



AKADEMIA E FORCAVE
TË ARMATOSURA

Hyrje në Modele të Mëdha Gjuhësore

Transformerat, arkitektura pas MMGj-ve

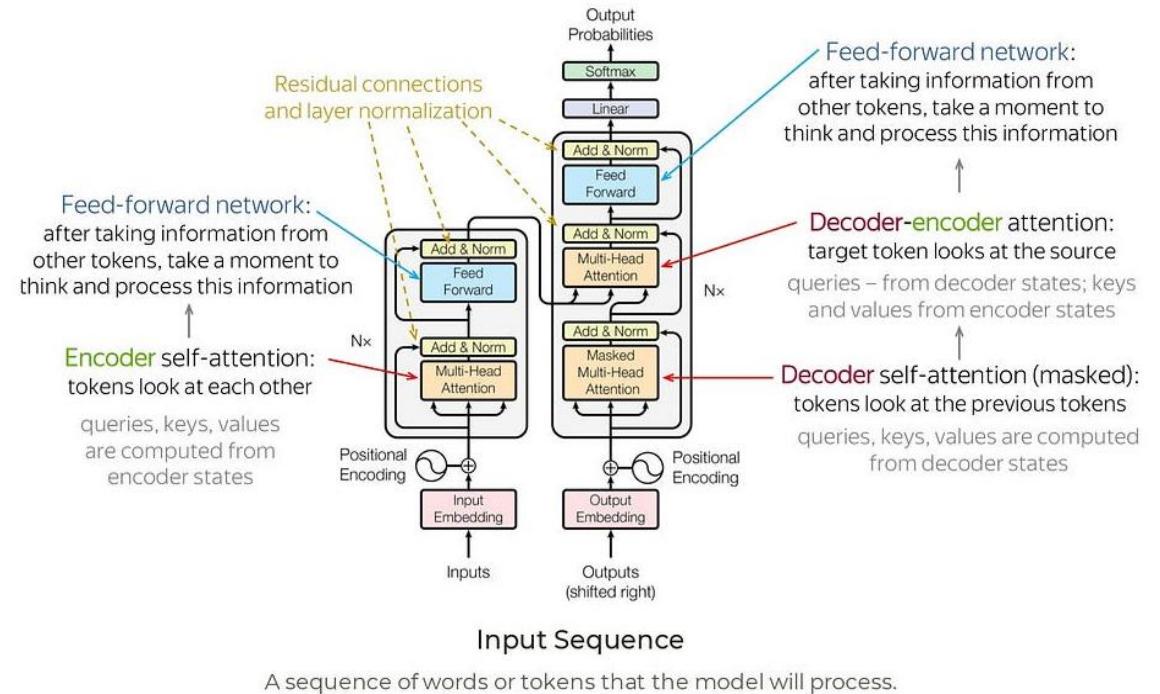
Dr. Fiorela Ciroku



Struktura e brendshme e transformers

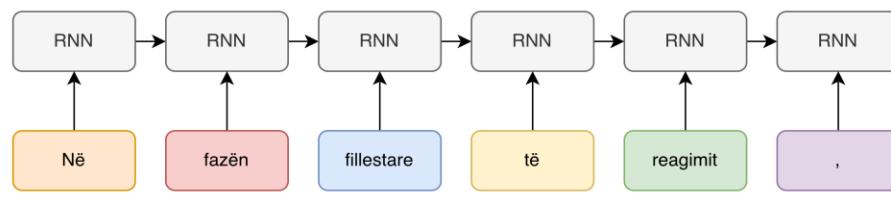
Elementët kryesorë të arkitekturës Transformer:

- Self-attention (vetë-vëmendja)
- Multi-head attention (vëmendja me shumë koka)
- Shtresat feed-forward
- Normalizimi i shtresave
- Lidhjet residuale
- Këto komponentë ndërveprojnë dhe lejojnë modelin që të kuptojë dhe gjenerojë tekst në varësi të konteksit.

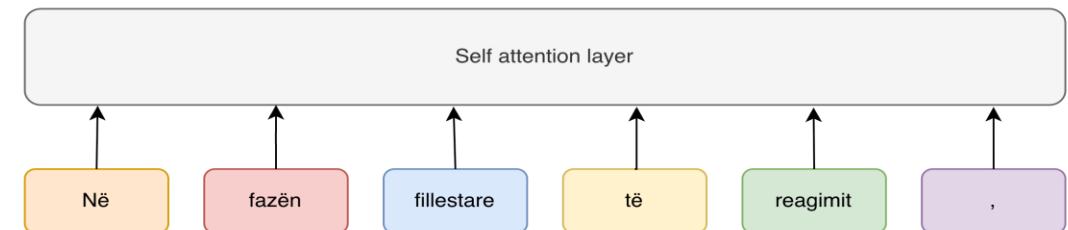


Çfarë problemi zgjidhin transformerat?

