arnes
povezujemo znanje

EURO
SLING

MREŽA ZNANJA
Ljubljana, 3.–5. december 2024

Gradnja in prilagajanje velikih jezikovnih modelov za slovenščino

prof. dr. Marko Robnik-Šikonja
Unievrza v Ljubljani, Fakulteta za računalništvo in informatiko

UNIVERSITY OF LJUBLJANA | FRI Faculty of Computer and Information Science

Gradnja in prilagajanje velikih jezikovnih modelov za slovenščino

prof. dr. Marko Robnik-Šikonja

Dnevi Sling

5. december 2024

Vsebina

Veliki jezikovni modeli (VJM, angl. large language models LLMs)

Nevronska arhitektura transformer

Gradnja in prilagajanje VJM

VJM za slovenščino

izzivi za visokozmogljivo računanje

nekatere slike povzete po: Jay Alammar, Jacob Devlin

Verjetnostno modeliranje jezika – jezikovni modeli

- Izračunati verjetnost stavka oz. zaporedja besed:

  $P(W) = P(w_1,w_2,\ldots w_n)$

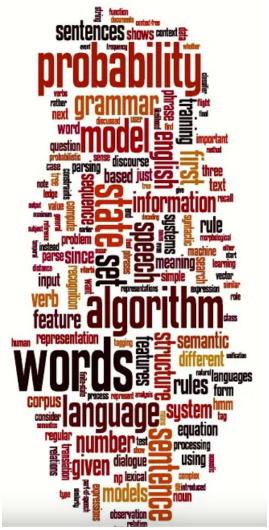
- Npr., strojno prevajanje:
  - P(**high** winds tonight) > P(**large** winds tonight)

- Izračun naslednje besede v zaporedju

  $P(w_i|w_1,w_2,\ldots w_{i-1})$
  Včeraj smo se potepali po ____

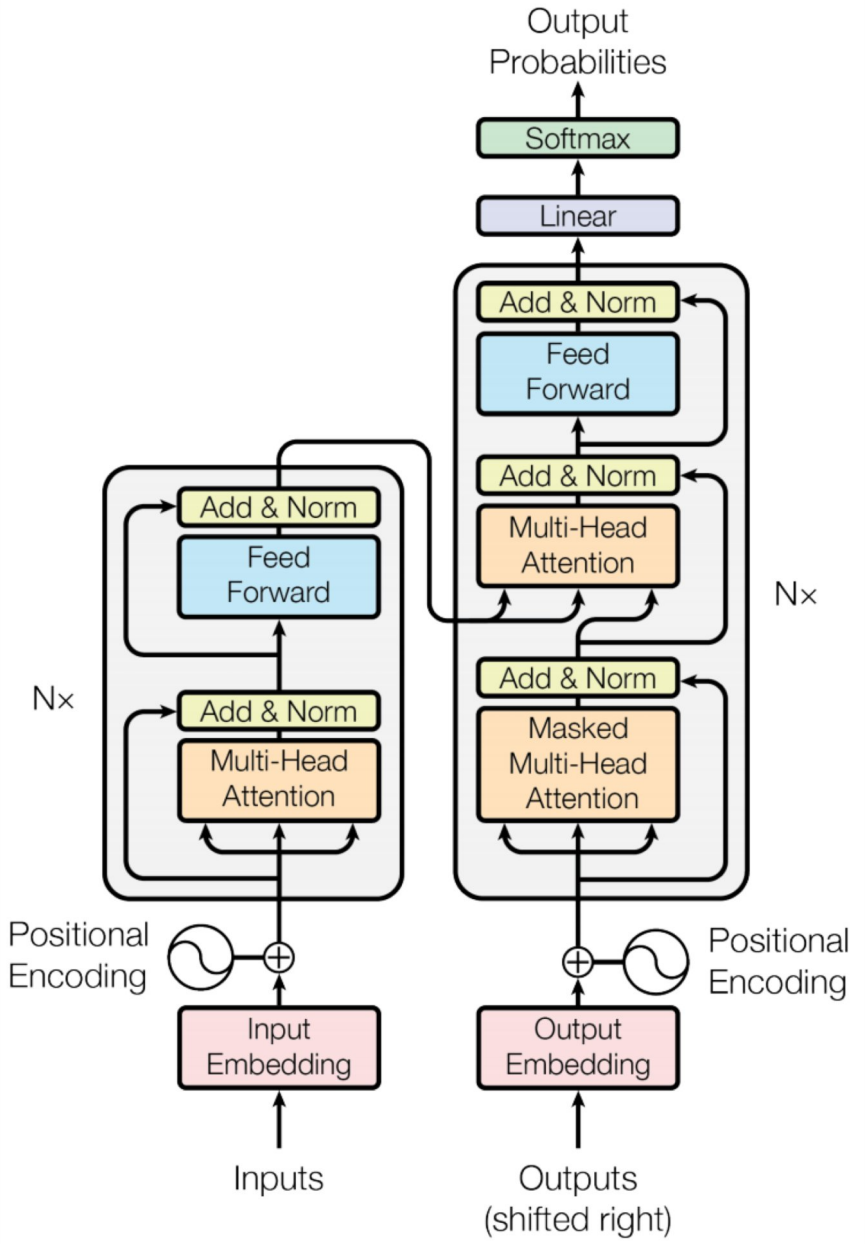
- Izračun maskirane besede v zaporedju
  $P(w_i|w_1,w_2,..w_{i-1},w_{i+1},\ldots w_n)$
  Kljub _____ se nista mogla odpovedati večernemu spehodu.

- Model za izračun teh verjetnosti imenujemo jezikovni model (language model, LM)



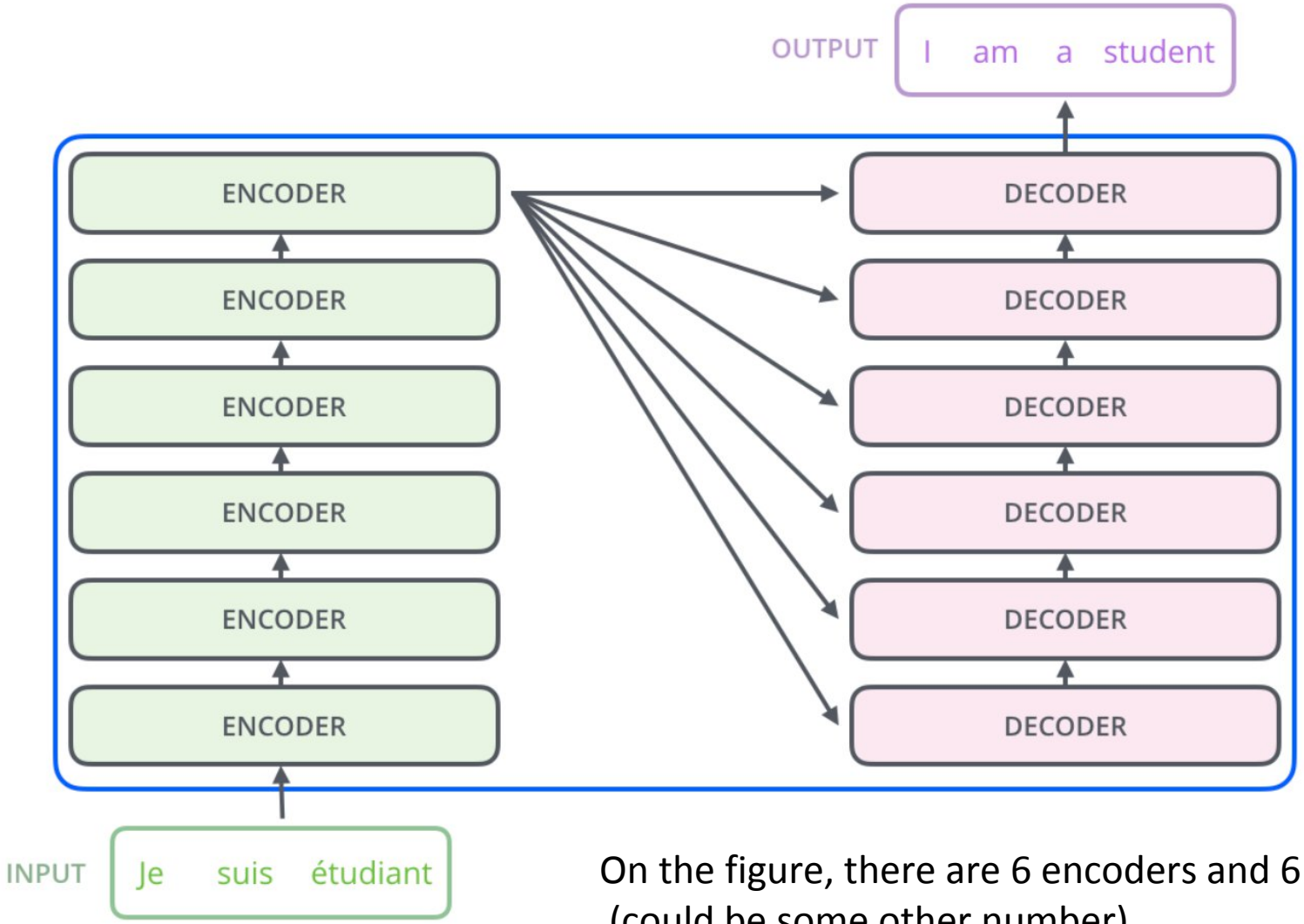
Arhitektura Transformer

- trenutno najuspešnejša arhitektura globokih nevronskih mrež
- nerekurzivna, kodirnik-dekodirnik arhitektura
- fiksna vhodna sekvenca (tipično precej dolga, > 512 žetonov)
- prilagojena GPU procesiranju
- prilagojena ekstremnemu paralizmu: skalarni produkti, matrične operacije
- temelji na uporabi mehanizma pozornosti
- dobra skalabilnost
- kodirnik in dekodirnik se lahko uporabljata ločeno
- vsebuje preskočne povezave za preprečevanje pozabljanja



