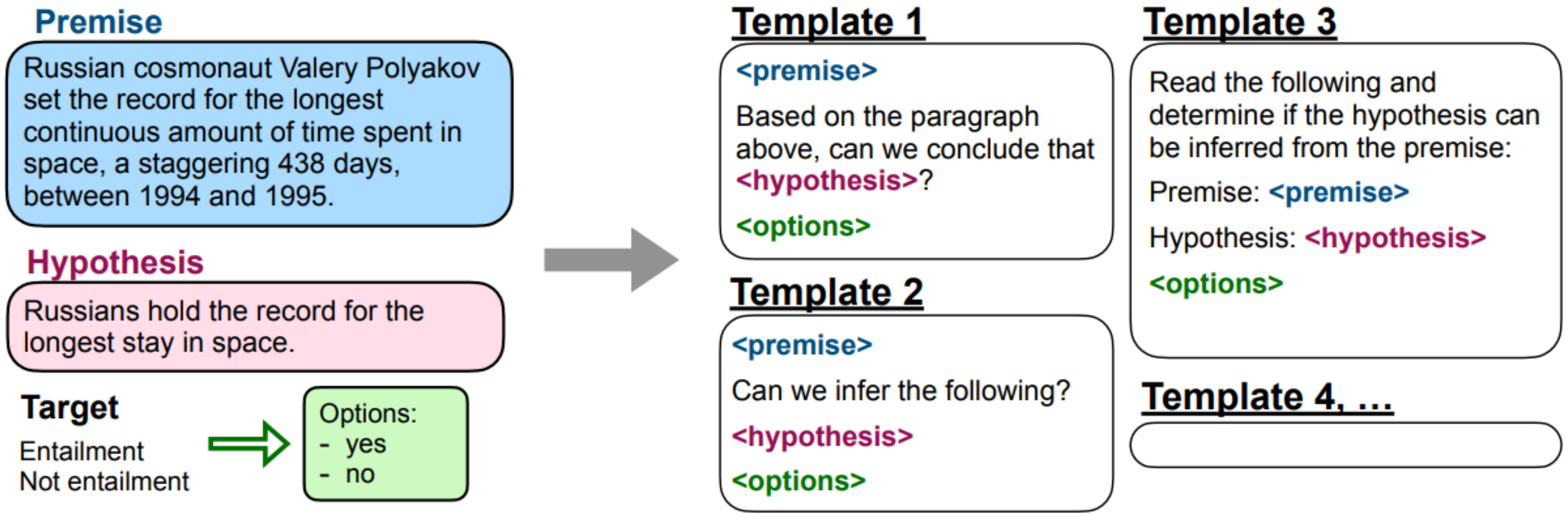


(C) Instruction tuning (FLAN)

Pretrained LM → Instruction-tune on many tasks: B, C, D, … → Inference on task A

Model learns to perform many tasks via natural language instructions

Inference on unseen task



Premise

Russian cosmonaut Valery Polyakov set the record for the longest continuous amount of time spent in space, a staggering 438 days, between 1994 and 1995.

Hypothesis

Russians hold the record for the longest stay in space.

Target

Entailment
Not entailment

Options:
- yes
- no

Template 1

<premise>

Based on the paragraph above, can we conclude that <hypothesis>?

<options>

Template 2

<premise>

Can we infer the following?

<hypothesis>

<options>

Template 3

Read the following and determine if the hypothesis can be inferred from the premise:

Premise: <premise>

Hypothesis: <hypothesis>

<options>

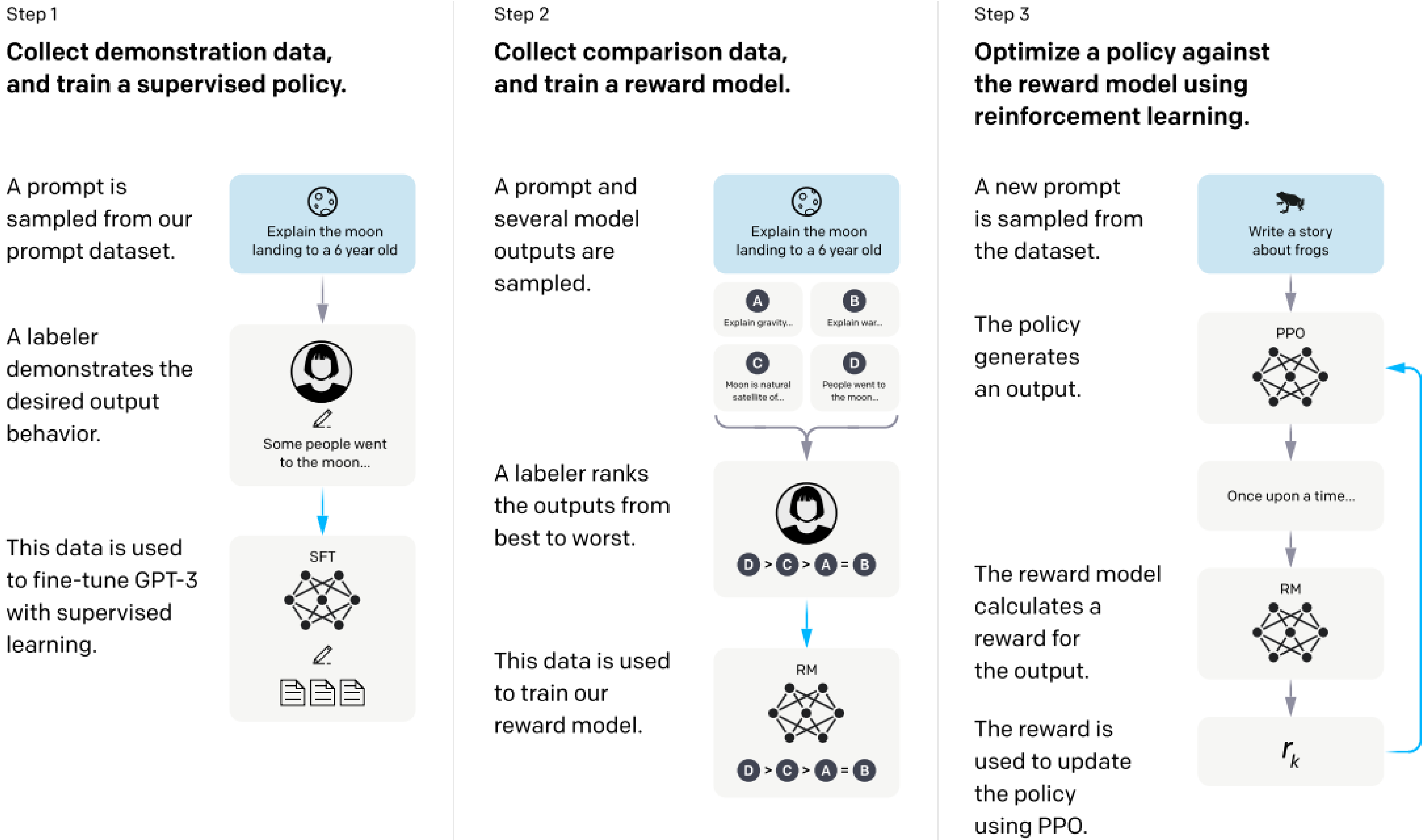
Template 4, …

Figure 4: Multiple instruction templates describing a natural language inference task.

FLAN
https://arxiv.org/abs/2109.01652.pdf

Instruct GPT / RLFH

- Training method behind ChatGPT



Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

Explain the moon landing to a 6 year old
Some people went to the moon...
SFT
Explain the moon landing to a 6 year old
A
Explain gravity...
B
Explain war...
C
Moon is natural satellite of...
D
People went to the moon...
D > C > A = B
RM
D > C > A = B
Write a story about frogs
PPO
Once upon a time...
RM
r_k

Supervised learning

Similar to FLAN

Reinforcement learning from human feedback

InstructGPT

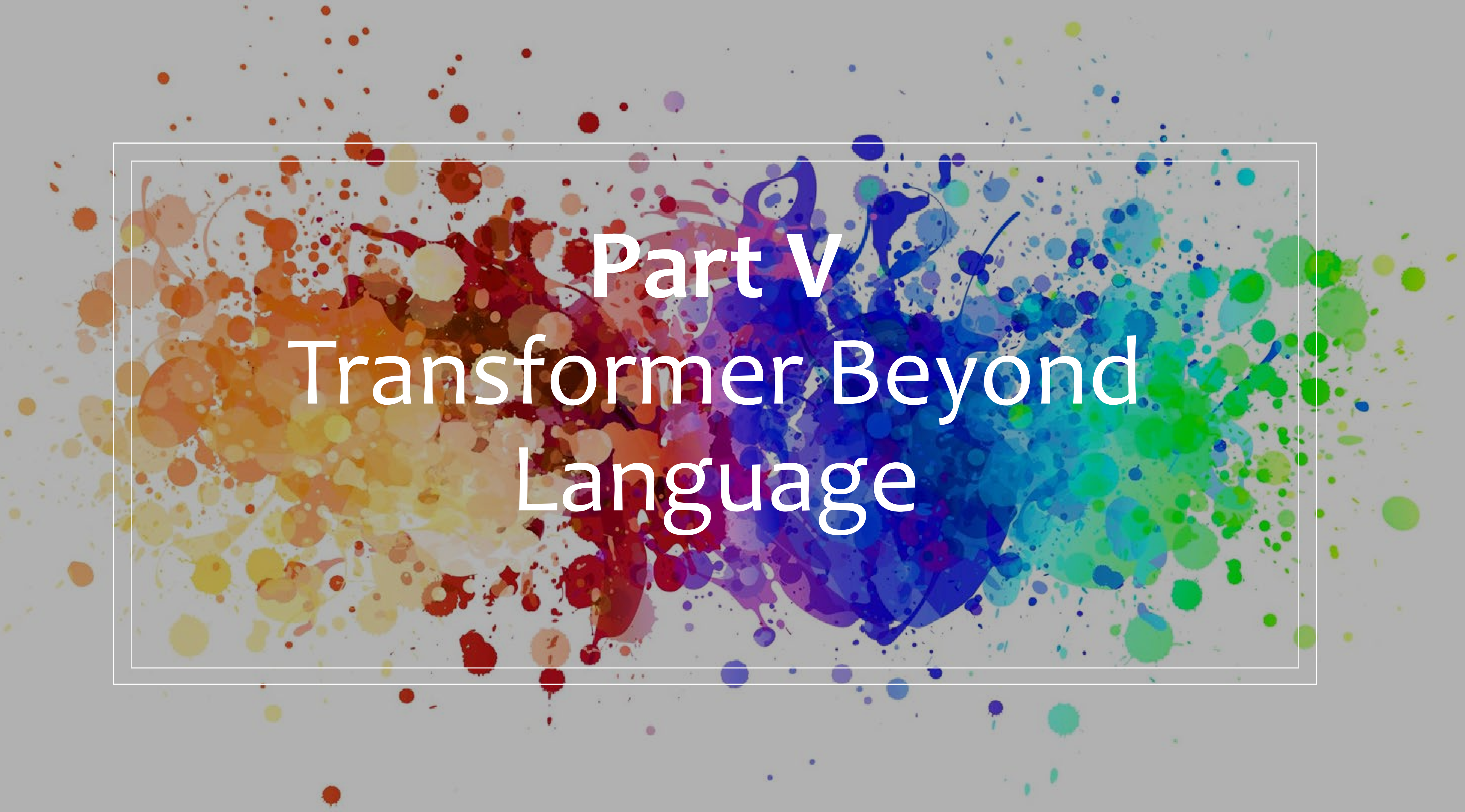
https://arxiv.org/abs/2203.02155

Interim Summary: Training transformer

- Language Transformer is usually pretrained by self-supervision.
    - Language modelling
    - Masked language modelling
    - Contrastive learning