- Zëvendëson Recurrent Neural Network (RNN) dhe Long Short-Term Memory (LSTM)
- Eleminon bottlenecks të sekuencave
- Mundëson përpunim paralel
- Shkallëzohet deri në miliarda parametra



Arkitektura RNN

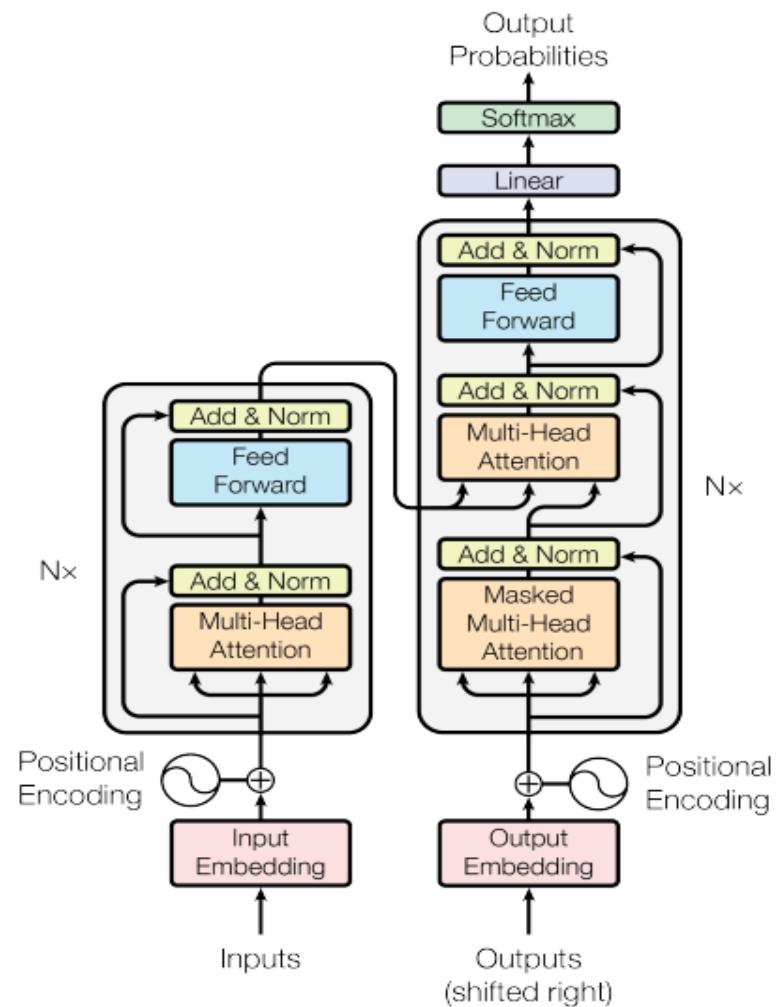


Arkitektura MMGj



Çfarë është vetë-vemendja?

- Vetë-vëmendja (self-attention) llogarit ngjashmérinë midis çdo çifti tokeni në një fjali.
- Çdo token “pyet”: **“Cilët token janë të rëndësishëm për mua për të kuptuar këtë kontekst?”**
- Kjo shtresë prodhon peshat e vëmendjes, të cilat përcaktojnë se sa kontribuon secili tokennë përfaqësimin e shtresës së ardhshme.
- Është mekanizëm për të matur rëndësinë midis tokeneve.
- Ndihamon modelin të përqendrohet te fjalët e rëndësishme.



Query, Key, dhe Value (Q/K/V)

Vetë-vëmendja transformon embeddings në tre vektorë:

- **Query (Q):** Çfarë kërkon ky token?
- **Key (K):** Çfarë ofrojnë tokenat e tjerë?
- **Value (V):** Çfarë kontribuon ky token në kuptim?

$$\text{Attention} = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d}}\right) \times V$$

Diagram illustrating the components of the Attention formula:

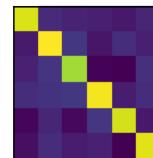
- Attention = softmax**: Converts probabilities.
- $\left(\frac{Q \cdot K^T}{\sqrt{d}}\right)$** : Dimension of vectors.
- $\times V$** : Combines with values.
- Llogaritet ngjashmëria mes tokenave**: Similarity between tokens.

Kjo formulë i tregon modelit se cilët tokenë janë të rëndësishëm për të kuptuar kontekstin.

