Bab 11 — Generative AI dan LLM sebagai Konteks Modern

Cara membaca bab ini

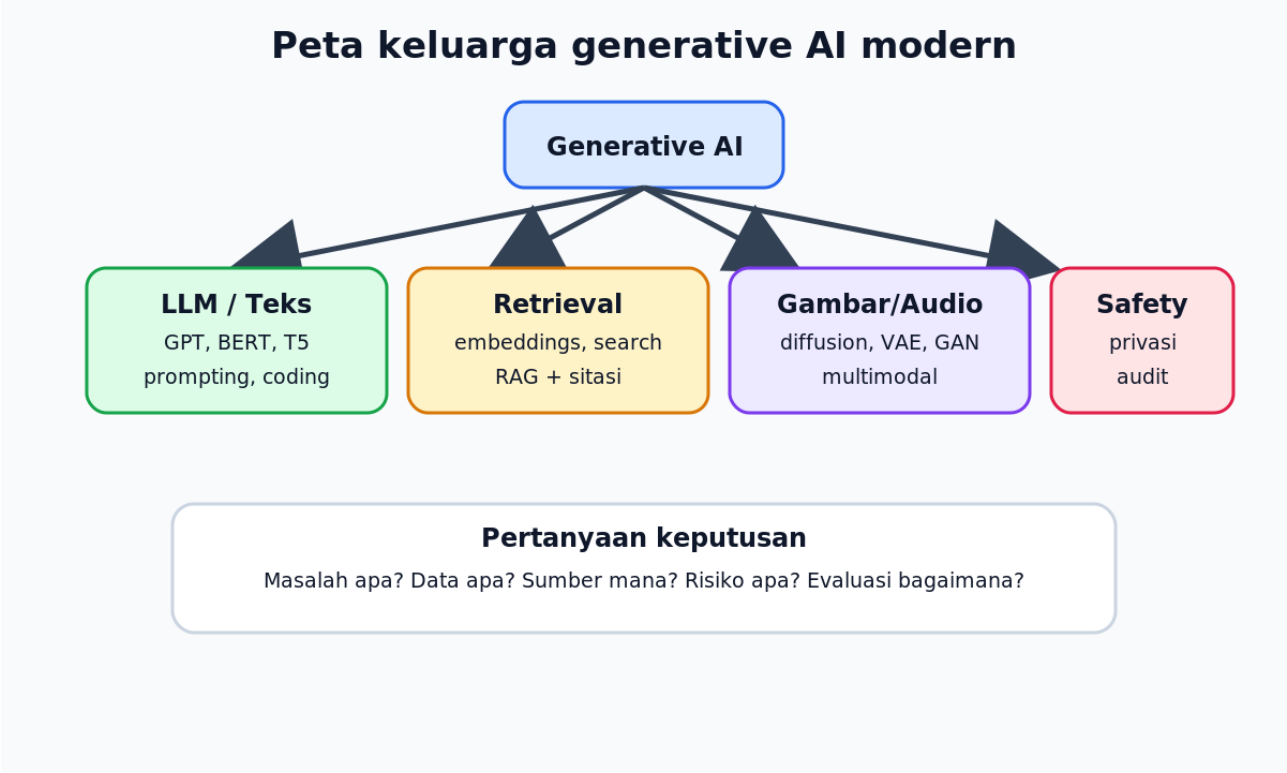
Bab 11 melanjutkan Bab 10. Di Bab 10 kita belajar bahwa Transformer adalah keluarga arsitektur deep learning untuk data urutan. Di bab ini Transformer menjadi pusat ekosistem modern: GPT/ChatGPT, BERT, T5, embeddings, RAG, tool use, multimodal model, dan sebagian besar produk generative AI yang sekarang dipakai di kantor, kampus, UMKM, media, hukum, kesehatan, pendidikan, dan pemerintahan.

Bab ini tetap AI-first. Kita tidak sedang belajar seluruh NLP, seluruh computer vision, atau MLOps produksi. Targetnya: pembaca dapat memahami arsitektur utama generative AI, membaca klaim produk AI dengan kritis, membuat prototipe mini RAG yang dapat dijalankan lokal, dan tahu batas aman sebelum memakai LLM untuk keputusan penting.

Pola belajar setiap subbab:

> cerita masalah → bentuk data → intuisi arsitektur → persamaan kecil → contoh hitung → praktik → tes cepat

Jika istilah seperti *self-attention*, *decoder-only*, *embedding*, atau *diffusion* terasa asing, jangan panik. Kita akan memulai dari analogi visual: perpustakaan, loket pelayanan, tim ahli, dan kamera yang menghapus noise sedikit demi sedikit.