- For downstream tasks, we do
    - Finetune (weights, heads, adapters)
    - Prompting

- Advanced technique
    - Instruction Tuning
    - RL from human feedback



Part V
Transformer Beyond Language

Transformer is a general **sequence** processing tool…

Many things are sequence…

Audio Signal

Music notes

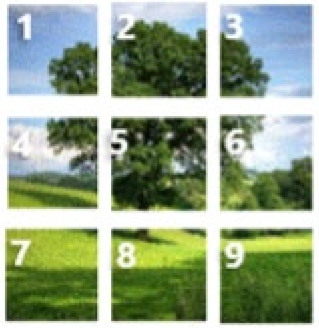
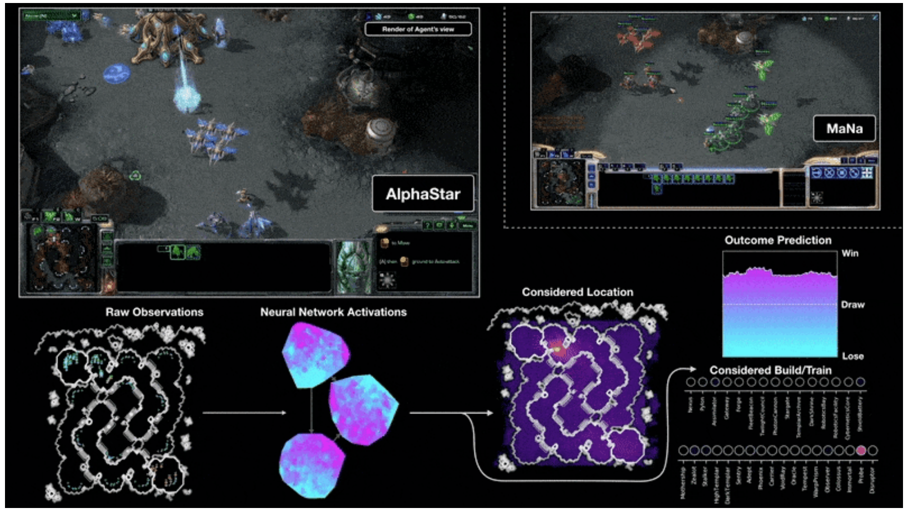
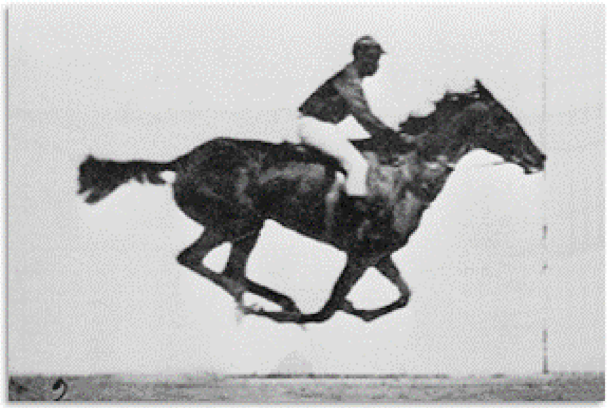


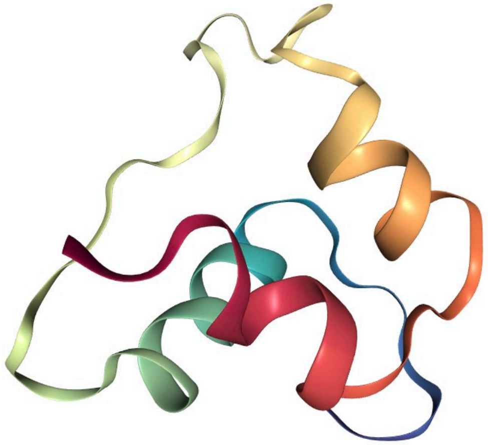
Image (as sequence of patches)