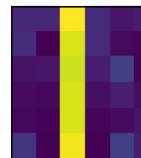
Multi-Head Attention

- Një “attention head” i vetëm mund të përqendrohet te lidhjet midis subjektit dhe foljes.
- Një “head” tjeter mund të kapë vendndodhjen dhe veprimin.
- Një tjetër kohën dhe kufizimin.
- Duke i grumbulluar (stackuar) disa “heads” së bashku, modeli mëson disa patterns gjuhësore njëkohësisht, duke formuar një botëkuptim shumë të pasur.

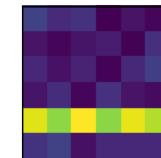
Head 1
subject-verb



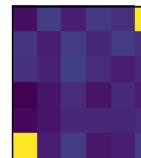
Head 2
location-action



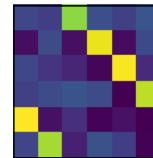
Head 3
time-constraint



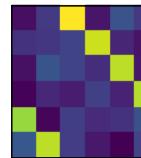
Head 4
long-distance



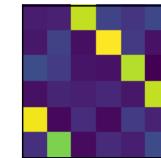
Head 5
mixed pattern



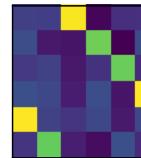
Head 6
mixed pattern



Head 7
mixed pattern



Head 8
mixed pattern

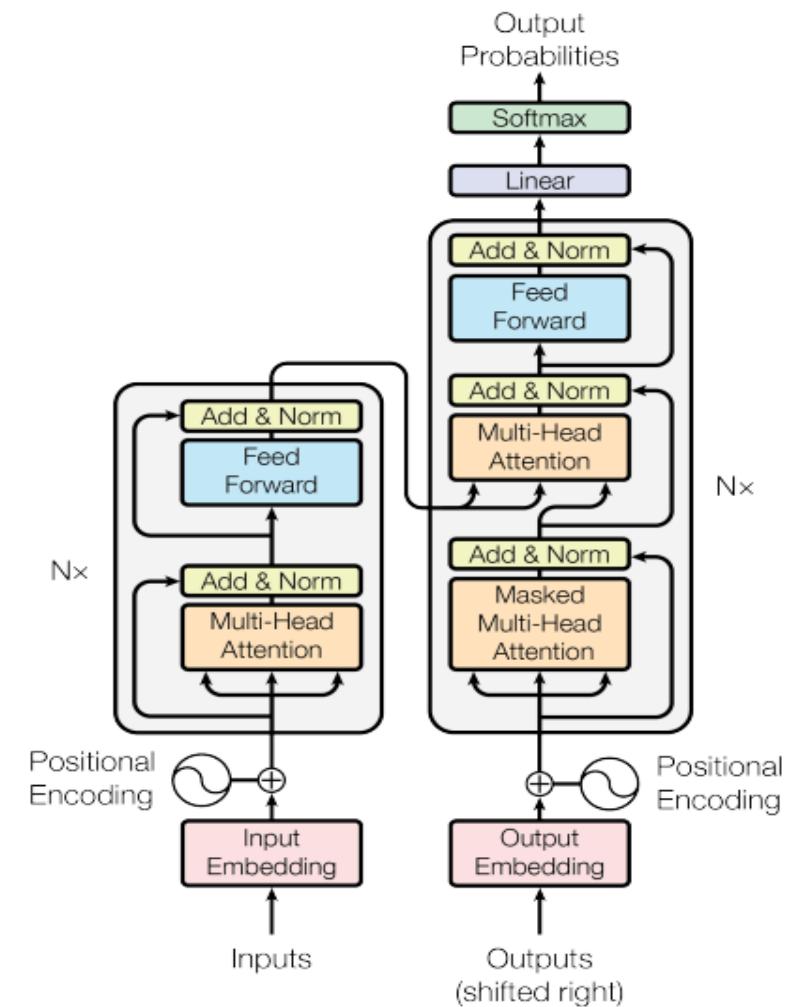


Positional Encoding

- Transformer-at nuk kanë një ndjesi të brendshme të rendit, sepse përpunojnë të gjithë tokenët paralelisht.
- Kodimi pozicional (positional encoding) ofron informacionin që mungon.

*Zjarri u përhap shpejt.
Shpejt, u përhap zjarri.*

- Të njëjtat fjalë, por kuptim i ndryshëm.
- Kodimi pozicional i ndihmojnë modelet të interpretojnë renditjen e fjalëve në mënyrë korrekte.



