Meeting 2: Arsitektur Transformer — Self-Attention

AI-40X: Generative AI & Large Language Models

Hendri Karisma, M.T.
Dosen Teknik Informatika STMIK Tazkia
VP Engineering at jejakin.com, 2026

Semester 4 — 2025/2026

Outline

Recap: Masalah dengan Arsitektur Recurrent

Dari pertemuan sebelumnya, kita tahu:

- RNN/LSTM memproses sekuens **satu token per satu waktu**: $h_1 \to h_2 \to \ldots \to h_T$.
- Ini berarti: h_t **bergantung** pada h_{t-1} — tidak bisa dihitung paralel.
- Dengan GPU modern yang memiliki ribuan core, arsitektur sekuensial = **pemborosan resource**.

Pertanyaan Vaswani et al. (2017):

> *"Bisakah kita menangkap relasi antar kata* **tanpa** *rekurensi sama sekali?"*

Jawaban: Ya — cukup gunakan **Attention saja**. → *"Attention Is All You Need"*

Recurrence-Free Architecture: Keuntungan Transformer

RNN/LSTM:

- Komputasi: $\mathcal{O}(T)$ langkah sekuensial
- Path length antar kata jauh: $\mathcal{O}(T)$
- Paralelisasi: **Tidak bisa**
- Training kalimat 1000 token: lambat

Transformer:

- Komputasi: $\mathcal{O}(1)$ langkah (semua posisi paralel)
- Path length antar kata jauh: $\mathcal{O}(1)$
- Paralelisasi: **Penuh di GPU**
- Training kalimat 1000 token: jauh lebih cepat

Trade-off: Kompleksitas memori $\mathcal{O}(T^2)$ karena setiap token harus "melihat" semua token lain (attention matrix berukuran $T \times T$).

Ini menjadi tantangan untuk konteks yang sangat panjang — topik yang akan kita bahas di pertemuan mendatang.

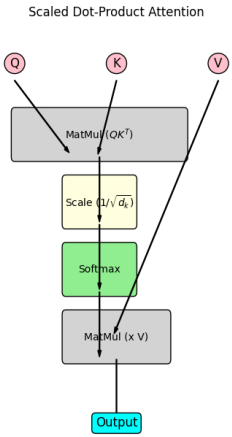
Query, Key, Value — Analogi Database

Analogi: Bayangkan Anda mencari buku di perpustakaan.

- **Query (Q):** Pertanyaan Anda — "Saya cari buku tentang AI."
- **Key (K):** Label/judul setiap buku di rak.
- **Value (V):** Isi buku yang sebenarnya.

Proses:

1. Cocokkan Query dengan setiap Key (hitung relevansi).
2. Buku yang judulnya paling cocok mendapat bobot tinggi.
3. Ambil isi (Value) dari buku-buku yang relevan, dicampur berdasarkan bobot.



Rumus: Scaled Dot-Product Attention

Formula

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Dimensi matriks (untuk sekuens panjang T, dimensi model d_k):

- $Q \in \mathbb{R}^{T \times d_k}$, $K \in \mathbb{R}^{T \times d_k}$, $V \in \mathbb{R}^{T \times d_v}$
- $QK^T \in \mathbb{R}^{T \times T}$ — matriks attention scores
- Setelah softmax: $\mathbb{R}^{T \times T}$ — matriks attention weights (setiap baris berjumlah 1)
- Output: $\mathbb{R}^{T \times d_v}$ — representasi baru yang sudah "sadar konteks"

Langkah demi Langkah: Membedah Formula

Langkah 1: Hitung QK^T (Dot Product)

- Setiap elemen (i, j) dari QK^T = dot product antara q_i dan k_j.
- Dot product mengukur **kemiripan arah** dua vektor.
- Skor tinggi $\rightarrow$ Query i sangat relevan dengan Key j.

Langkah 2: Scaling $\div\sqrt{d_k}$

- Tanpa scaling, dot product bisa menghasilkan nilai sangat besar.
- Softmax pada nilai besar $\rightarrow$ gradien sangat kecil (saturasi).

Langkah 3: Softmax — Mengubah skor menjadi distribusi probabilitas.

Langkah 4: Kalikan dengan V — Weighted sum dari Values.

Mengapa Perlu Scaling $\frac{1}{\sqrt{d_k}}$? (Analisis Variansi)

Bukti intuitif:
Misalkan q_i dan k_j adalah vektor dengan elemen i.i.d. bermean 0 dan variansi 1.