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

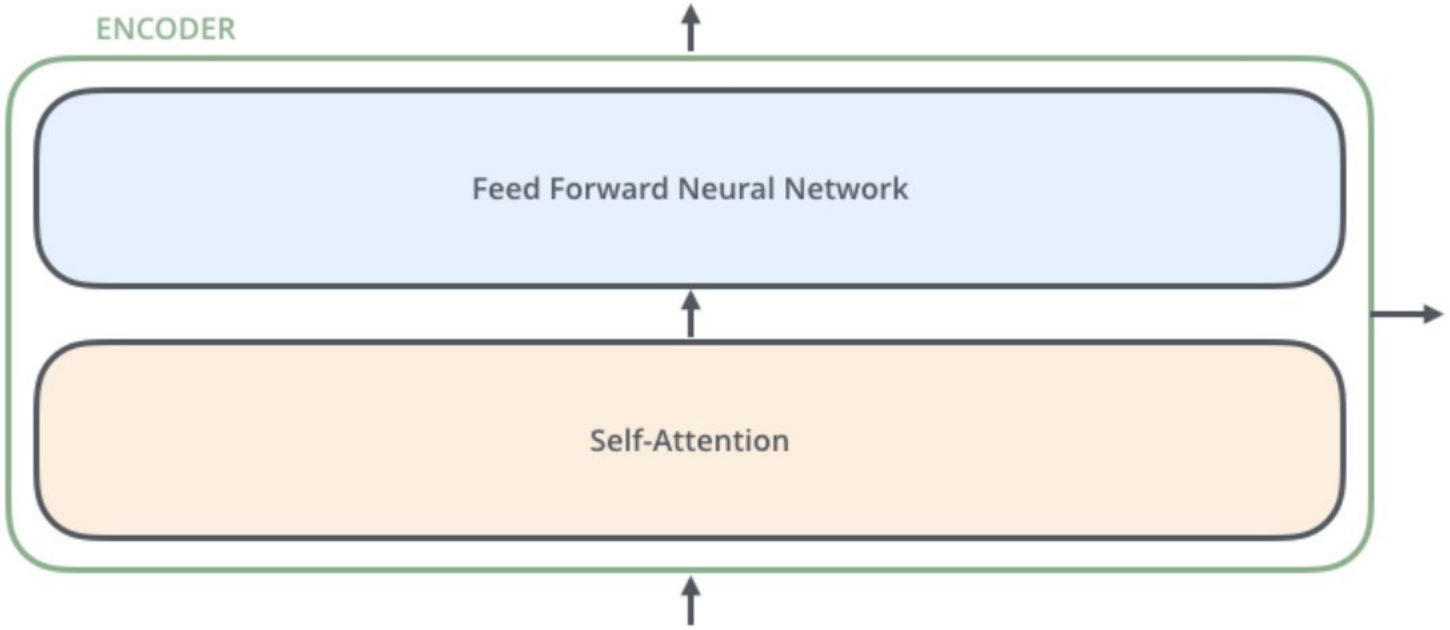
Transformer is an encoder-decoder model



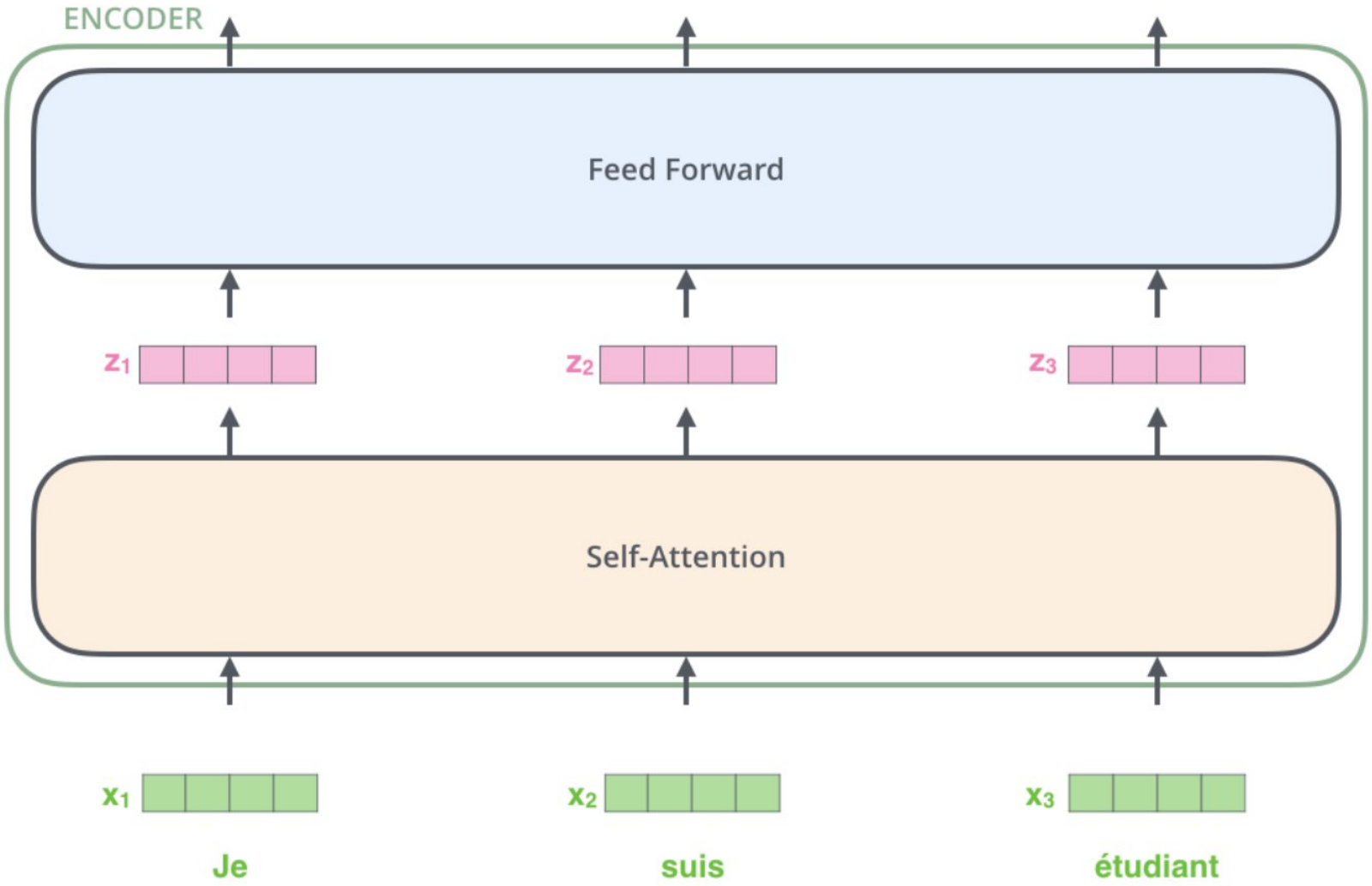
On the figure, there are 6 encoders and 6 decoders (could be some other number).

Transformer: encoder

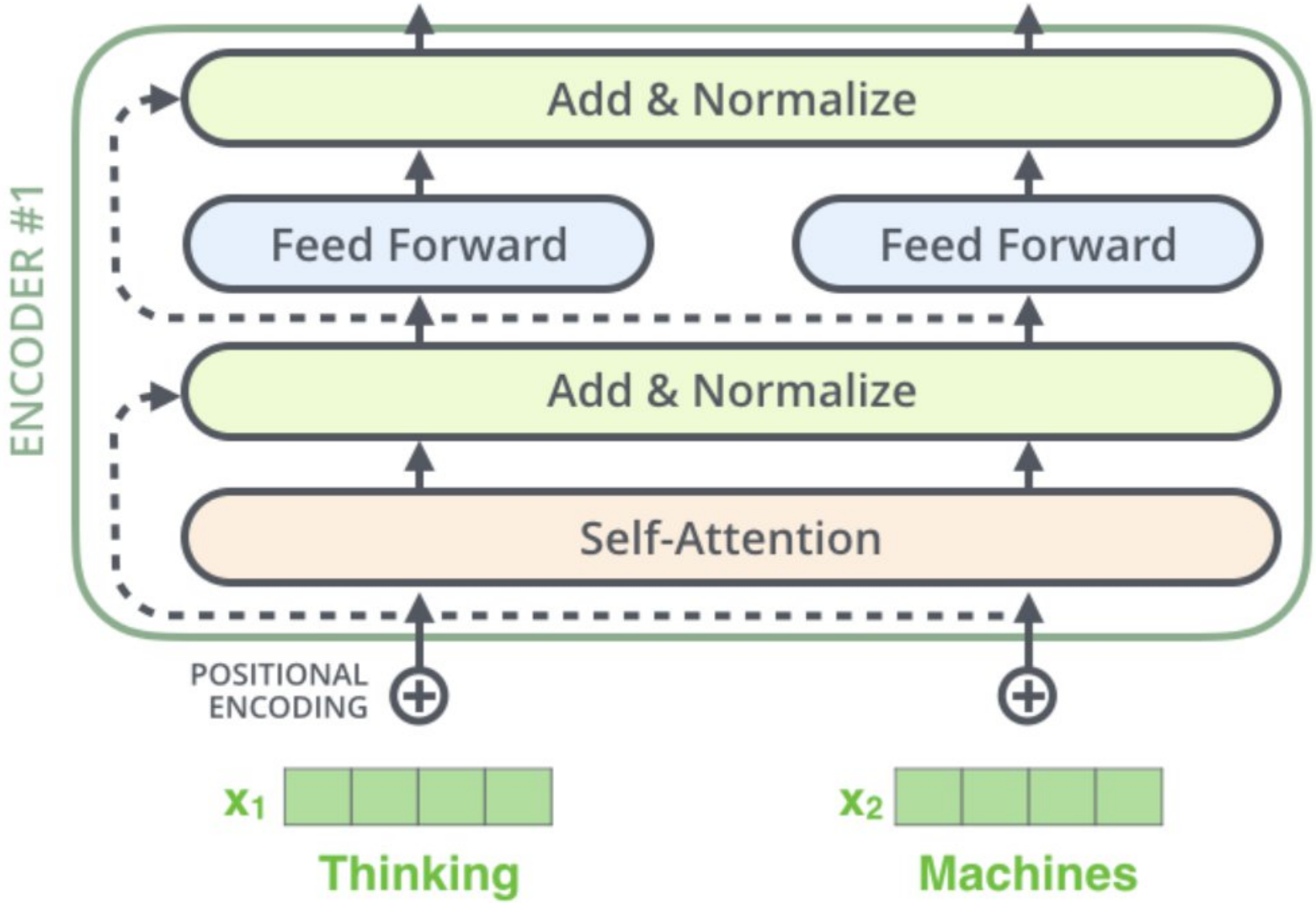
- Each layer has two parts
- no weight sharing between different encoders
- self-attention helps to focus on relevant parts of the input