Action sequence in games



Video sequence



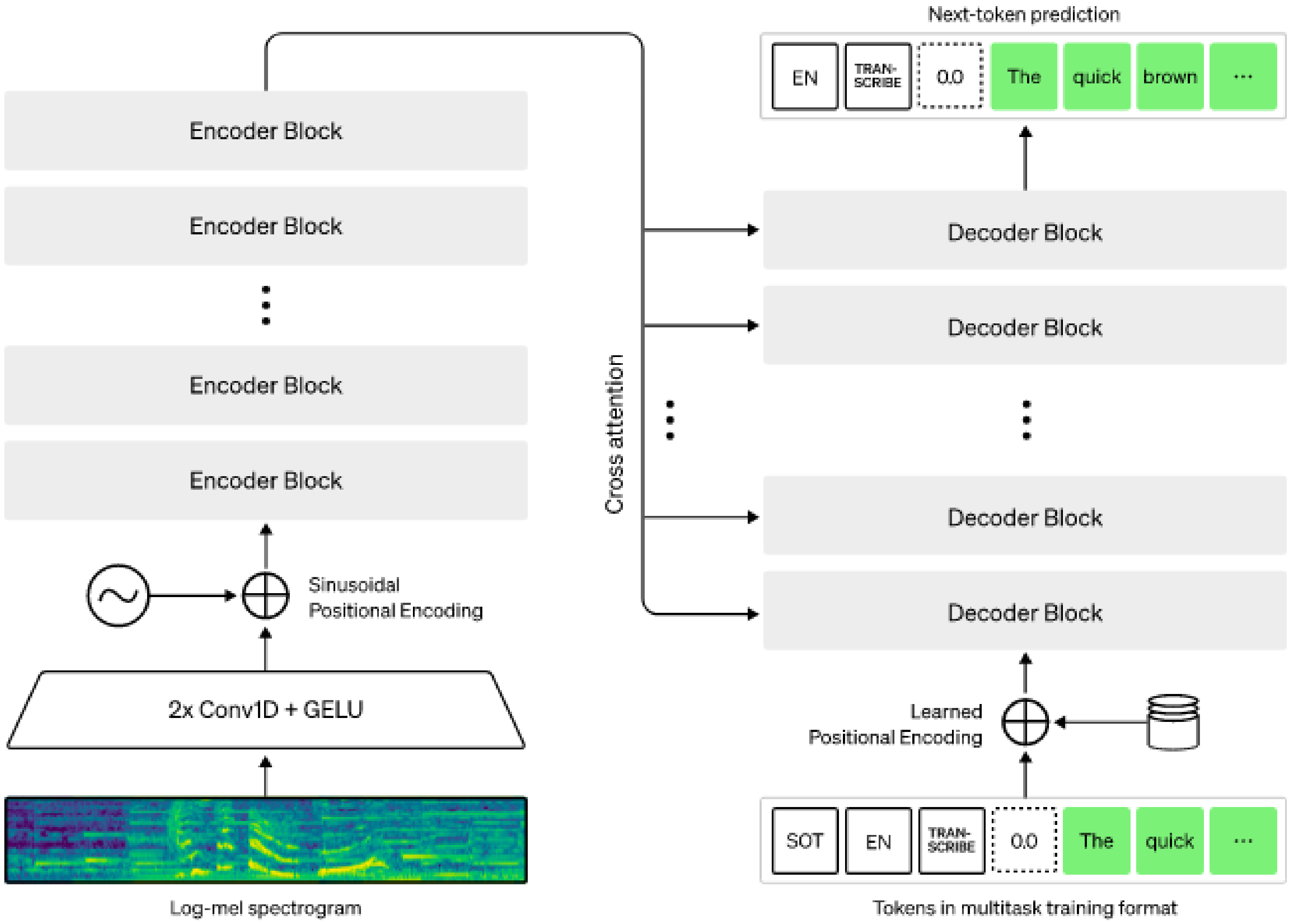
Protein sequence

https://towardsdatascience.com/transformers-an-exciting-revolution-from-text-to-videos-dc70a15e617b

One single architecture to *rule* them all.

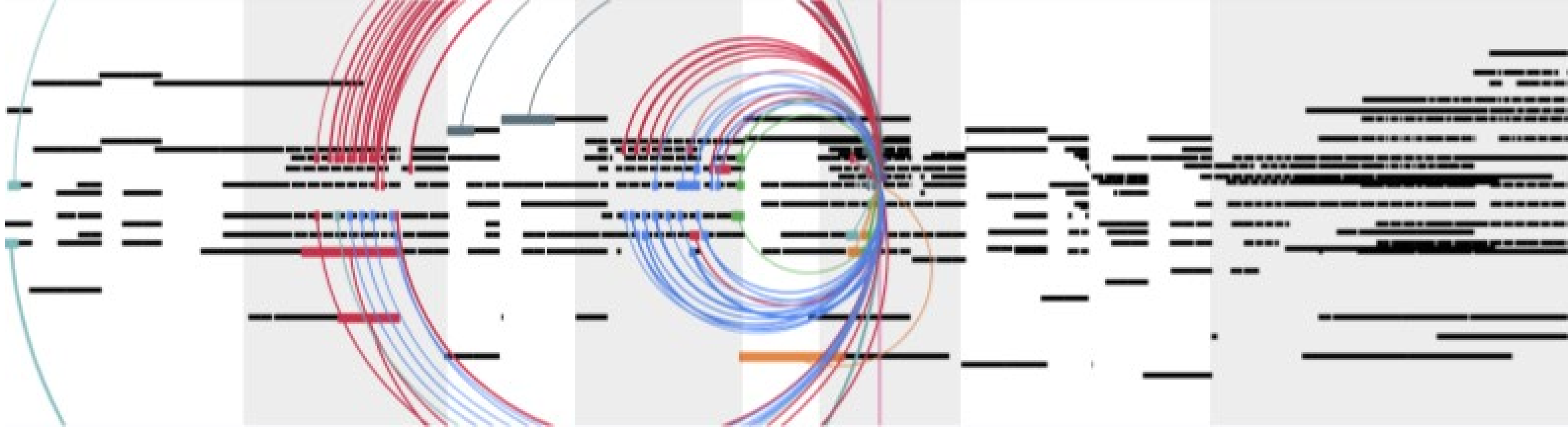
"I want to beat 10 of them!" – Donnie Yen