Feed-Forward Networks

- Pas vetë-vëmendjes (self-attention), modeli ka vendosur se cilët tokenë janë të rëndësishëm për cilët tokenë të tjerë. Megjithatë, vëmendja **nuk e transformon thellësisht kuptimin**.
- **Rrjeti feed-forward (FFN)** kryen transformimin real të përfaqësimit të token.
- FFN aplikohet në mënyrë të pavarur për çdo token.
- Tokenët **nuk ndërveprojnë** me njëri-tjetrin në këtë fazë. I gjithë ndërveprimi ka ndodhur gjatë vëmendjes.

Në mënyrë konceptuale:

- **Vëmendja** përgjigjet: “Kë duhet të dëgjoj?”
- **Feed-forward** përgjigjet: “Si duhet ta riinterpretoj atë që mësova?”



Feed-Forward Networks

- Çdo përfaqësim tokeni kalon nëpër të njëtin rrjet nervor të vogël:
Linear → ReLU/GELU/SwiGLU→ Linear
- ReLU = Rectified Linear Unit; GELU = Gaussian Error Linear Unit; SwiGLU = Swish-Gated Linear Unit



Pse 2 shtresa lineare?

- Një shtresë është thjeshtë një transformim linear, nuk modelon pattern komplekse.
- Duke shtuar dimensione te fshehura dhe shtresen jo-lineare, rrjeti fiton aftësinë të ndajë dhe kombinojë feature, të mësojë transformime më të pasura, të zbulojë pattern abstrakte/komplekse, të enkodojë koncepte të gradave të larta.



AI Mbrojtja Civile

Në dokumentet e Mbrojtjes Civile, kjo mundëson transformime si:

- shndërrimin e termit “përmbytjes” në një koncept abstrakt si "**rrezik hidrometeorologjik**";
- kombinimin e **vendndodhjes + veprimit + urgjencës** në një vektor semantik më të pasur.

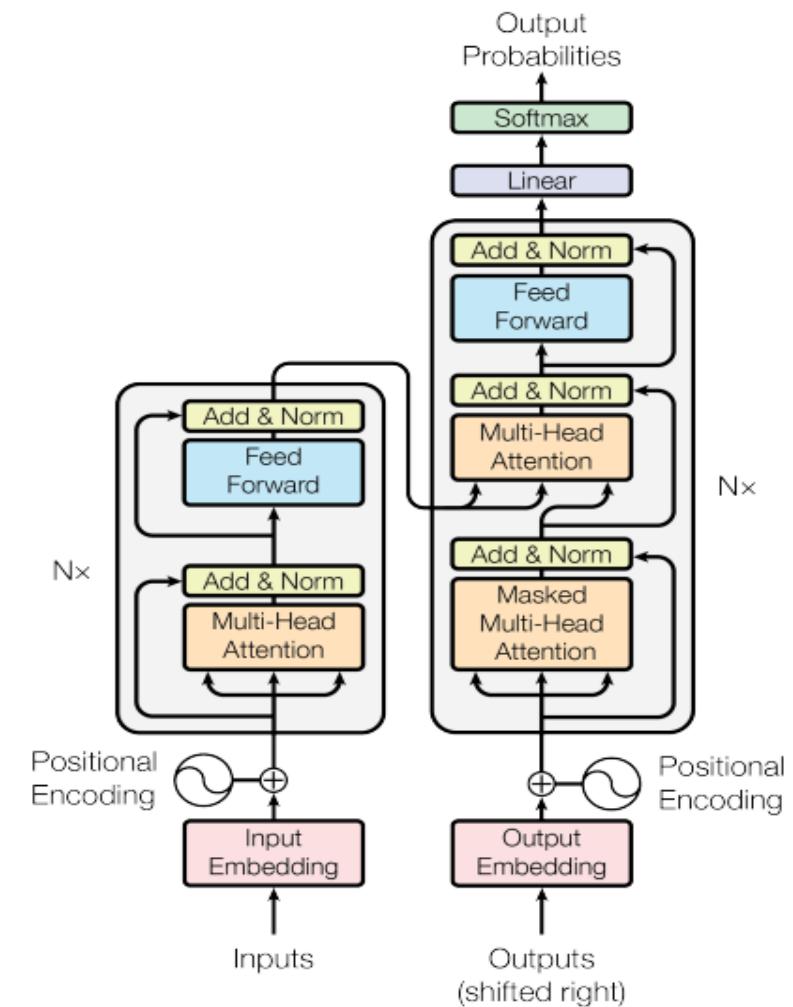
Kjo shtresë kontribuon rendësishëm në fuqinë shprehëse të modelit.

Shtresa e normalizimit

- Ndërsa modelet bëhen më të thella, paqëndrueshmëria numerike shndërrohet në një problem serioz.
- Pa normalizim, aktivizimet mund të shpërthejnë dhe trajnimi bëhet i paqëndrueshëm ose dështon.