Normalization keeps vectors small enough

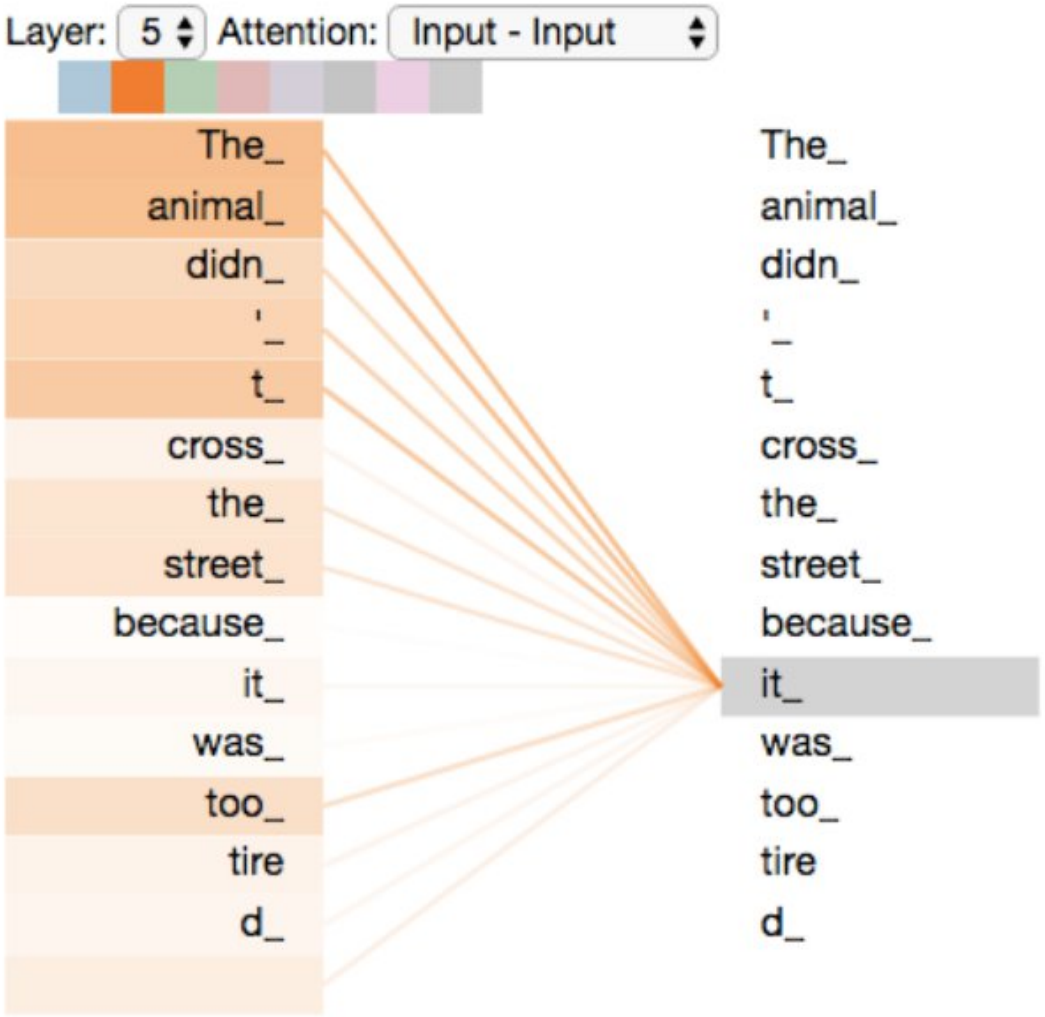


Encoder – top level view

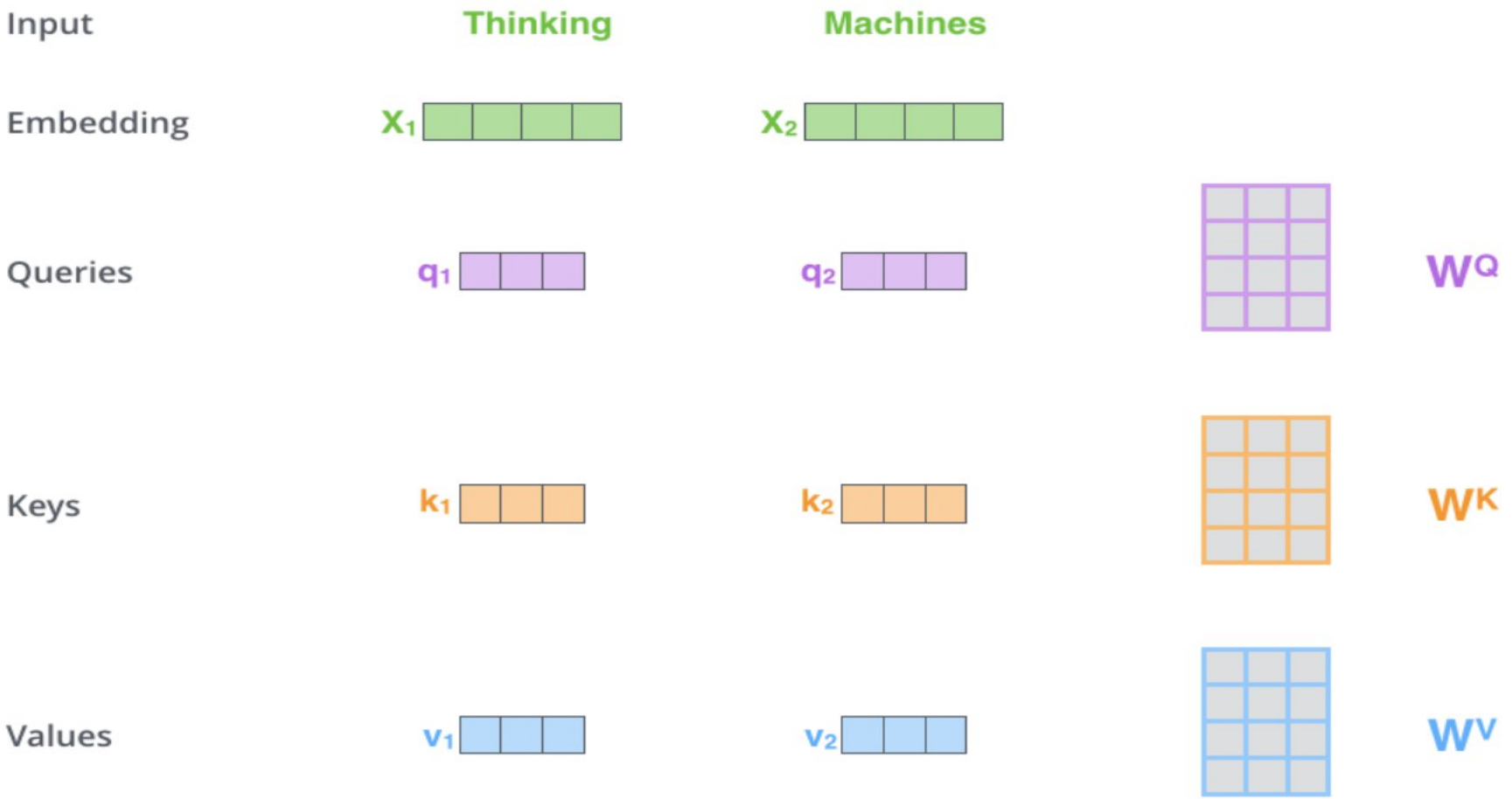


Illustrating self-attention

- As the model processes each each position in the input sequence (i.e. word), self-attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word
- *”The animal didn't cross the street because it was too tired”*
- Is *”* it *”* referring to the street or to the animal?
- *What about here:*
- *”The animal didn't cross the street because it was too wide”*
- As we are encoding the word "it" in encoder #5 (the top encoder in the stack), part of the attention mechanism was focusing on "The Animal" and baked a part of its representation into the encoding of "it".