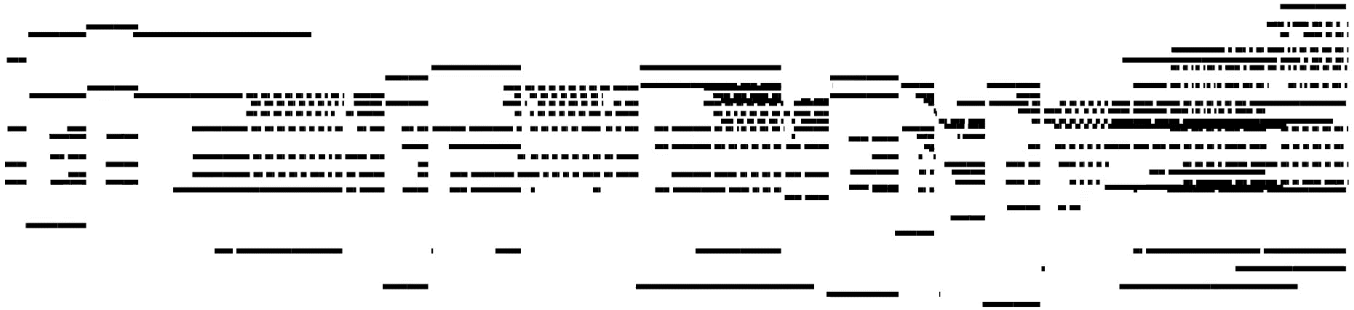
Audio transformer: Whisper



OpenAI, Whisper, 2022
https://openai.com/research/whisper

Music transformer

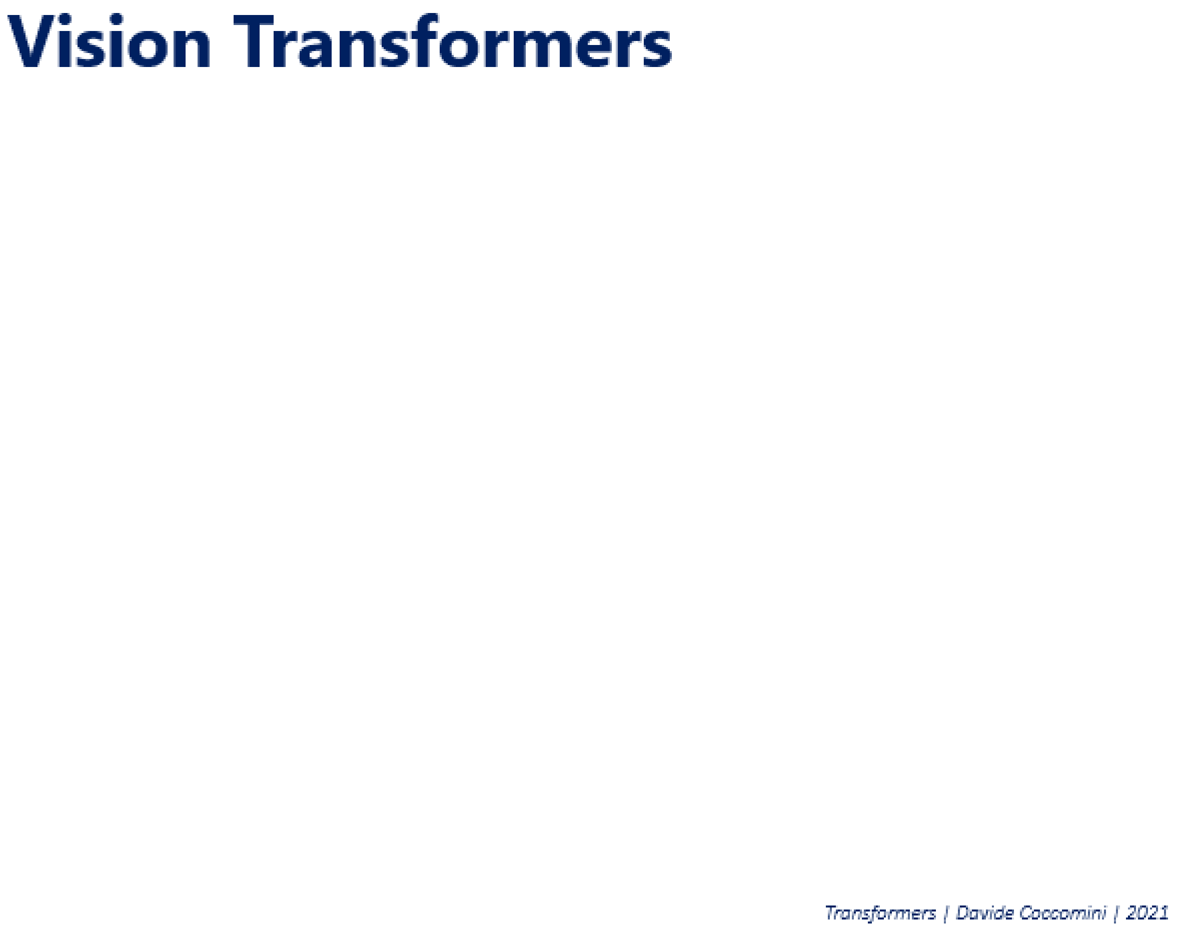




Huang, Cheng-Zhi Anna, et al. "Music transformer." (2018).
https://magenta.tensorflow.org/music-transformer

Vision transformers

- Patches as tokens / words
- Self attention between patch tokens.
- Extra [CLASS] token used for classification.