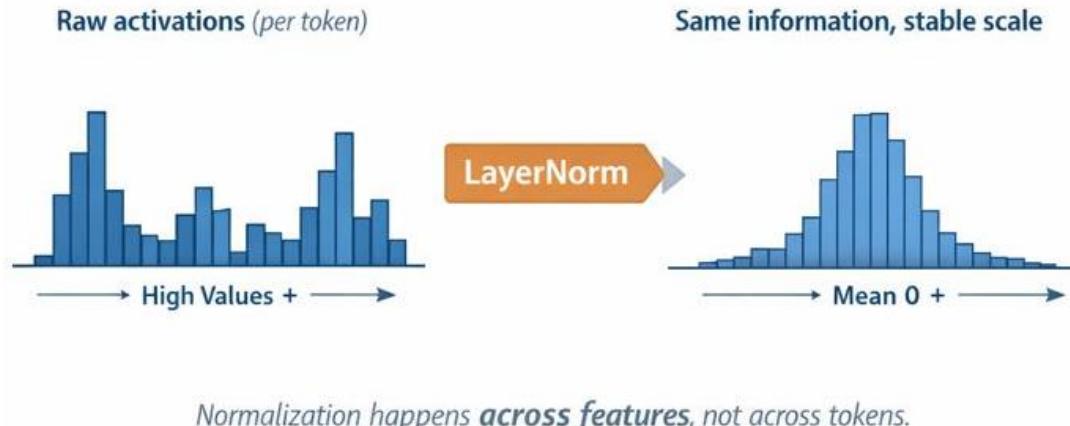
Normalizimi i shtresës e zgjidh këtë duke:

- normalizuar aktivizimet përgjatë veçorive,
- mbajtur vlerat brenda një intervali të qëndrueshëm,
- e bërë trajnimin të qëndrueshëm në të gjitha shtresat.



Shtresa e normalizimit

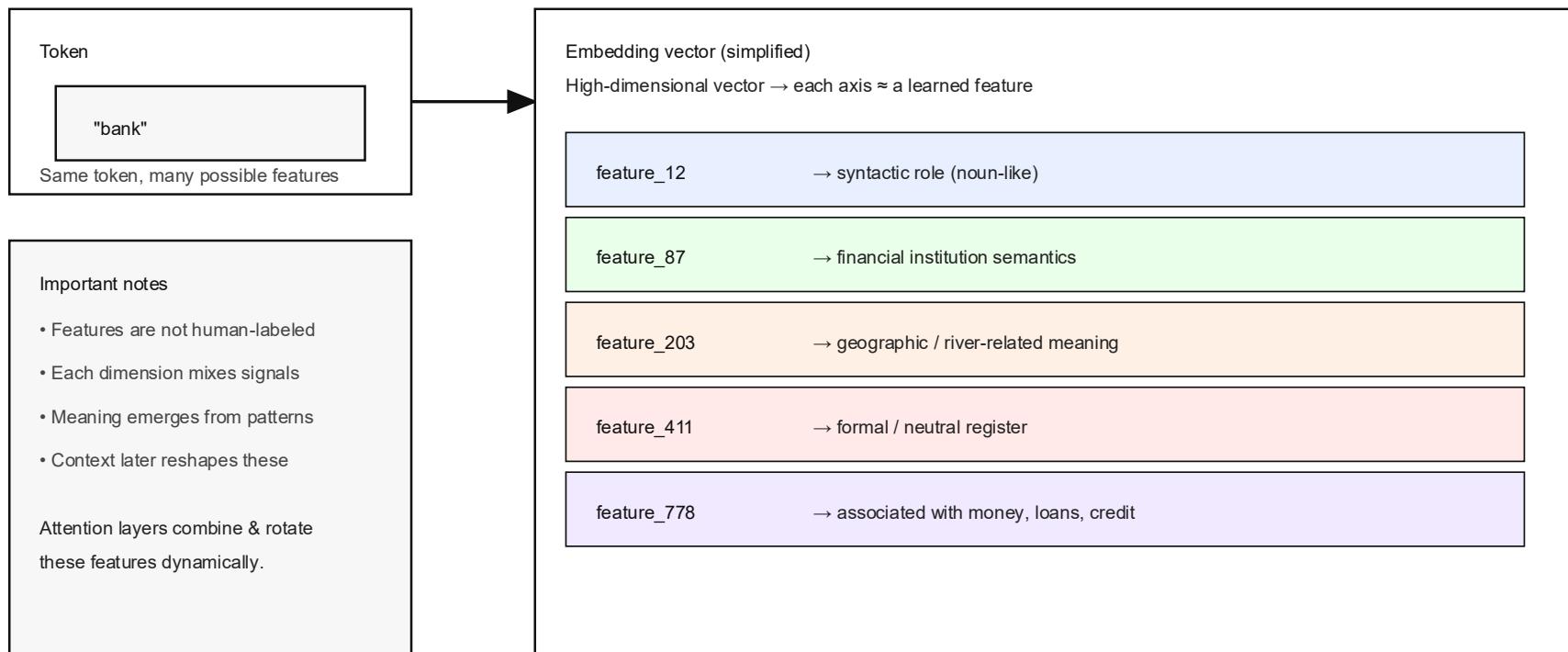
- LayerNorm aplikohet brenda çdo blloku Transformer
- Zakonisht aplikohet dy herë: pas vëmendjes (attention) dhe pas rrjetit feed-forward.
- Mundëson transformer-a me dhjetëra ose qindra shtresa.
- **Pa shtresen e normalizimit, MMGj-të moderne thjesht nuk do të mund të trajnoheshin.**