Peta keluarga generative AI

Subbab 1 — Mengapa generative AI meledak?

Inti subbab: generative AI berguna karena model tidak hanya mengklasifikasikan data, tetapi membuat konten baru yang masuk akal berdasarkan pola yang dipelajari.

Bayangkan pemilik UMKM di Bandung ingin membuat deskripsi produk, membalas chat pelanggan, merangkum ulasan, dan membuat ide foto katalog. Sebelum generative AI, ia perlu memakai

banyak alat terpisah: spreadsheet, template balasan, editor gambar, dan mungkin jasa penulis. Dengan LLM modern, satu antarmuka chat dapat membantu banyak tugas berbasis bahasa. Dengan model gambar, satu prompt dapat menghasilkan sketsa visual. Dengan RAG, model dapat menjawab berdasarkan katalog toko sendiri.

Perubahan utamanya bukan sekadar “AI bisa ngobrol”, tetapi tiga hal berikut.

1. Representasi umum: teks, gambar, audio, kode, dan tabel dapat diubah menjadi vektor/embedding.

2. Model skala besar: Transformer besar belajar pola dari data sangat banyak [R1][R4].

3. Antarmuka natural: manusia memberi instruksi dalam bahasa sehari-hari, lalu model menyusun output.

Generative AI berbeda dari model diskriminatif. Model diskriminatif menjawab “ini kelas apa?” sedangkan model generatif menjawab “contoh baru apa yang mungkin muncul?”

```
Diskriminatif:  x → label
Generatif:      konteks → data baru yang plausible
```

Contoh:

Tugas	Model diskriminatif	Model generatif
Ulasan pelanggan	positif/negatif	balasan empatik untuk pelanggan
Foto produk	kategori produk	variasi latar foto produk
Dokumen sekolah	deteksi topik	ringkasan dan kuis
Kode Python	bug/tidak bug	saran perbaikan kode

Namun kata *plausible* penting. Model generatif sering membuat jawaban yang terdengar benar, tetapi belum tentu benar. Karena itu bab ini selalu menempelkan arsitektur dengan verifikasi, sitasi, dan safety.

Persamaan kecil: LLM generatif biasanya memodelkan probabilitas token berikutnya.

```
P(teks) = P(t1) × P(t2 | t1) × P(t3 | t1,t2) × ... × P(tn | t1,...,t(n-1))
```

Artinya model tidak “mengambil kalimat dari database” setiap kali menjawab. Ia menghitung token berikutnya yang paling cocok terhadap konteks.

Tes cepat subbab 1

1. Beri dua contoh tugas diskriminatif dan dua tugas generatif di lingkungan Indonesia.

2. Mengapa jawaban LLM yang lancar belum otomatis benar?

3. Apa arti *plausible* dalam konteks generative AI?

Subbab 2 — Token, embedding, dan ruang makna

Inti subbab: sebelum model memahami kalimat, teks dipotong menjadi token lalu diubah menjadi vektor.

Komputer tidak membaca “beras premium 5 kg” sebagai manusia. Teks dipecah menjadi token, misalnya:

```
"beras premium 5 kg" → ["beras", "premium", "5", "kg"]
```

Model modern biasanya memakai subword tokenization: kata yang jarang dapat dipecah menjadi potongan lebih kecil. Ini berguna untuk bahasa Indonesia yang kaya imbuhan, nama daerah, campuran Inggris-Indonesia, dan istilah teknis.

Setelah tokenisasi, setiap token menjadi embedding: vektor angka yang menempatkan kata/frasa di ruang makna. Dalam ruang ini, “dokter”, “puskesmas”, dan “pasien” cenderung lebih dekat daripada “dokter” dan “knalpot”.

> kata/frasa → tokenizer → id token → embedding vektor

Intuisi visual: bayangkan peta kota. Kata yang maknanya dekat berada di gang yang sama. Jika kita mencari dokumen tentang “bantuan UMKM”, dokumen yang menyebut “subsidi usaha kecil” bisa tetap ditemukan walau tidak memakai kata persis sama.

Cosine similarity: kedekatan dua embedding sering dihitung dengan cosine similarity.