Dot product: $q \cdot k = \sum_{i=1}^{d_k} q_i \cdot k_i$

- $\mathbb{E}[q_i \cdot k_i] = 0$ (mean tetap 0)
- $\text{Var}(q_i \cdot k_i) = \text{Var}(q_i) \cdot \text{Var}(k_i) = 1$
- $\text{Var}(q \cdot k) = \sum_{i=1}^{d_k} \text{Var}(q_i \cdot k_i) = d_k$

Jadi standar deviasi dot product $= \sqrt{d_k}$.

Dengan $d_k = 512$: dot product bisa bernilai ± 22.6, menyebabkan softmax **saturasi**.

Solusi: Bagi dengan $\sqrt{d_k}$ agar variansi kembali $\approx 1 \rightarrow$ softmax bekerja di zona gradien yang baik.

Apa itu Self-Attention?

- Pada Bahdanau Attention: Query dari **decoder**, Key/Value dari **encoder** (dua sumber berbeda).
- Pada **Self-Attention**: Q, K, dan V semuanya berasal dari **sumber yang sama**.
- Setiap token dalam kalimat "bertanya" kepada **semua token lain** (termasuk dirinya sendiri): *"Siapa yang paling relevan untuk memahami makna saya?"*

Secara teknis:

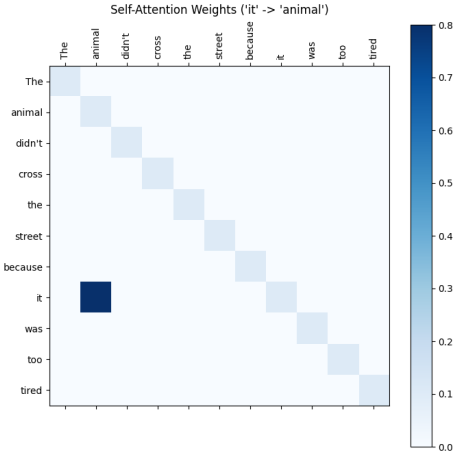
- Input: matriks embedding $X \in \mathbb{R}^{T \times d_{\text{model}}}$
- Proyeksi linear: $Q = XW^Q, \quad K = XW^K, \quad V = XW^V$
- $W^Q, W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$
- Matriks bobot W^Q, W^K, W^V dipelajari saat training.

Contoh: Resolusi Pronomina

"The animal didn't cross the street because **it** was too tired."

- Kata "**it**" merujuk ke kata apa?
- Manusia langsung tahu: "it" = "animal" (karena "tired" cocok dengan makhluk hidup).
- **Self-Attention memungkinkan model melakukan hal yang sama:**
  - Token "it" menjadi Query.
  - Semua token lain ("The", "animal", "didn't", "cross", "street", ...) menjadi Keys.
  - Model belajar memberikan attention weight **tinggi** pada "animal" dan **rendah** pada "street".
- Tanpa self-attention, model hanya melihat posisi lokal — sulit menghubungkan "it" dengan "animal" yang jaraknya 5 token.

Visualisasi Self-Attention Heatmap



Cara membaca heatmap:

- Baris = token sebagai Query.
- Kolom = token sebagai Key.
- Warna cerah = attention weight tinggi.
- Diagonal biasanya cerah (token memperhatikan dirinya sendiri).

Yang perlu diperhatikan:

- Pola *off-diagonal* menunjukkan relasi antar kata yang jauh.
- Setiap baris berjumlah 1 (hasil softmax).
- Pola ini **dipelajari**, bukan di-hardcode.

Mengapa Satu Attention Head Tidak Cukup?

Masalah: Satu set Q, K, V hanya bisa menangkap **satu jenis relasi** pada satu waktu.

Padahal bahasa memiliki **banyak jenis hubungan simultan**:

- **Sintaksis:** subjek-predikat ("kucing *makan* ikan")
- **Koreferensi:** pronomina-anteseden ("Ani... *dia*")
- **Semantik:** relasi makna ("dokter... *rumah sakit*")
- **Posisional:** kata terdekat sering relevan secara gramatikal

Solusi: Jalankan **banyak attention secara paralel**, masing-masing dengan bobot W^Q, W^K, W^V yang berbeda. Setiap "head" belajar menangkap aspek berbeda dari bahasa.