Embedding features

LLM token feature embeddings — examples

Each embedding dimension captures a latent feature learned from usage patterns

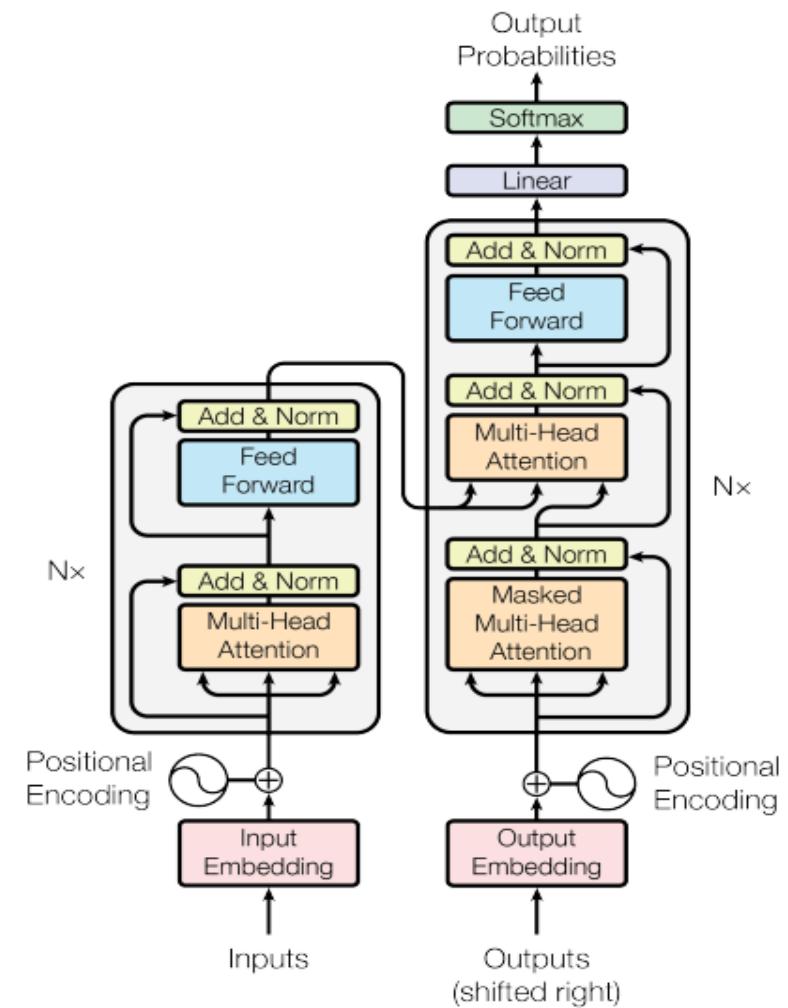


Residual Connections

- **Lidhjet residuale (residual connections)** e parandalojnë modelin që të harrojë atë që tashmë “di”.
- Në vend që të zëvendësohet inputi, çdo nën-shtresë i shton asaj rezultatin e vet.

$$\begin{aligned}x_1 &= \text{LayerNorm}(x + \text{Attention}(x)) \\x_2 &= \text{LayerNorm}(x_1 + \text{FFN}(x_1))\end{aligned}$$

- Kjo është e rëndësishme sepse ruhet informacioni nga shtresat e mëparshme dhe arkitektura shumë të thella bëhen të trajnueshme.



Residual Connections

- **Kjo shtresë është thelbësore sepse:**
- Transformer-at grumbullojnë shumë shtresa.
- Çdo shtresë rafinon pak nga pak kuptimin.
- Lidhjet residue sigurojnë që kuptimet semantike të hershme të mos fshihen.