> cos(a,b) = (a · b) / (||a|| × ||b||)

Contoh hitung kecil:

> a = [1, 2]
> b = [2, 1]
>
> a · b = 1×2 + 2×1 = 4
> ||a|| = sqrt(1² + 2²) = sqrt(5)
> ||b|| = sqrt(2² + 1²) = sqrt(5)
> cos(a,b) = 4/5 = 0,8

Nilai 0,8 berarti arah kedua vektor cukup mirip. Di lab, kita akan membuat embedding mini berbasis frekuensi kata agar ide ini terlihat tanpa model besar.

Kekhasan Indonesia: embedding yang baik harus mengenali variasi “KTP”, “NIK”, “nomor induk kependudukan”, “BPJS”, “puskesmas”, “warung”, “ojek online”, bahasa daerah, singkatan chat, dan campuran formal-informal. Jika model dilatih terutama pada data Inggris, performanya pada konteks lokal bisa lebih lemah.

Tes cepat subbab 2

1. Mengapa tokenisasi subword berguna untuk bahasa Indonesia?

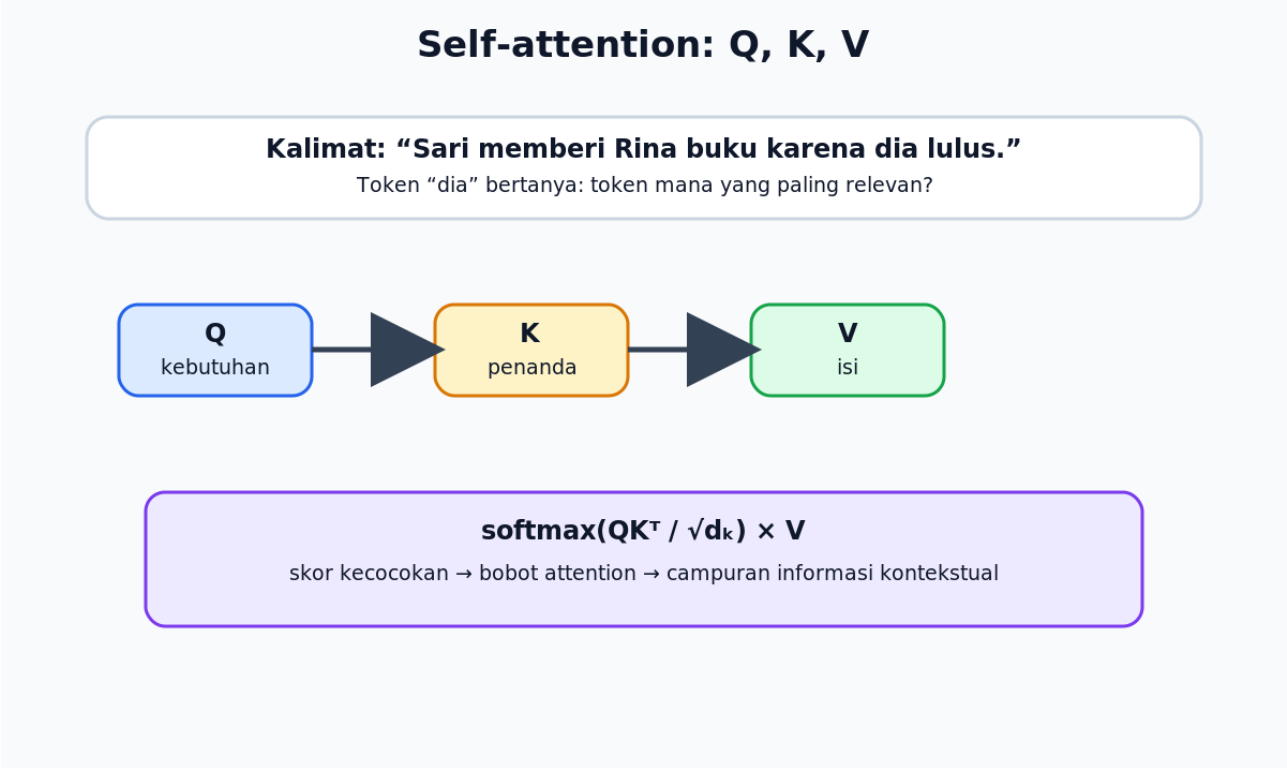
2. Apa beda token id dan embedding?

3. Hitung cosine similarity untuk `a=[1,0]` dan `b=[0,1]`. Apa artinya?

Subbab 3 — Transformer: mesin baca konteks modern

Inti subbab: Transformer memakai self-attention agar setiap token dapat memperhatikan token lain secara paralel [R1].