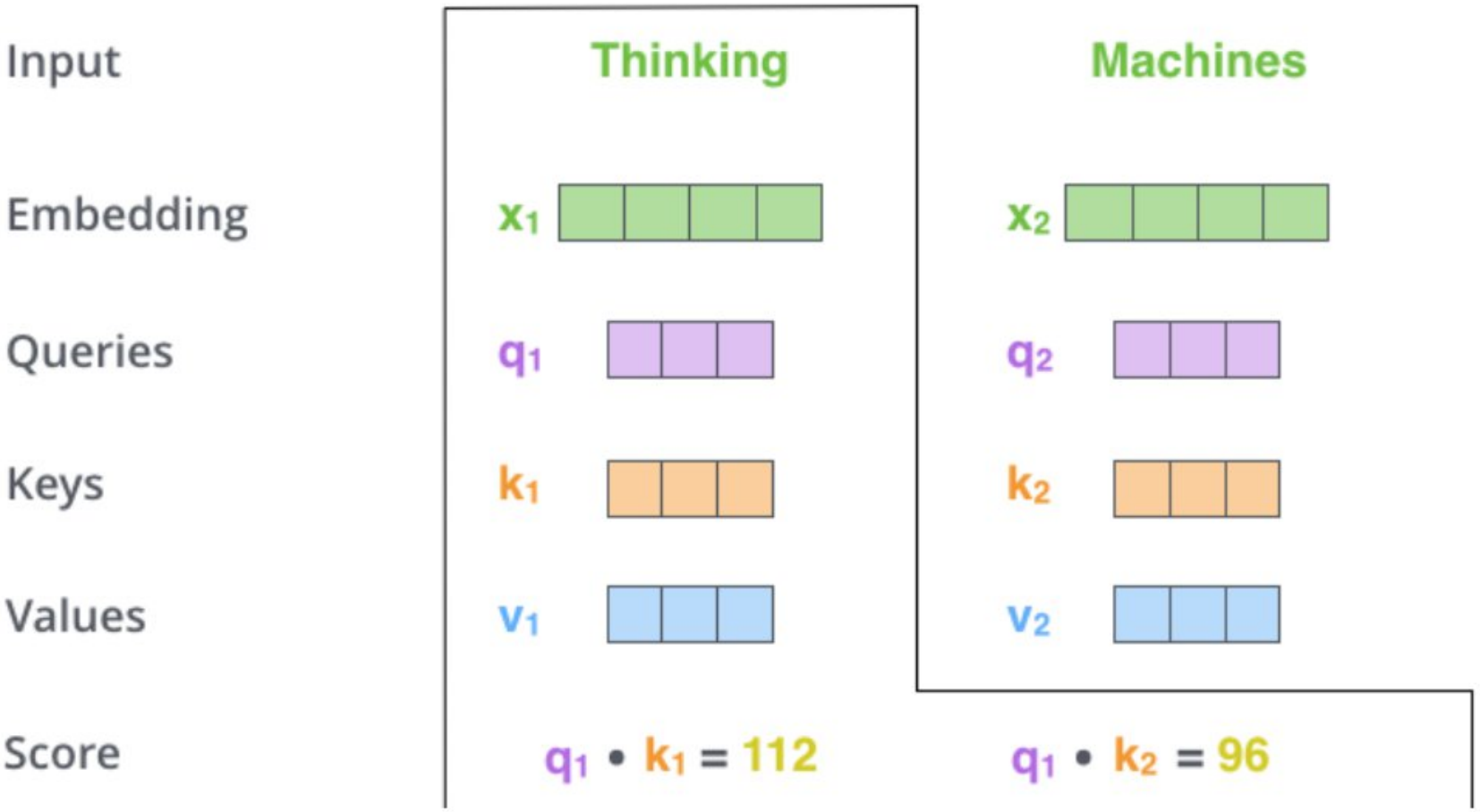


Self-attention details 1/4

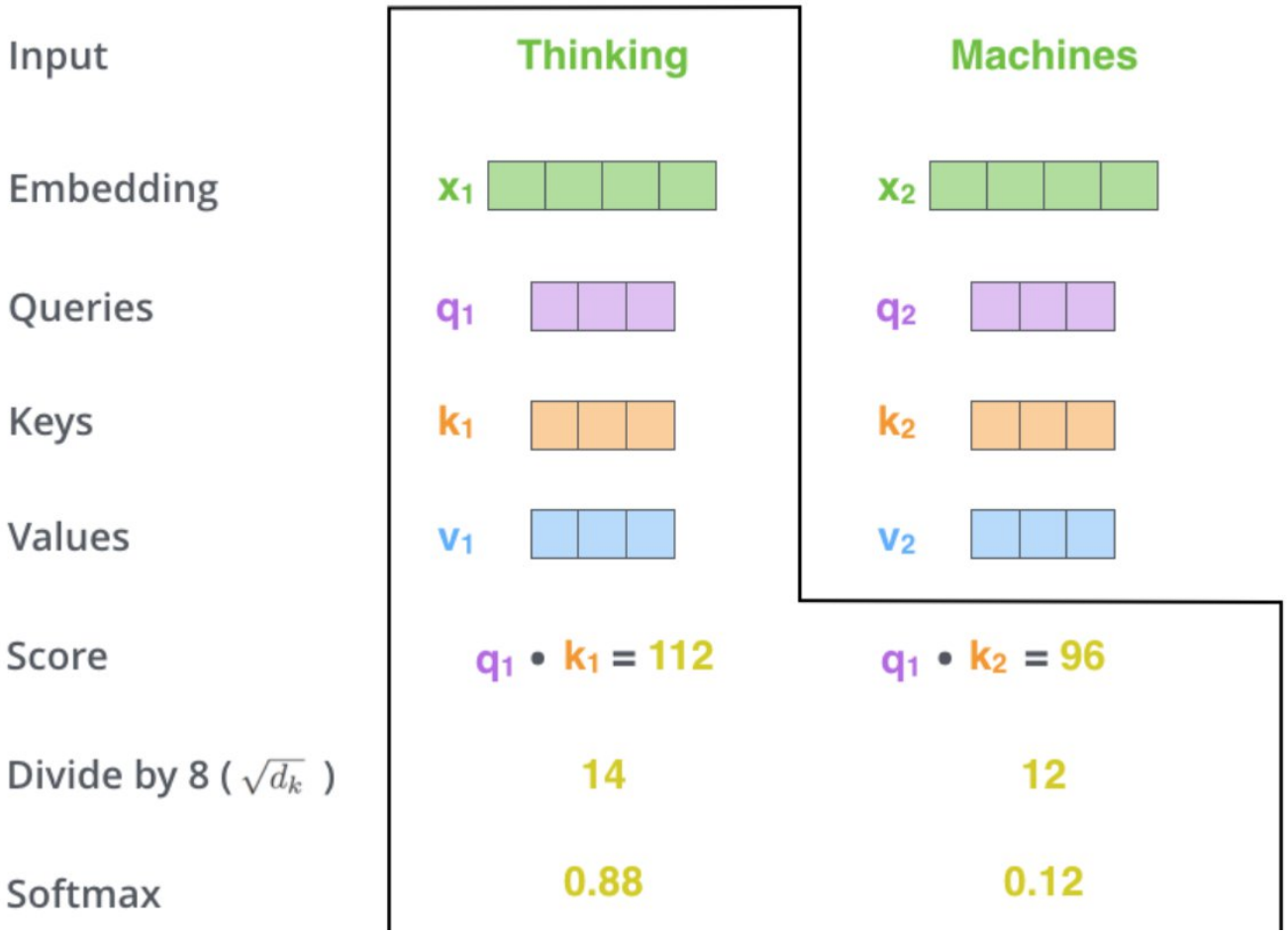


Multiplying x_1 by the W^Q weight matrix produces q_1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