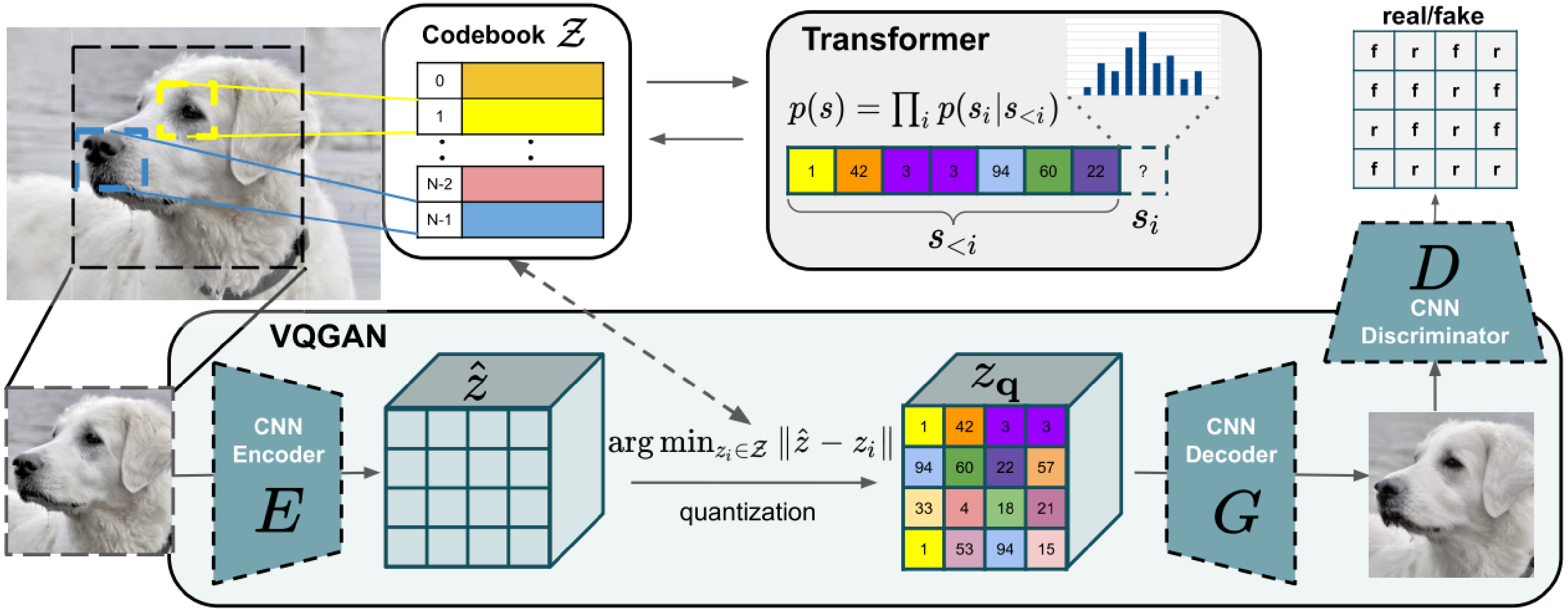
Vision Transformers

Transformers | Davide Coccomini | 2021

https://towardsdatascience.com/transformers-an-exciting-revolution-from-text-to-videos-dc70a15e617b

Transformers for image generation

- Encoded patches discretized as tokens
- Transformer model the sequence of tokens
- Images decoded from the tokens.



f	r	f	r
f	f	r	f
r	f	r	f
f	r	r	r

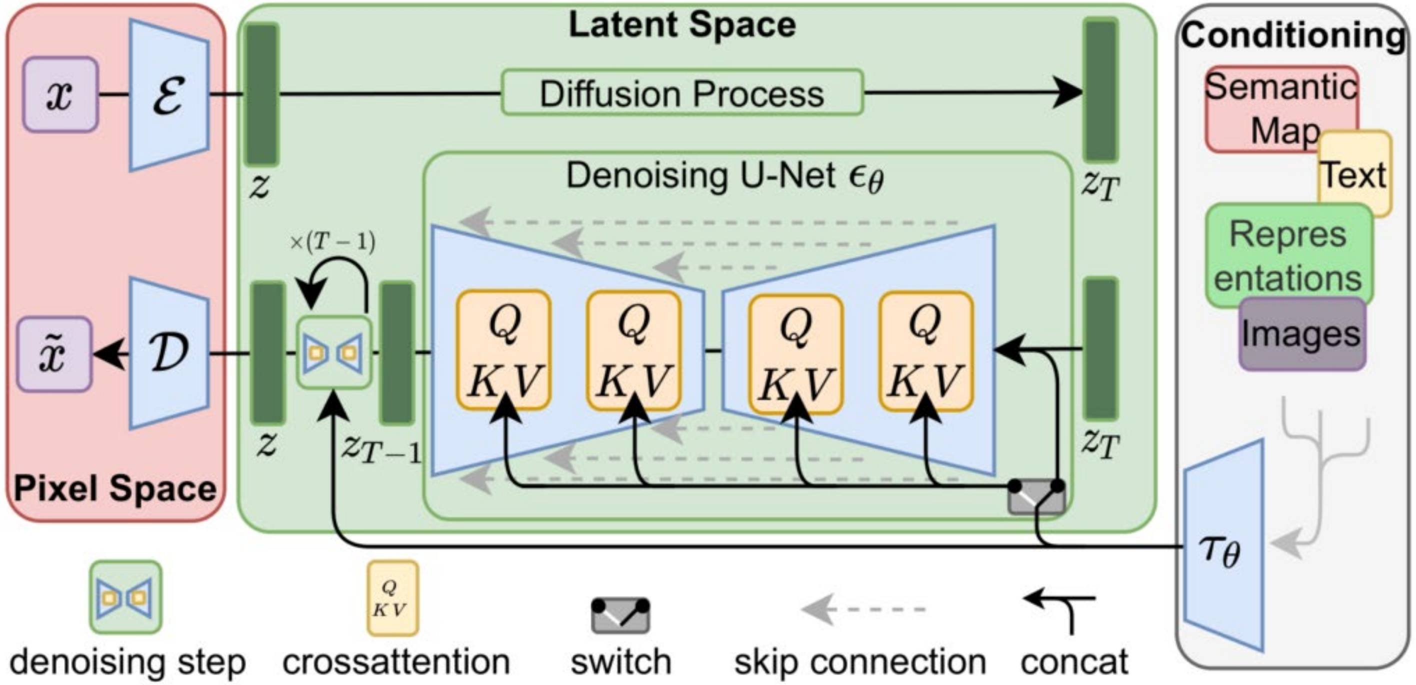
1	42	3	3
94	60	22	57
33	4	18	21
1	53	94	15

Figure 2. Our approach uses a convolutional *VQGAN* to learn a codebook of context-rich visual parts, whose composition is subsequently modeled with an autoregressive transformer architecture. A discrete codebook provides the interface between these architectures and a patch-based discriminator enables strong compression while retaining high perceptual quality. This method introduces the efficiency of convolutional approaches to transformer based high resolution image synthesis.