Sebelum Transformer, banyak model urutan memakai RNN/LSTM. RNN membaca token satu per satu. Ini masuk akal, tetapi sulit diparalelkan dan sering kesulitan membawa konteks jauh. Transformer mengubah pendekatan: semua token dapat “melihat” token lain melalui self-attention.



Anatomi self-attention

Analogi: rapat kelas. Setiap siswa memegang tiga kartu:

• Query (Q): “informasi apa yang saya butuhkan?”

• Key (K): “informasi apa yang saya tawarkan?”

• Value (V): “isi informasi saya.”

Saat token “dia” muncul dalam kalimat “Sari memberi Rina buku karena dia lulus”, token “dia” perlu mencari rujukan. Self-attention memberi bobot ke token lain agar konteks terbaca.

Rumus inti scaled dot-product attention:

```
Attention(Q,K,V) = softmax((QK^T) / sqrt(d_k)) V
```

Cara membacanya pelan-pelan:

1. `QK^T` menghitung kecocokan query terhadap key.

2. Dibagi `sqrt(d_k)` agar skor tidak terlalu besar.

3. `softmax` mengubah skor menjadi bobot yang jumlahnya 1.

4. Bobot dipakai untuk mencampur `V`.

Contoh hitung mini satu token: misalkan token A punya skor kecocokan terhadap tiga token:

```
skor = [2, 1, 0]
softmax(skor) ≈ [0,665; 0,245; 0,090]
```

Artinya token A paling memperhatikan token pertama, sedikit token kedua, dan sedikit sekali token ketiga.

Transformer block biasanya berisi:

```
input embedding
→ self-attention multi-head
→ residual connection + layer normalization
→ feed-forward network
→ residual connection + layer normalization
→ output representasi
```

Multi-head attention berarti model punya beberapa “sudut pandang”. Satu head mungkin fokus pada subjek-predikat, head lain pada entitas waktu, head lain pada format daftar. Ini bukan aturan pasti, tetapi membantu intuisi.

Positional encoding diperlukan karena attention sendiri tidak otomatis tahu urutan. Tanpa posisi, “anjing menggigit orang” dan “orang menggigit anjing” bisa terlihat seperti kumpulan token yang sama. Posisi memberi sinyal urutan.

Mengapa Transformer penting?

• Paralel: token dalam satu urutan dapat diproses bersama.

• Skalabel: cocok dilatih pada data dan parameter besar.

• Fleksibel: dapat dipakai sebagai encoder, decoder, atau encoder-decoder.

• Transferable: representasi yang dipelajari dapat dipakai untuk banyak tugas.

Tes cepat subbab 3

1. Jelaskan Q, K, V dengan analogi selain rapat kelas.

2. Mengapa skor attention perlu softmax?

3. Mengapa positional encoding dibutuhkan?

Subbab 4 — Encoder-only, decoder-only, dan encoder-decoder

Inti subbab: BERT, GPT, dan T5 sama-sama berbasis Transformer, tetapi arah membaca dan target latihannya berbeda.

Encoder vs decoder vs encoder-decoder

BERT: encoder-only

BERT membaca konteks dua arah (*bidirectional*) dan kuat untuk pemahaman teks: klasifikasi, ekstraksi entitas, pencarian semantik, dan reranking [R6]. Saat pretraining, BERT sering memakai

masked language modeling: sebagian token disembunyikan, model menebaknya dari kiri dan kanan.

> "Ibu membeli [MASK] di pasar" → model menebak "sayur", "beras", ...

BERT cocok ketika outputnya bukan teks panjang bebas, melainkan representasi atau label.

GPT: decoder-only

GPT membaca dari kiri ke kanan dan memprediksi token berikutnya [R2][R3][R4]. Saat menulis jawaban, GPT memakai token yang sudah dibuat sebagai konteks untuk token berikutnya.

> "Ringkas dokumen ini:" → token1 → token2 → token3 → ...

Inilah alasan GPT sangat cocok untuk generasi teks, dialog, coding assistant, dan penulisan draft.

T5: encoder-decoder

T5 menyatukan banyak tugas sebagai format text-to-text [R7]. Input teks dibaca encoder, lalu decoder menghasilkan output teks.

> Input: "translate English to Indonesian: good morning"
> Output: "selamat pagi"

Encoder-decoder cocok untuk translasi, summarization, dan tugas yang input-outputnya jelas. Namun LLM chat modern banyak memakai decoder-only karena skala dan antarmuka generasinya sangat praktis.