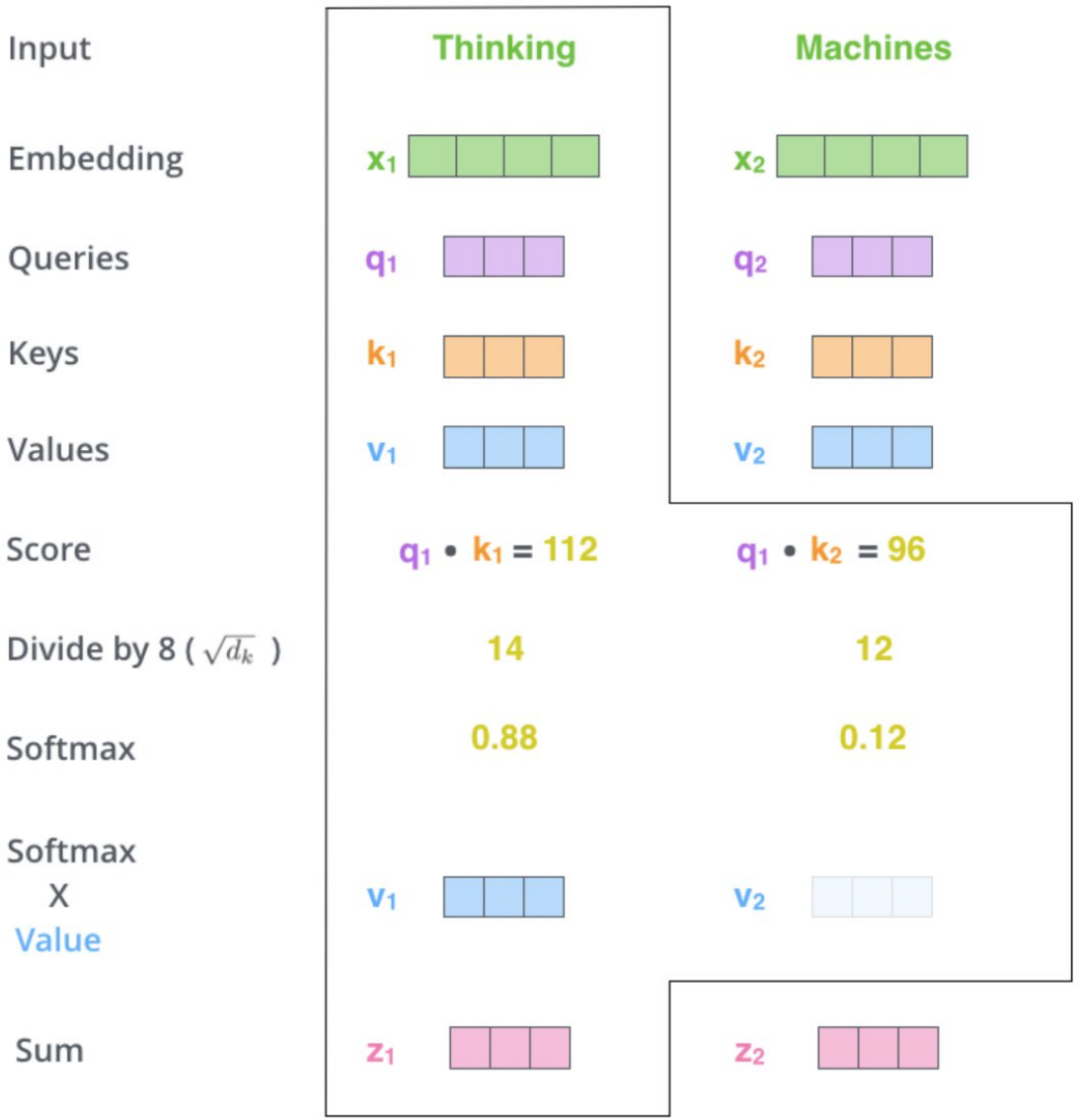
Details 2/4: scoring



Details 3/4: normalization of scores

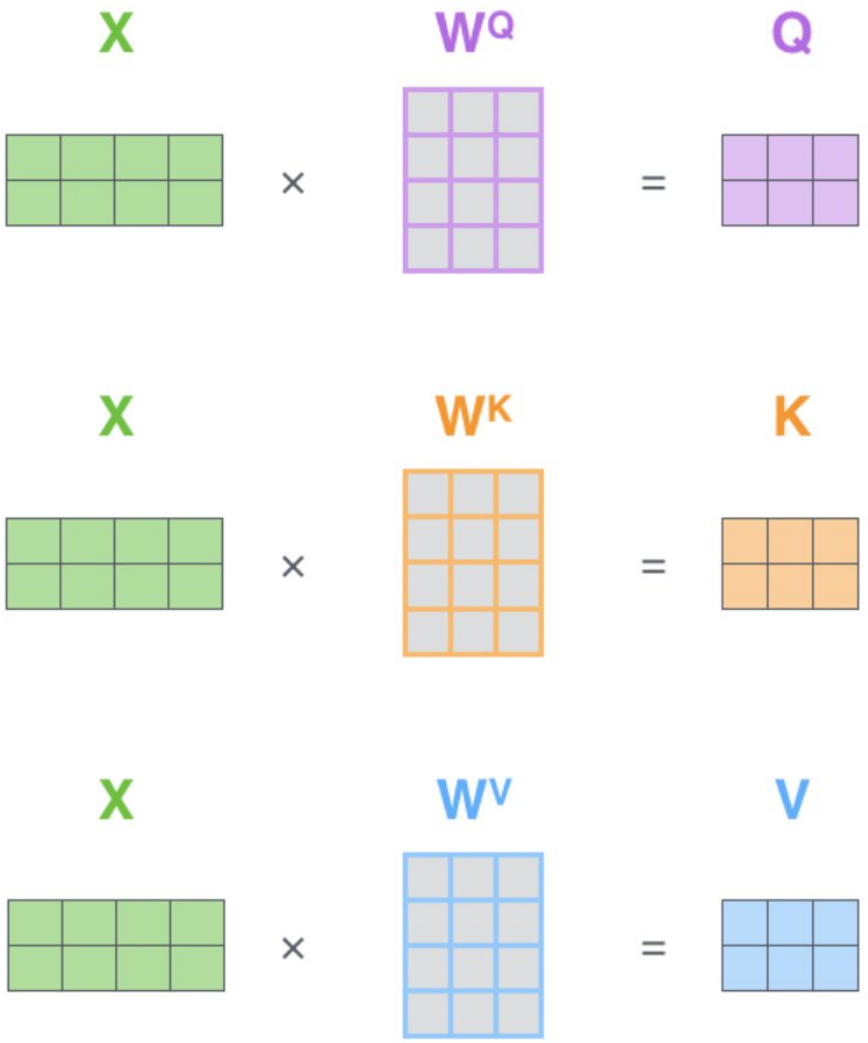


Details 4/4: self-attention output



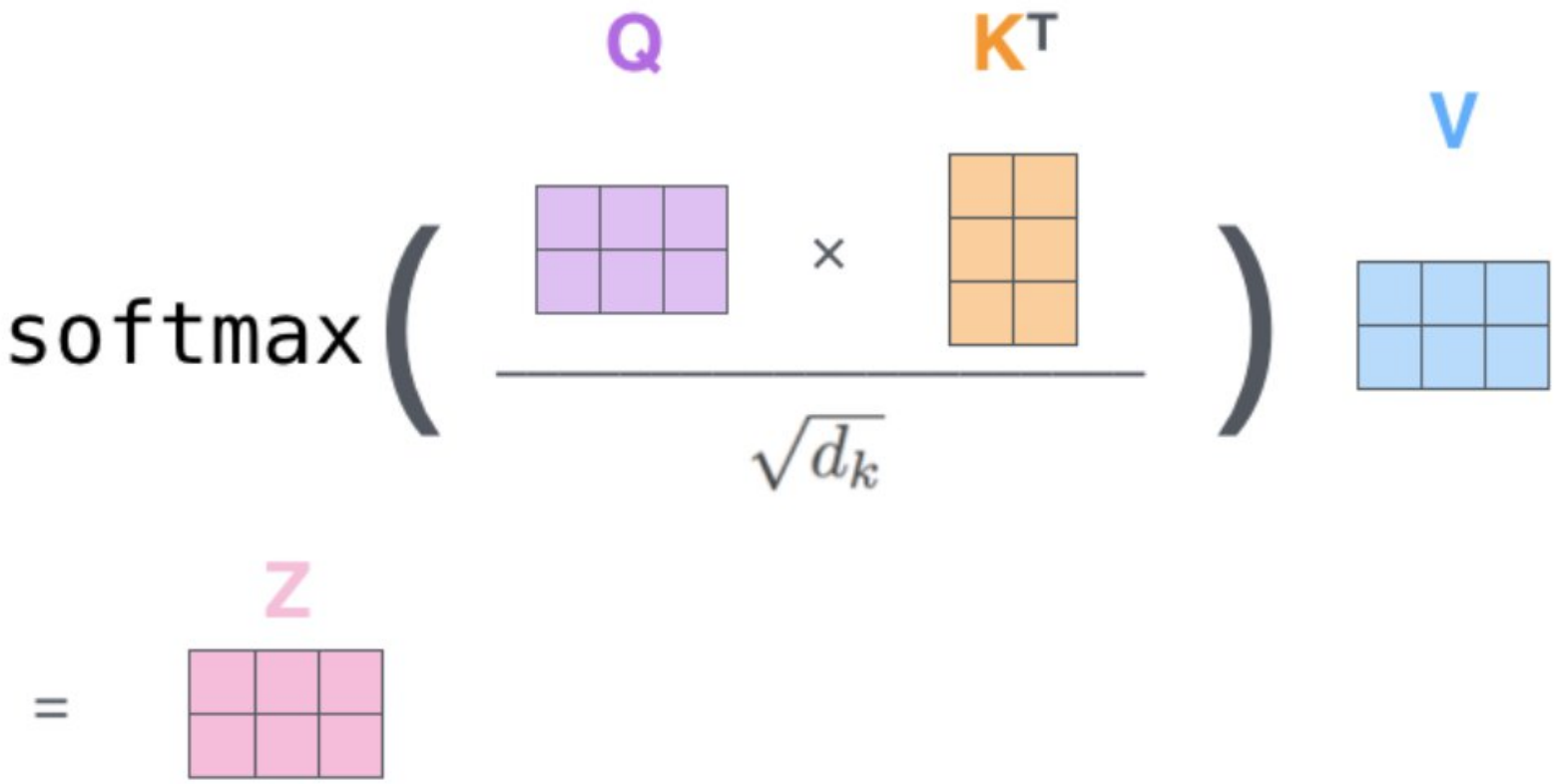
Matrix calculation of self-attention 1/2

Every row in the X matrix corresponds to a word in the input sentence.
The embedding vector x (512) is larger then the q/k/v vectors (64)