Mekanisme Multi-Head Attention

Formula

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\, W^O$$

di mana setiap head:

$$\text{head}_i = \text{Attention}(QW_i^Q,\ KW_i^K,\ VW_i^V)$$

Detail dimensi (Transformer base: $d_{\text{model}} = 512$, $h = 8$ heads):

- Setiap head: $d_k = d_v = d_{\text{model}}/h = 512/8 = 64$
- $W_i^Q, W_i^K \in \mathbb{R}^{512\times 64}$, $W_i^V \in \mathbb{R}^{512\times 64}$
- Output setiap head: $\mathbb{R}^{T\times 64}$
- Concat 8 heads: $\mathbb{R}^{T\times 512}$
- $W^O \in \mathbb{R}^{512\times 512}$: proyeksi linear akhir

Total parameter per head **lebih sedikit**, *tapi secara kolektif menangkap representasi yang lebih kaya.*

Apa yang Dipelajari Setiap Head?

Riset menunjukkan bahwa head yang berbeda belajar menangkap pola berbeda:

- **Head 1:** Hubungan sintaksis (subjek ↔ verba)
- **Head 2:** Kata terdekat (adjacent tokens)
- **Head 3:** Koreferensi (pronomina ↔ anteseden)
- **Head 4:** Entitas bernama (NER-like)
- **Head 5:** Relasi semantik
- **Head 6:** Posisi relatif
- **Head 7–8:** Pola lebih abstrak/kompleks

Ini terjadi secara otomatis melalui proses training — tidak di-program secara manual!

Analogi: Seperti melihat kalimat dari 8 "sudut pandang" berbeda, lalu menggabungkan semua perspektif menjadi pemahaman yang utuh.

Alur Lengkap: Dari Input ke Output Multi-Head Attention

Langkah-langkah komputasi:

1. **Input:** Embedding $X \in \mathbb{R}^{T \times d_{\text{model}}}$
2. **Proyeksi:** Untuk setiap head $i = 1, \ldots, h$:

$$Q_i = XW_i^Q, \quad K_i = XW_i^K, \quad V_i = XW_i^V$$

3. **Attention per head:**

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

4. **Concatenation:** $\text{Concat}(\text{head}_1, \ldots, \text{head}_h) \in \mathbb{R}^{T \times (h \cdot d_v)}$
5. **Proyeksi akhir:** Kalikan dengan W^O untuk mendapatkan output $\in \mathbb{R}^{T \times d_{\text{model}}}$

Catatan: Semua head dihitung **secara paralel** — ini yang membuat Transformer sangat cepat di GPU!

Kesimpulan Pertemuan 2

1. **Transformer** membuang rekurensi dan hanya mengandalkan mekanisme Attention $\rightarrow$ paralelisasi penuh.
2. **Scaled Dot-Product Attention:** $\text{softmax}(QK^T/\sqrt{d_k})V$ — menghitung relevansi setiap pasangan token.
3. **Scaling** $\frac{1}{\sqrt{d_k}}$ diperlukan untuk mencegah saturasi softmax (variansi dot product $= d_k$).
4. **Self-Attention** memungkinkan setiap token "melihat" seluruh konteks kalimat — menangkap relasi jarak jauh dalam $\mathcal{O}(1)$ langkah.
5. **Multi-Head Attention** menjalankan beberapa attention paralel, masing-masing menangkap aspek bahasa yang berbeda.

Pertemuan berikutnya: Arsitektur Transformer lengkap — Positional Encoding, Feed-Forward, Layer Norm, Encoder-Decoder.

Referensi

- Vaswani, A. et al. (2017). “Attention Is All You Need.” *Advances in Neural Information Processing Systems (NeurIPS)*.
- Alammar, J. (2018). “The Illustrated Transformer.” `https://jalammar.github.io/illustrated-transformer/`
- Bahdanau, D., Cho, K. & Bengio, Y. (2014). “Neural Machine Translation by Jointly Learning to Align and Translate.” *arXiv:1409.0473*.
- Clark, K. et al. (2019). “What Does BERT Look At? An Analysis of BERT’s Attention.” *ACL Workshop BlackboxNLP*.
- Voita, E. et al. (2019). “Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting.” *ACL*.
- Huang, A. (2020). “Attention? Attention!” *Lil’Log*. `https://lilianweng.github.io/posts/2018-06-24-attention/`