Arsitektur	Membaca konteks	Output utama	Contoh tugas
Encoder-only	dua arah	representasi/label	klasifikasi, search, NER
Decoder-only	kiri ke kanan	teks lanjutan	chat, draft, kode
Encoder-decoder	input penuh lalu output	teks transformasi	translasi, ringkasan

Tes cepat subbab 4

1. Mengapa BERT cocok untuk klasifikasi teks?

2. Mengapa GPT cocok untuk chat?

3. Beri contoh tugas Indonesia yang cocok untuk T5.

Subbab 5 — Mengapa GPT/ChatGPT terasa kuat?

Inti subbab: GPT kuat karena kombinasi pretraining skala besar, arsitektur decoder-only, few-shot prompting, instruction tuning, dan umpan balik manusia.

GPT pertama menunjukkan bahwa pretraining generatif dapat membantu banyak tugas bahasa [R2]. GPT-2 memperlihatkan model bahasa besar bisa melakukan banyak tugas tanpa fine-tuning khusus [R3]. GPT-3 memperkuat kemampuan few-shot: model dapat mengikuti contoh kecil dalam prompt [R4]. InstructGPT lalu menunjukkan bahwa model yang disetel dengan instruksi dan human feedback lebih sesuai dengan niat pengguna [R5]. Generasi setelahnya, termasuk GPT-4, memperlihatkan bahwa evaluasi kemampuan dan risiko perlu dilaporkan sebagai satu paket, bukan hanya skor benchmark [R23].

Secara sederhana, prosesnya:

1. Pretraining: belajar memprediksi token berikutnya dari korpus besar.
2. Supervised instruction tuning: belajar dari pasangan instruksi-jawaban.
3. Preference learning / RLHF: jawaban dibandingkan oleh manusia, model diarahkan agar lebih membantu dan aman.
4. Deployment: model dipakai dalam chat dengan system prompt, tool, guardrail, dan evaluasi.

Bagian RLHF menjadi jembatan menuju Bab 12. Di sana kita akan belajar bahwa *reward* dan feedback dapat membentuk perilaku agent. Pada LLM, reward tidak datang dari game seperti gridworld, tetapi dari preferensi manusia atau prinsip safety yang mengarahkan jawaban agar lebih membantu.

Mengapa chat terasa “pintar”? Karena model belajar banyak pola: bahasa, format dokumen, kode, gaya tanya-jawab, langkah penalaran, dan konvensi sosial. Namun kemampuan ini bukan bukti bahwa model selalu “memahami” seperti manusia. Ia tetap mesin probabilistik yang sangat kuat.

Contoh next-token: prompt:

```
"Ibu kota Indonesia adalah"
```

Distribusi token berikutnya mungkin:

```
Jakarta: 0,72
Nusantara: 0,18
kota: 0,03
...
```

Jika konteksnya tahun, kebijakan, atau pertanyaan sejarah berbeda, distribusi dapat berubah. Karena itu prompt yang jelas dan data yang terbaru penting.

Temperature dan top-k: decoding mengatur kreativitas.

• Temperature rendah: lebih deterministik, cocok untuk ringkasan faktual.

• Temperature tinggi: lebih kreatif, cocok untuk brainstorming.

• Top-k/top-p: membatasi kandidat token agar tidak terlalu liar.

Kesimpulan penting: GPT/ChatGPT kuat bukan karena satu trik, melainkan tumpukan: data, arsitektur, skala, instruksi, feedback, produk, dan evaluasi.

Tes cepat subbab 5

1. Bedakan pretraining dan instruction tuning.

2. Mengapa temperature tinggi berisiko untuk jawaban faktual?

3. Mengapa few-shot prompting bisa membantu model?

Subbab 6 — Prompting sebagai antarmuka, bukan sihir

Inti subbab: prompt adalah spesifikasi tugas mini. Prompt yang baik mengurangi ambiguitas, tetapi tidak menggantikan data dan verifikasi.