Esser, Patrick, Robin Rombach, and Bjorn Ommer. "Taming transformers for high-resolution image synthesis." *CVPR*. 2021.

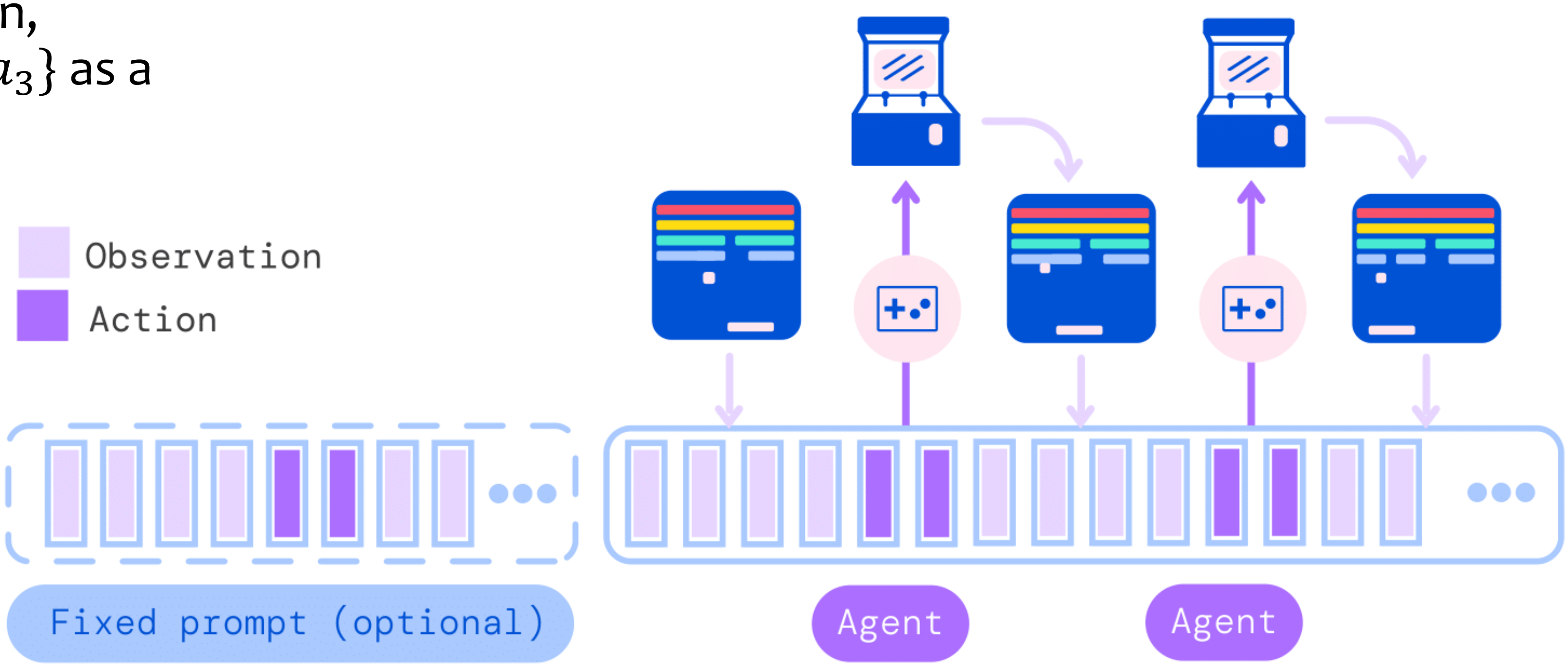
Stable Diffusion

- Image is influenced by text via cross attention.
- Image features influence each other via self attention.