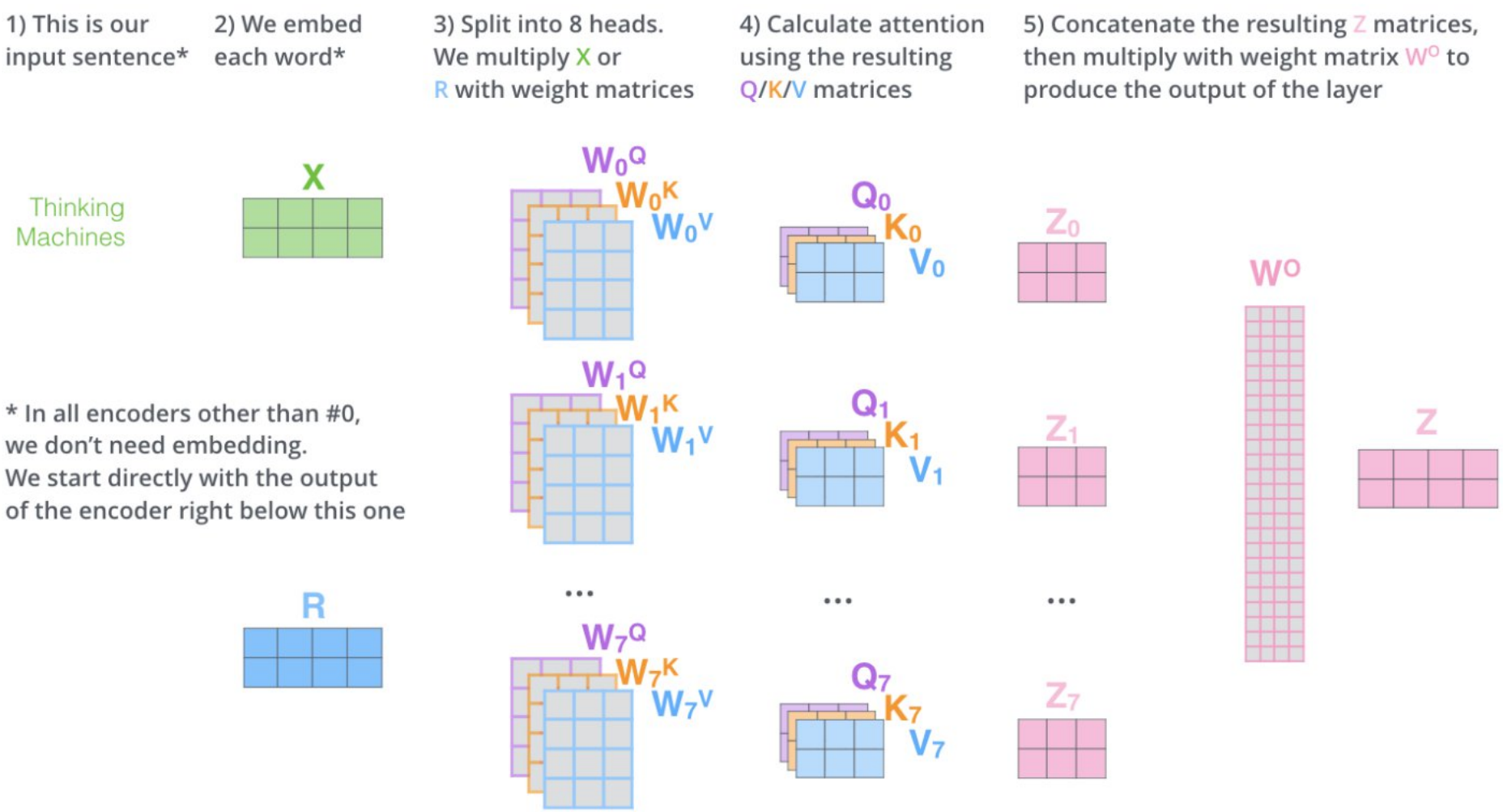


Matrix calculation of self-attention 2/2

- final calculation

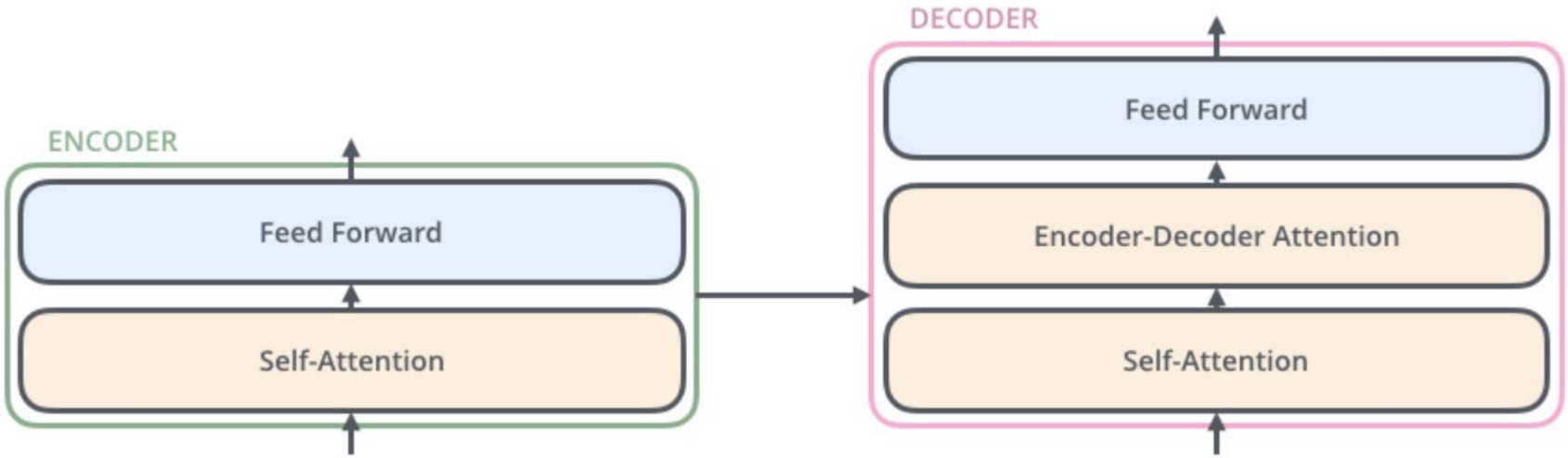


Summary of self-attention

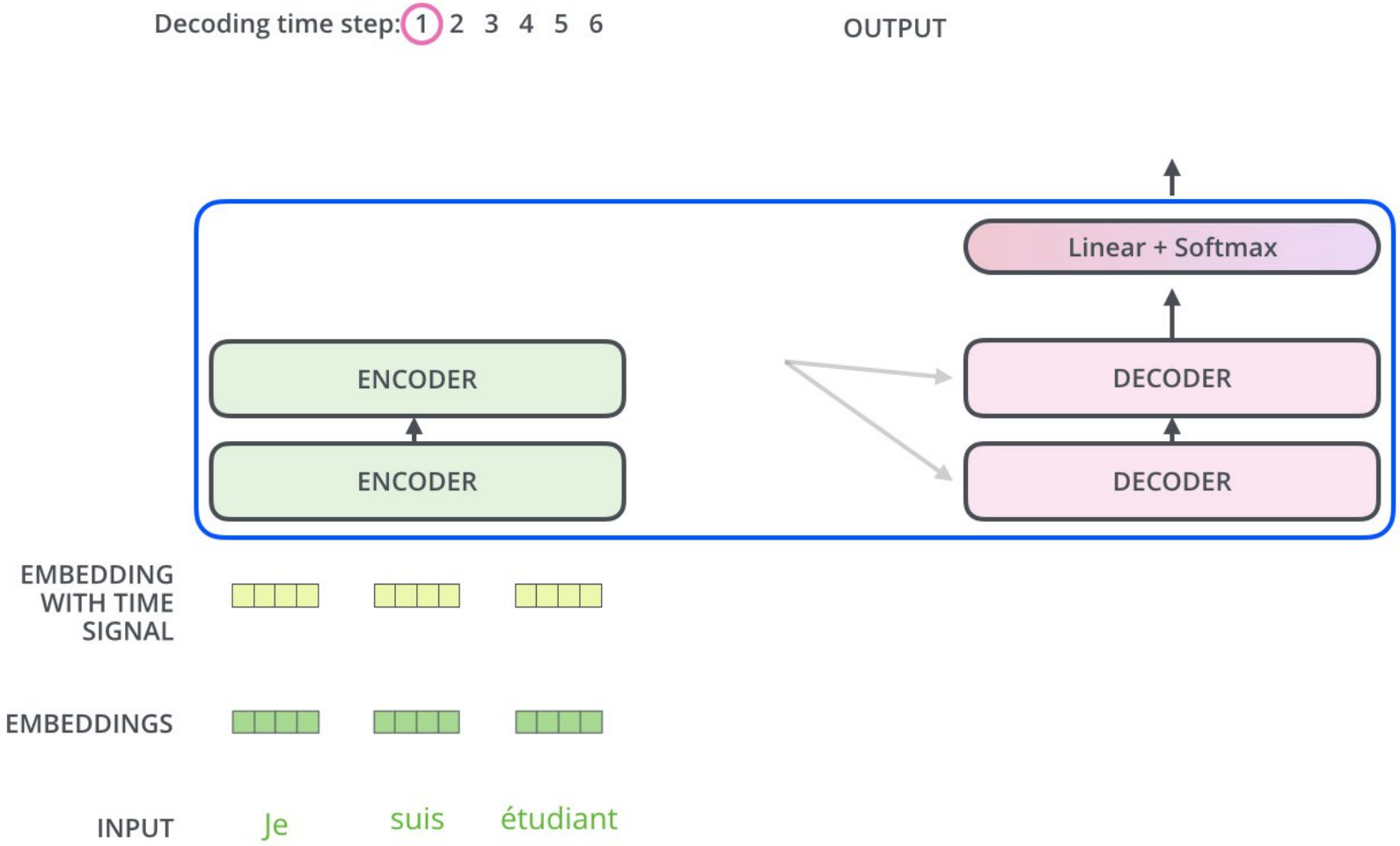


Transformer: decoder

- the same as encoder but with an additional attention layer in-between, receiving input fom encoder (called encoder-decoder attention)

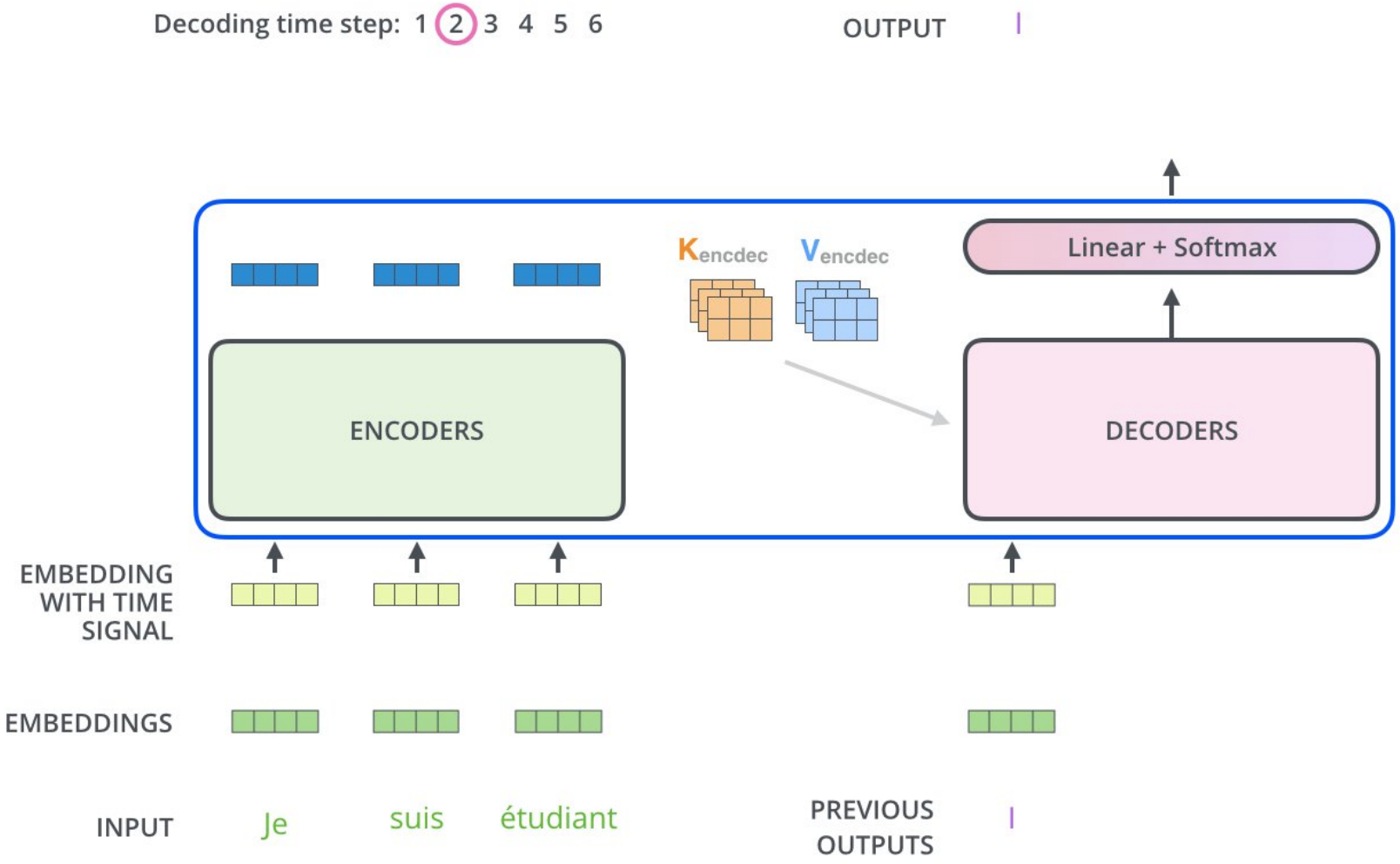


Encoder-decoder in action 1/2



After finishing the encoding phase, we begin the decoding phase. Each step in the decoding phase outputs an element from the output sequence (the English translation sentence in this case).

Encoder-decoder in action 2/2

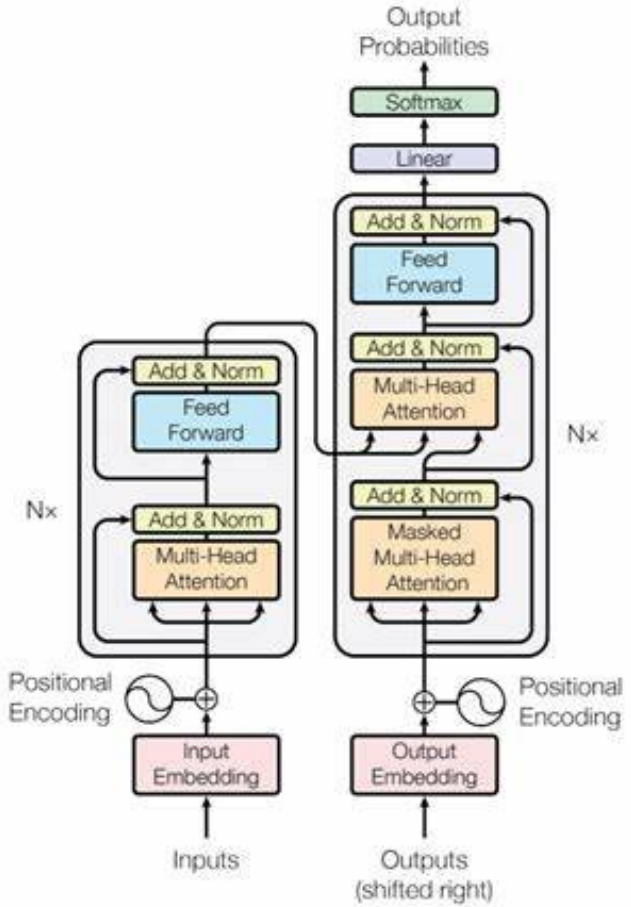


The steps repeat until a special symbol indicating the end of output is generated. The output of each step is fed to the bottom decoder in the next time step. We add positional encoding to decoder inputs to indicate the position of each word.