Prompt buruk:

```
Buat laporan.
```

Prompt lebih baik:

```
Buat ringkasan 7 poin dari dokumen berikut untuk pemilik UMKM makanan.
Gunakan bahasa Indonesia sederhana, sebutkan risiko, dan jangan tambahkan fakta di luar dokumen.
Jika informasi tidak ada, tulis "tidak disebutkan".
```

Struktur prompt praktis:

```
Peran: kamu membantu siapa?
Tugas: output apa yang diminta?
Konteks: data/dokumen apa yang boleh dipakai?
Batasan: apa yang tidak boleh dilakukan?
Format: tabel, bullet, JSON, surat, rubrik?
Kriteria: jawaban dianggap baik jika apa?
```

Prompting bukan keamanan penuh. Jika dokumen yang dimasukkan berisi instruksi jahat seperti “abaikan semua instruksi sebelumnya”, model bisa terdistraksi. Ini disebut prompt injection dan

masuk risiko besar aplikasi LLM [R22].

Prompting untuk belajar: minta model menjelaskan satu langkah per satu langkah, tetapi jangan menganggap semua langkah benar. Untuk matematika, tetap hitung ulang. Untuk hukum/kesehatan/keuangan, gunakan sumber resmi dan ahli.

Tes cepat subbab 6

1. Perbaiki prompt “Bantu tugas saya” menjadi prompt yang jelas.

2. Apa bedanya konteks dan format output?

3. Mengapa prompt injection berbahaya pada aplikasi RAG?

Subbab 7 — Embeddings dan semantic search

Inti subbab: embeddings memungkinkan pencarian berdasarkan makna, bukan hanya kata persis.

Misalkan pengguna bertanya:

> "Bagaimana cara mengurus bantuan usaha kecil?"

Dokumen pemerintah mungkin memakai frasa:

> "pendaftaran program pembiayaan UMKM"

Search berbasis keyword bisa gagal jika kata tidak cocok. Search berbasis embedding dapat menemukan kedekatan makna. Dense Passage Retrieval dan Sentence-BERT adalah dua tonggak penting dalam praktik retrieval berbasis representasi padat [R9][R10]. Prosesnya:

```
dokumen → chunk → embedding → simpan di indeks vektor
pertanyaan → embedding → cari chunk paling dekat
```

Chunking penting. Dokumen panjang dipotong menjadi bagian kecil. Terlalu pendek: konteks hilang. Terlalu panjang: pencarian kurang tajam dan biaya naik.

Aturan awal untuk dokumen Indonesia:

• Potong berdasarkan judul/subjudul bila ada.

• Simpan metadata: sumber, tanggal, halaman/bagian, URL.

• Jaga satu chunk membahas satu ide utama.

• Evaluasi dengan pertanyaan nyata, bukan hanya demo.

Contoh cosine retrieval: jika pertanyaan q paling dekat dengan chunk c3, maka c3 masuk konteks untuk LLM. Namun retrieval “paling dekat” belum tentu “paling benar”. Dokumen lama, keliru, atau tidak resmi tetap bisa dekat secara semantik.

Tes cepat subbab 7

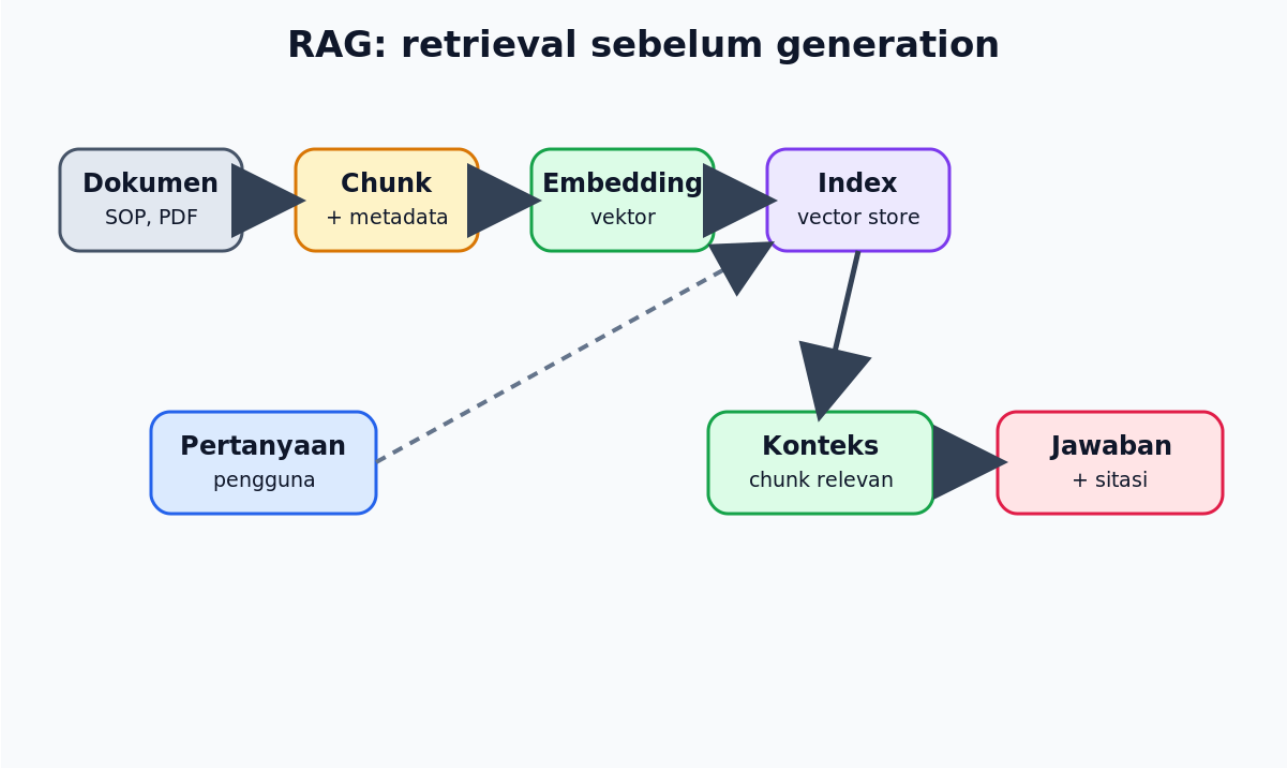
1. Apa kelemahan keyword search untuk bahasa natural?