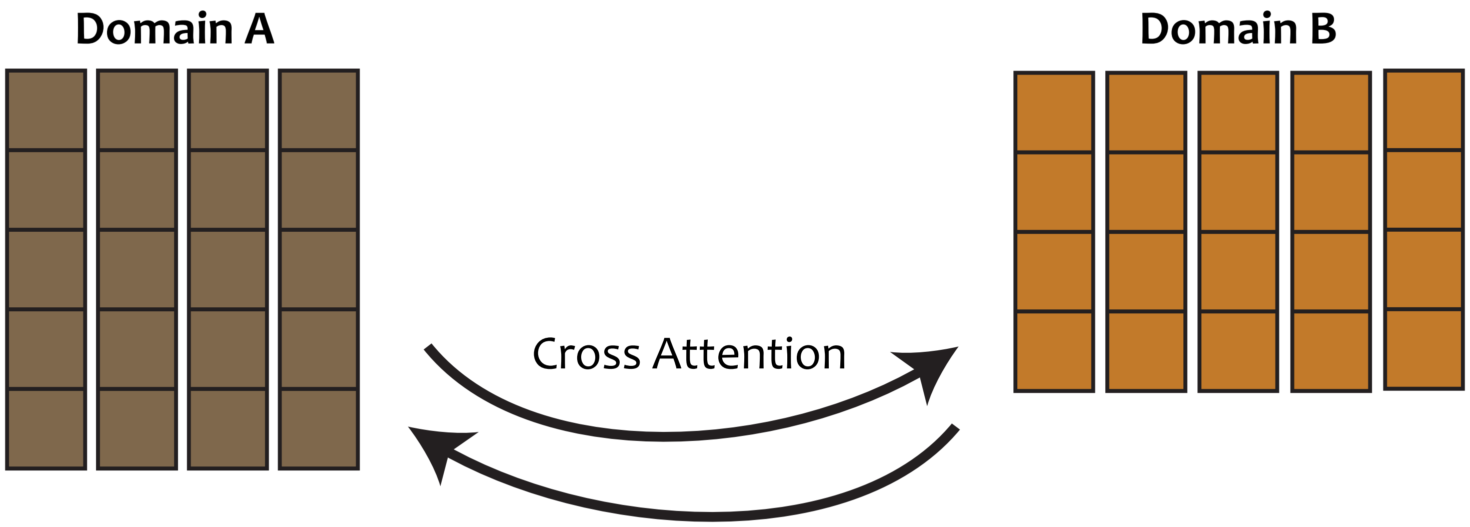
Transformers for Control / Game

- Game playing / control as sequence completion
  - Interleaved sequence of action, observation $\{s_0, a_1, s_1, a_2, s_3, a_3\}$ as a sequence of tokens.
  - Predict next action from past observation and actions.



Gato, DeepMind, 2022

Attention provides a flexible mechanism to let modalities influence each other

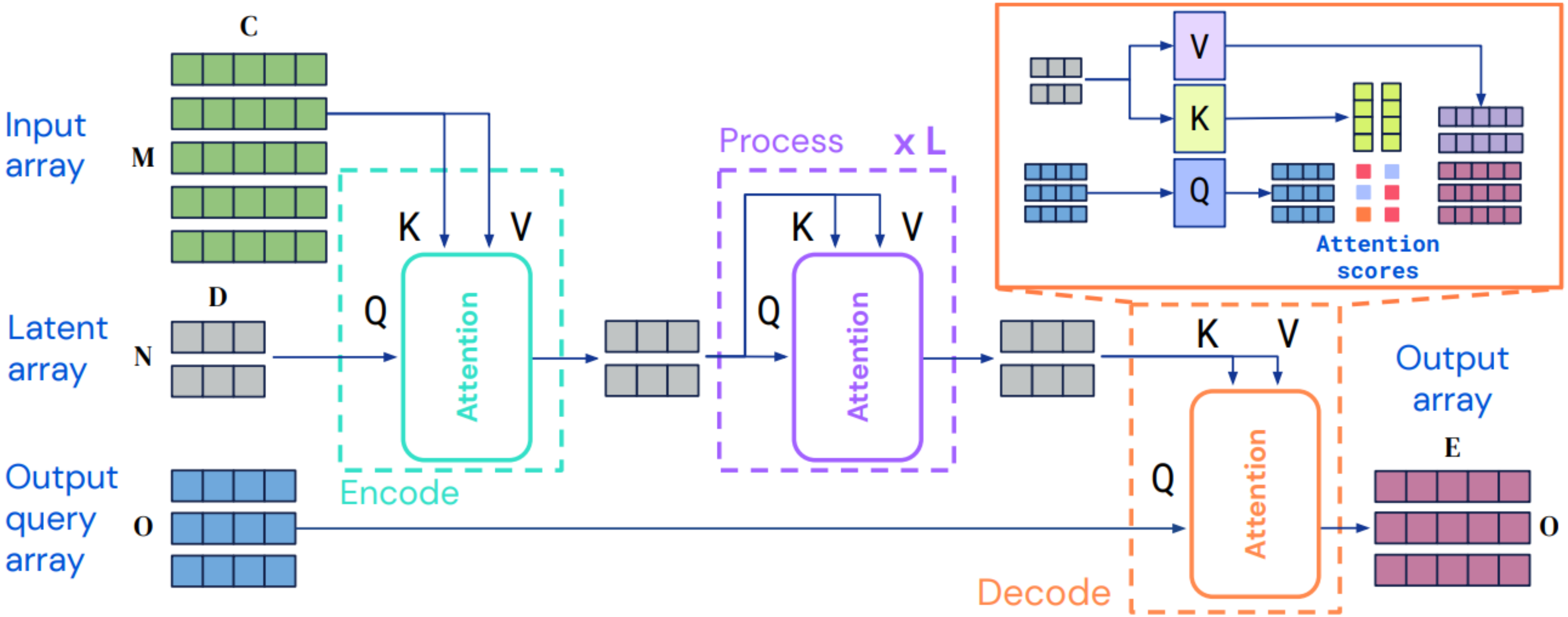


Modality A & B can be

Image, audio, video, text, action sequence, protein sequence, neural activity time series, …

Perceiver / Perceiver IO:
Transformer for general data perception

- General data processing method given data can be mapped into sequence of vectors
  - Use cross attention to fetch information from input
  - Self attention to process input.
  - Use cross attention to fetch relevant information and send to output.



Jaegle, Andrew, et al. "Perceiver: General perception with iterative attention." *ICML*, 2021.
Jaegle, Andrew, et al. "Perceiver io: A general architecture for structured inputs & outputs." *ICLR,* 2021

Transformers are powerful, but think about

- How to represent your signal as a sequence of tokens (image -> sequence)

- Use discrete tokens or continuous embedding vectors

- $O(N^2)$ memory cost for long sequence

- Higher computational cost c.f. simpler architecture (slower)

- Special token / header for the task.

- Training objective.

Interim Summary: beyond language

- Transformer / attention is a general info processing architecture, in one or more data modality.

- Found usage in vision, audio, action, music, image generation, neural signal processing etc.

Summary Recap

- Two pillars of NLP: good representation and language modelling.
- Attention mechanism & Transformers architecture
- Self-supervised pre-training objectives (e.g. BERT, GPT, CLIP)
- Pre-training and finetuning paradigm; Prompting; Instruction Tuning
- Application of transformer beyond language. (e.g. Audio, Music, Vision, Game, Control etc.)

Thanks for your listening!