Animated workings of transformer

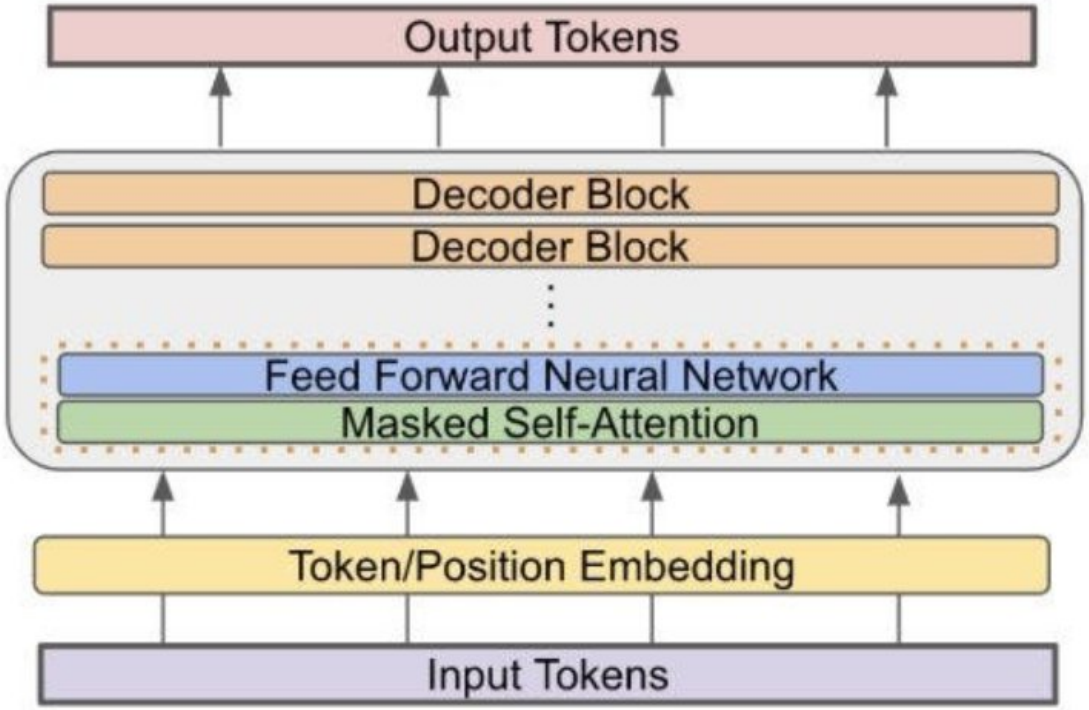
Sodobni veliki jezikovni modeli

- Arhitektura nevronskih mrež transformer
- Učenje v dveh fazah
- 1. faza: prednaučeni modeli
  - napoved maskirane besede, naslednje besed
  - ogromno besedila
  - zelo dolgo učenje
- 2. faza: model doučimo za specifično nalogo
  - veliko hitreje kot 1. faza
  - prenos splošnega znanja o jeziku iz 1. faze
- Na voljo so številni prostodostopni VJM, tudi večjezični in za slovenščino



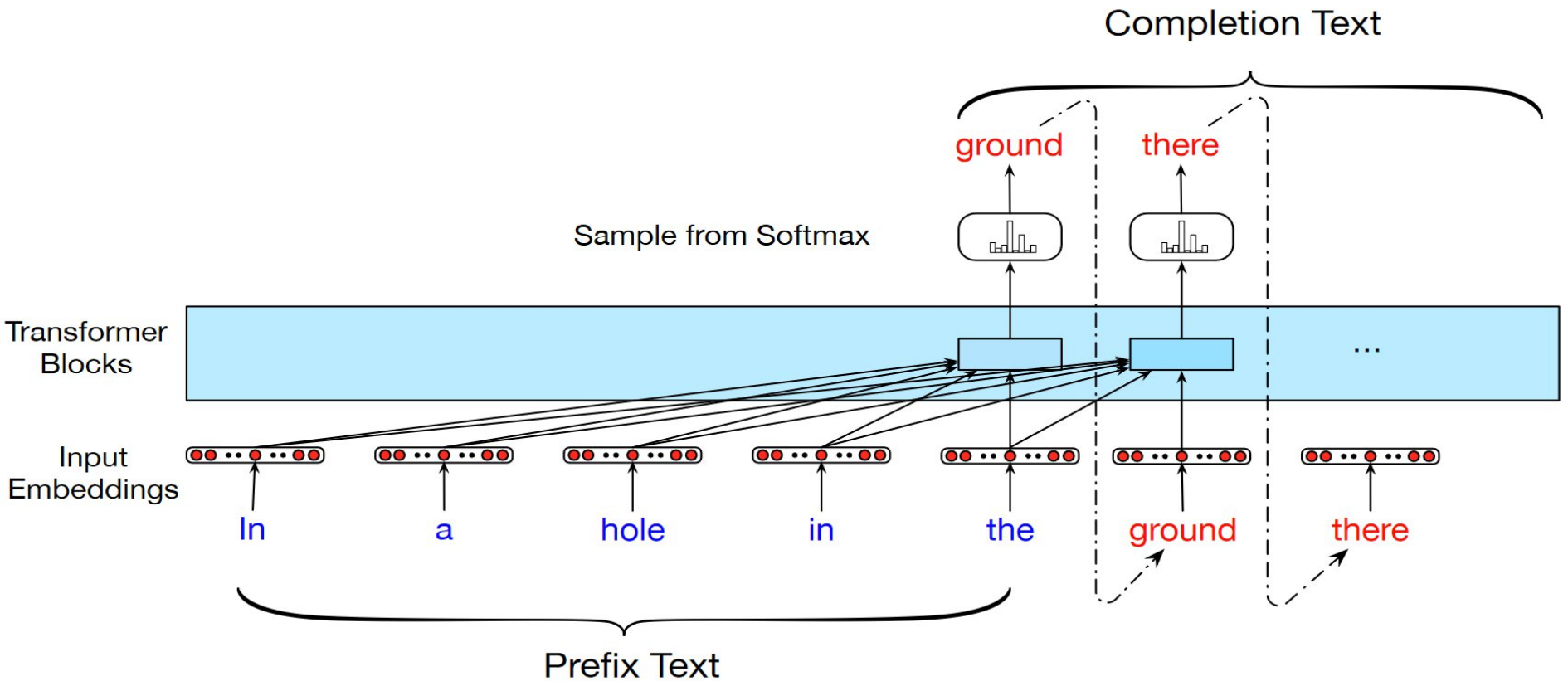
Generativni modeli

- arhitektura transformer: samo dekoder
- GPT, GPT-2, GPT-3, ChatGPT, GPT-4
- LLaMA, LLaMA-2
- MPT, Falcon
- Mistral
- OPT, Bloom
- GaMS



Avtoregresivni generatorji

- generator usmerimo z vhodnim kontekstom, npr. vprašanjem



šanja,

povzemanje in druge generativne naloge

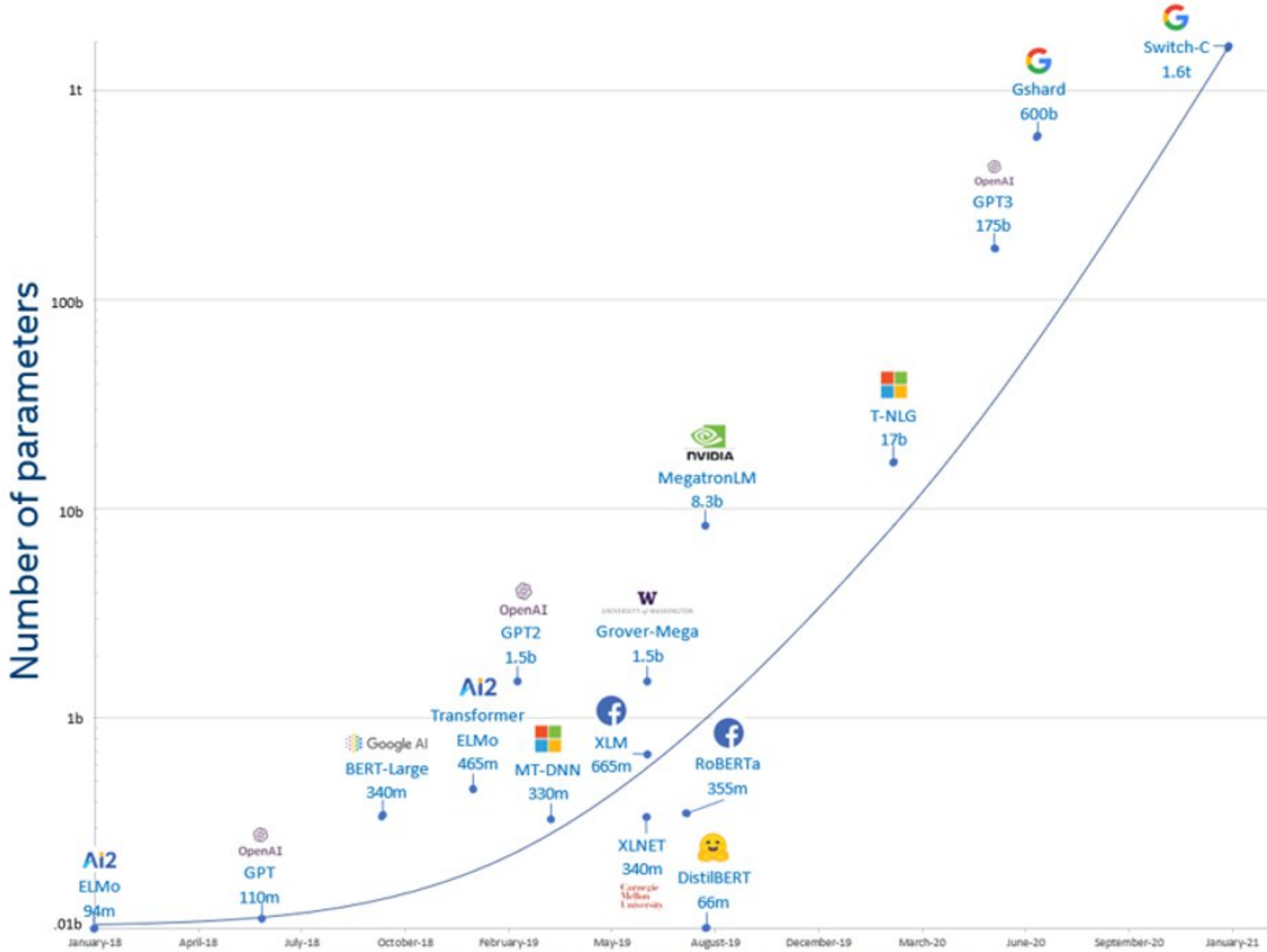


Figure 1: Exponential growth of number of parameters in DL models

GPT-2 and GPT-3

- layer norm applied to input of each subblock
- GPT-3 also uses some sparse attention layers
- more data, larger batch sizes (GPT-3 uses batch size of 3.2M)
- the models are scaled:

GPT-2:
48 layers, 25 heads
d_m = 1600, d = 64
context size = 1024
~ 1.5B parameters

GPT-3:
96 layers, 96 heads
d_m = 12288, d = 128
context size = 2048
~ 175B parameters

[1] Radford et al.: Language Models are Unsupervised Multitask Learners, 2019.
[2] Brown et al.: Language Models are Few-Shot Learners, 2020.

Veliki generativni jezikovni modeli

- OpenAI: ChatGPT november 2022; GPT-4 2023
  nevronska mreža s 175 milijardami parametrov oz.
  8 x 220 milijard parametrov
- dostop na https://chatgpt.com/
- na podlagi GPT-3.5 z dodatnim učenjem za dialog in sledenje ukazom
- uporablja RLHF (spodbujevano učenje s človeškimi povratnimi informacijami)
- Ideja: dodaten napovedni model, ki se nauči rangirati kakovost odgovorov. Ta model dodatno nadzoruje učenje LLM pri dialogih
- velik vpliv na javnost,
- potencialno velike spremembe za vse poklice, ki uporabljajo jezik, tudi za učenje, študij, poučevanje, znanstveno pisanje, intelektualne poklice
- LLaMa, Alpaca, Koala, LiMa: manjši, odprti in zelo konkurenčni modeli

Učenje VJM je podatkovno izjemno požrešno

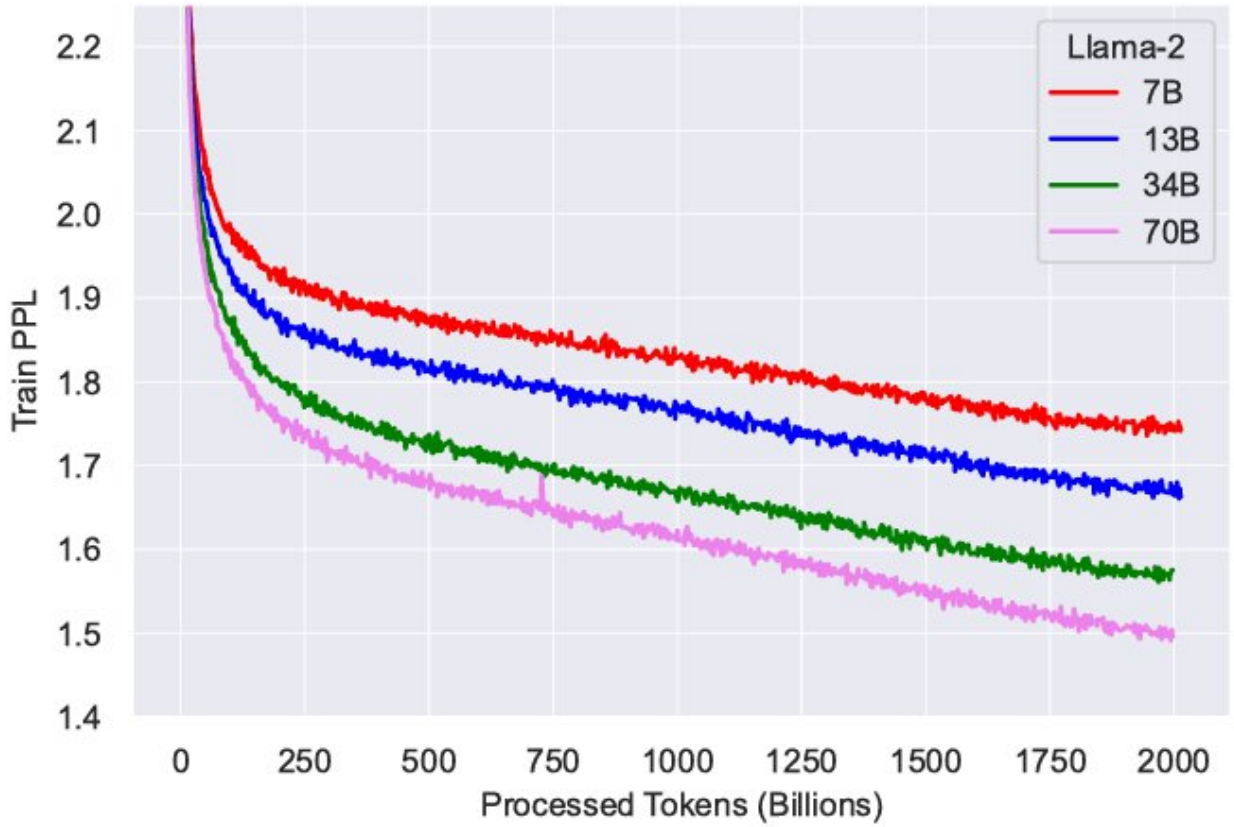


Figure 5: Training Loss for Llama 2 models. We compare the training loss of the Llama 2 family of models. We observe that after pretraining on 2T Tokens, the models still did not show any sign of saturation.

Učenje GaMS 1B

HPC Vega – 60 GPU vozlišč (vsak 4 NVIDIA A100 GPUs s 40 GB RAM na vozlišče)
GaMs 1B na podalgi OPT 1B
 uporaba 6 vozlišč naenkrat s tensor parallel rank 4 (model porazdeljen preko 1 vozlišča)
NeMo framework
Ena epoha: 16 ur
10 različnih verzij (različni tokenizatorji, prenos vložitev, test števila epoh, različni podatki)
solidni rezultati prilagajanja, premajhen model za uporabo v pogovoru
zbiranje slovenskih besedil in preizkus modela: https://povejmo.si/
v pripravi: GaMS 10B (na podlagi Gemma 2 9B)

HPC lekcije

UL FRI

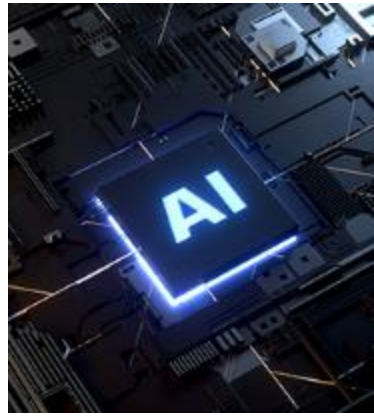
- Izgradnja VJM od začetka je računsko izjemno draga
  - LLaMa-2 7B > 360,000 GPU ur (cca 1 leto Vege)
- Izgradnja z nadaljevanjem učenja na obstoječem VJM
  - količina podatkov (spletni viri, zbiranje slovenskih virov, strojno prevajanje): Hoffmanovi zakoni skaliranja
  - kakovost podatkov
  - izbor slovarja za model
  - paralelizacija modela (znotraj 1 GPU vozlišča, i.e. 4 or 8 A/H100),
  - podatkovna paralelizacija (preko vozlišč), zmanjšanje komunikacije med vozlišči
  - 1B model GaMS uporablja predstavitev bf16 (1b sign, 8b exponent, 7b significand)
  - za 10 B model Vega ni dovolj: premalo pomnilnika, ne podpira fp8
  - 10B model GaMS bo učen na EuroHPC JU Leonardo z A100 64GB
  - prilagajanje specifičnim nalogam, QLORA: NF4, 1 vozlišče, potrebujemo učne množice za sledenje navodilom
  - za izgradnjo VJM potrebujemo precej HPC znanja
- Uporaba platform kot je NVidia NeMo:
  - +hitrost
  - +lažja paralelizacija
  - -na voljo je manj modelov
  - -zahteva specifično prilagajanje modelov in parametrov
- Nasvet: naj gre vse na eno računsko vozlišče (če je mogoče)

Zaključki

UL | FRI

- VJM ponujajo nove možnosti pridobivanja informacij iz slik, zvoka, besedil, sekvenc, itd.
- mnoge koristne aplikacije generativne UI
- številne nove metodološke, etične in družbene dileme
- potreba po pripravi slovenskih virov in specifičnega znanja

- HPC je nujno potrebna infrastruktura za UI
- transformerji so povsod: besedila, slike, govor, časovne vrste
- potreba po enostavnejšem dostopu do HPC
- potreba po HPC izobraževanju v UI in računalništvu
- potreba o razumevanju UI in VJM v HPC izobraževanju



UL FRI VJM ekipa

- Domen Vreš
- Iztok Lebar Bajec
- Tjaša Arčon
- Gašper Jelovčan

- Marko.RobnikSikonja@fri.uni-lj.si

Financiranje

Projekt EuroCC 2 financira Evropska unija. Financiran je s sredstvi Skupnega podjetja za evropsko visokozmogljivo računalništvo (EuroHPC JU) ter Nemčije, Bolgarije, Avstrije, Hrvaške, Cipra, Češke republike, Danske, Estonije, Finske, Grčije, Madžarske, Irske, Italije, Litve, Latvije, Poljske, Portugalske, Romunije, Slovenije, Španije, Švedske, Francije, Nizozemske, Belgije, Luksemburga, Slovaške, Norveške, Turčije, Republike Severne Makedonije, Islandije, Črne gore in Srbije v okviru sporazuma o dodelitvi sredstev št. 101101903.

Delovanje Nacionalnega kompetenčnega centra SLING sofinancira Ministrstvo za visoko šolstvo, znanost in inovacije.

Medijski sponzor

Računalniške novice
www.racunalniske-novice.com