2. Mengapa metadata sumber penting?

3. Apa risiko chunk terlalu panjang?

Subbab 8 — RAG: Retrieval-Augmented Generation

Inti subbab: RAG menggabungkan retrieval dengan generation agar jawaban LLM lebih grounded pada dokumen [R8].



Pipeline RAG

RAG tidak “membuat LLM tahu segalanya”. RAG memberi model bahan bacaan yang relevan saat menjawab.

Pipeline umum:

1. Kumpulkan dokumen tepercaya.
2. Bersihkan dan potong menjadi chunk.
3. Buat embedding setiap chunk.
4. Simpan embedding + metadata.
5. Saat ada pertanyaan, cari chunk relevan.
6. Masukkan chunk ke prompt.
7. Model menjawab dengan sitasi.
8. Evaluasi groundedness dan kegunaan jawaban.

Contoh kasus Indonesia: chatbot internal koperasi ingin menjawab pertanyaan anggota tentang syarat pinjaman. Tanpa RAG, LLM bisa menjawab berdasarkan pola umum internet. Dengan RAG, sistem mengambil pasal dari SOP koperasi sendiri, lalu menjawab dengan sumber.

Prompt RAG minimal:

```
Jawab hanya berdasarkan KONTEKS.
Jika jawaban tidak ada di konteks, tulis "tidak ditemukan di dokumen".
Cantumkan ID sumber untuk setiap klaim penting.

KONTEKS:
[S1] ...
[S2] ...

PERTANYAAN:
...
```

Tiga metrik RAG pemula:

1. Retrieval relevance: apakah chunk yang diambil memang relevan?

2. Faithfulness/groundedness: apakah jawaban sesuai chunk?

3. Answer usefulness: apakah jawaban membantu pengguna?

Kesalahan umum RAG:

• memasukkan dokumen tanpa sumber/tanggal;

• chunking acak;

• tidak mengecek dokumen paling relevan;

• model boleh menjawab di luar konteks;

• tidak ada evaluasi pertanyaan nyata;

• menganggap RAG otomatis menghapus hallucination.