Shembull nga Mbrojtja Civile:

- Një token si “**tërmet**” ruan kuptimin e tij bazë.
- Edhe pas dhjetëra shtresash, ai mbetet i identifikueshëm.
- Shtresat e mëvonshme vetëm e rafinojnë kuptimin, nuk e mbishkruajnë atë.

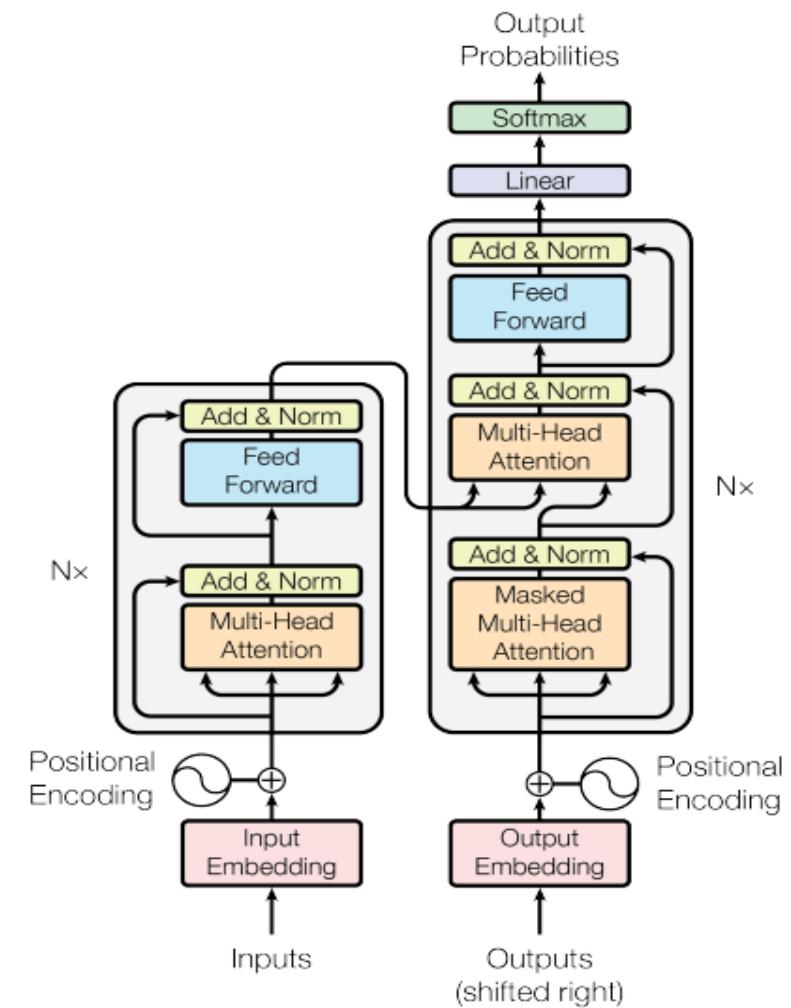


Blloku i transformerit

Një bllok i vetëm transformer përbëhet nga:

- vetë-vëmendja (self-attention)
- lidhje residuale
- normalizim i shtresës (layer normalization)
- rrjet feed-forward
- lidhje residuale
- normalizim i shtresës

Transformer-at nuk e kuptojnë tekstin në një hap të vetëm. Kuptimi shfaqet gradualisht përgjatë shtresave.



Blloku i transformerit

Ky bllok përsëritet shumë herë:

- 12 shtresa (modele të vogla)
- 24–32 shtresa (modele të mesme)
- 70–100+ shtresa (LLM të mëdha)

Çdo shtresë:

- rafinon përfaqësimet,
- ndërton kuptim më abstrakt,
- integron kontekst më të gjerë.



Encoder vs Decoder vs Encoder-Decoder

Arkitektura e transformer është fleksibile. Ekzistojnë tre familje kryesore:

- **Modele vetëm me encoder:** Kuptojnë tekstin (*klasifikim, kërkim, retrieval*), shembull *BERT*.
- **Modele vetëm me decoder:** Gjenerojnë tekst, shembull *GPT, LLaMA, Mistral*.
- **Modele encoder-decoder:** Transformojnë tekstin (*përkthim, përmbledhje*), shembull *T5, FLAN*.

MMGj-të janë modele vetëm me decoder sepse:

- *gjenerimi* është *autoregresiv*
- *çdo token parashikon tokenin pasues*
- *vëmendja (attention)* është e maskuar për të mos “parë” të ardhmen



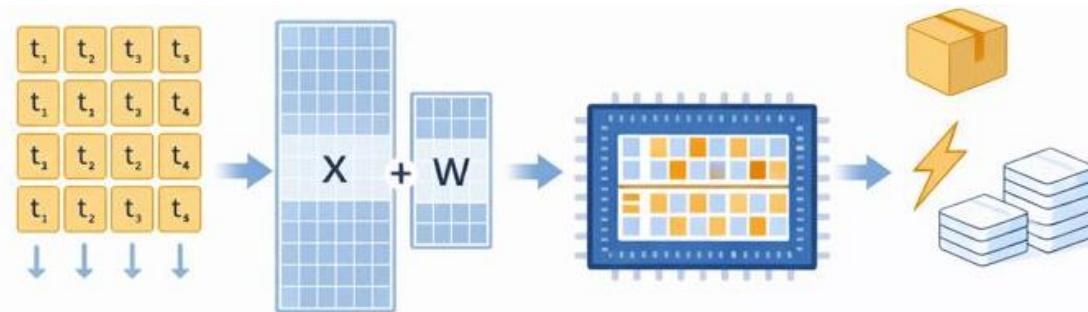
Pse transformers shkallëzohen mirë?

Transformer-at shkallëzohen (scale) mirë për **arsye inxhinierike**, jo për “magji”.

Arsyet kryesore:

- Të gjithë tokenët përpunohen paralelisht.
- Kompjutimi reduktohet në shumëzime matricash.
- GPU-të dhe TPU-të janë të optimizuara për këtë proces.

Kjo mundëson batching shumë të mëdha, përdorim efikas të harduerit dhe trajnim mbi triliona tokenave.



Si mundësohet arsyetimi?

- Ndryshe nga RNN-të, **vëmendja (attention) nuk degradohet me distancën.**
- Një token mund t'i kushtojë vëmendje fjalës së mëparshme, një fjalie shumë më herët në tekstu, apo një paragrafi qindra token më larg.
- Prandaj transformer-at shkëlqejnë në pyetje përgjigje mbi dokumente (document QA), arsyetim ligjor dhe kuptim procedurash.
- Kjo është thelbësore për dokumente të gjata.



AI Mbrojtja Civile

Kur AI i përgjigjet një pyetjeje për zonat e evakuimit, informacioni përkatës mund të jetë i shpërndarë në disa pjesë të ndryshme të një dokumenti të gjatë.

Për shembull:

- rregullat e evakuimit në seksionin 3
- përgjegjësitë në seksionin 7
- përjashtimet në një shtojcë

Vëmendja i lejon modelit t'i lidhë këto automatikisht.



Diagnostikimi i sjelljes së transformerit

- Transformer-at janë shumë të fuqishëm, por **jo transparentë**.
- Studiuesit i diagnostikojnë ato duke përdorur harta vëmendjeje (attention maps), analizë të aktivizimeve (activation probing), atribuim të bazuar në gradientë (gradient-based attribution).
- Këto mjete na ndihmojnë të kuptojmë mënyrat e dështimit, zbulojmë hallucinacione dhe të dizenjojmë prompt-e më të sigurta.
- **Vetë prompt engineering bëhet një mjet diagnostikues pasi duke ndryshuar udhëzimet**, mund të vëzhgojmë zhvendosjet në vëmendje dhe të testojmë qëndrueshmërinë e arsyetimit.



Çfarë do të trajtojmë në vazhdimësi?

-  Hyrje
-  Tokenizimi & Embeddings
-  Arkitektura Transformer
- 4. Bazat e të dhënave vektoriale (Vector Databases)
- 5. RAG (Gjenerim i përforcuar nga kërkimi)
- 6. Promptimi
- 7. Agjentët (Agents)
- 8. Përshtatja e modelit (Fine-Tuning)
- 9. Vlerësimi (Evaluation)
- 10. Siguria & Përafrimi (Safety & Alignment)
- 11. Vendosja e modelit (Deployment)
- 12. Integrimi i Projektit Final



Bibliografia

1. J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.
2. I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The long-document transformer,” *arXiv preprint arXiv:2004.05150*, 2020.
3. T. B. Brown *et al.*, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.
4. K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, “What does BERT look at? An analysis of BERT’s attention,” in *Proc. ACL*, 2019, pp. 276–286.
5. T. Dao *et al.*, “FlashAttention: Fast and memory-efficient exact attention with IO-awareness,” in *Advances in Neural Information Processing Systems*, 2022.
6. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
7. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, 2016, pp. 770–778.
8. J. Kaplan *et al.*, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.
9. Z. Lin *et al.*, “A structured self-attentive sentence embedding,” in *Proc. ICLR*, 2017.
10. C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
11. P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” in *Proc. NAACL-HLT*, 2018, pp. 464–468.
12. A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.
13. J. Vig, “A multiscale visualization of attention in the transformer model,” in *Proc. ACL Workshop on BlackboxNLP*, 2019, pp. 37–42.