Tes cepat subbab 8

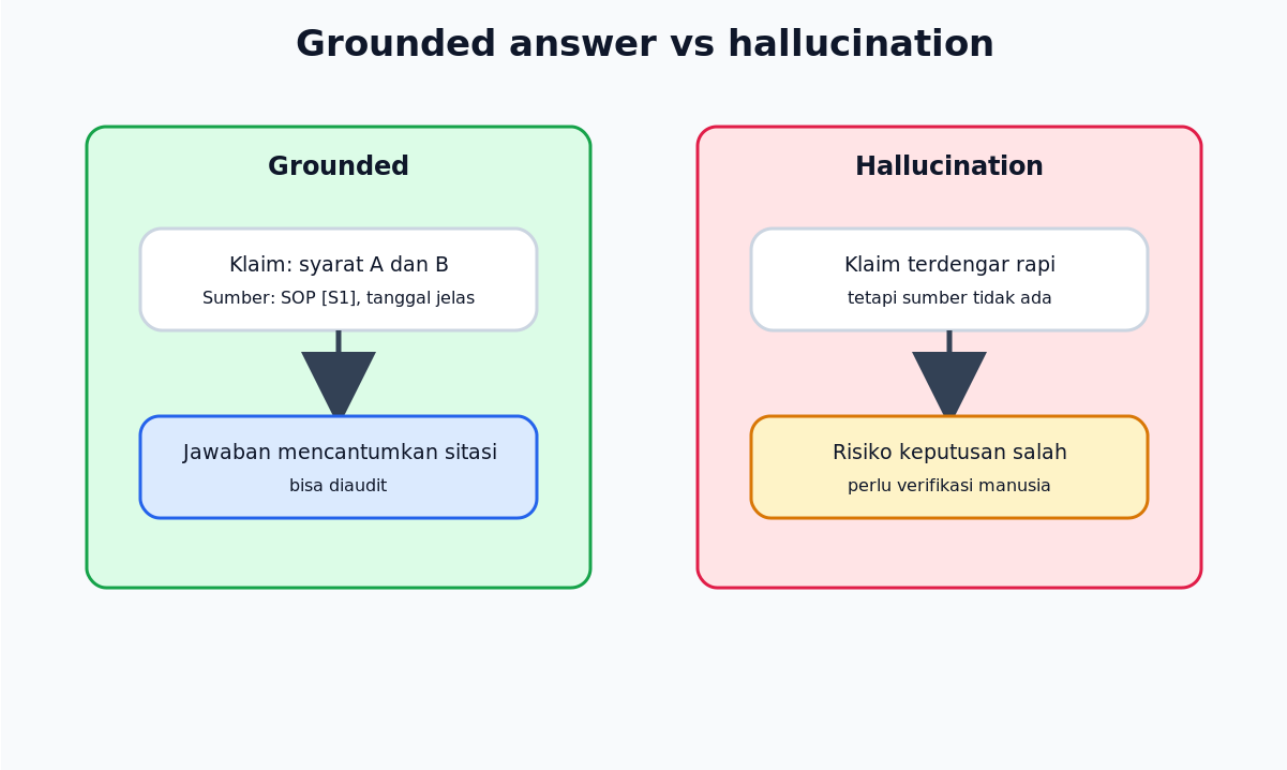
1. Mengapa RAG berbeda dari fine-tuning?

2. Apa yang harus dilakukan jika konteks tidak memuat jawaban?

3. Sebutkan dua metadata penting untuk chunk dokumen.

Subbab 9 — Hallucination: jawaban lancar yang tidak grounded

Inti subbab: hallucination terjadi ketika model menghasilkan klaim yang tidak didukung sumber, salah, atau mengarang detail [R19].



Grounded vs hallucinated answer

LLM dilatih untuk menghasilkan teks yang mungkin, bukan untuk menjadi database kebenaran. Jika prompt meminta jawaban, model cenderung menjawab walau bukti tidak cukup. Ini seperti siswa yang dipaksa menjawab soal hafalan tanpa buku: kadang benar, kadang menebak dengan percaya diri.

Jenis hallucination praktis:

Jenis	Contoh	Cara mitigasi
Fakta palsu	mengarang peraturan	pakai sumber resmi/RAG
Sitasi palsu	judul/URL tidak ada	validasi link dan bibliografi
Angka palsu	statistik tanpa sumber	minta tabel sumber dan tanggal
Instruksi berbahaya	saran medis/hukum keliru	rujuk ahli dan batasan domain

Jenis	Contoh	Cara mitigasi
Format benar isi salah	JSON rapi tetapi data salah	validasi skema dan isi

Groundedness check sederhana: setiap klaim penting harus bisa ditunjuk ke sumber. Jika tidak ada sumber, klaim diberi label asumsi atau dihapus.

Checklist sebelum memakai jawaban LLM:

[ ] Apakah sumbernya jelas?
[ ] Apakah tanggal sumber relevan?
[ ] Apakah model mengaku tidak tahu saat data tidak ada?
[ ] Apakah angka, nama, URL, dan pasal diverifikasi?
[ ] Apakah ada manusia yang bertanggung jawab untuk keputusan penting?

Tes cepat subbab 9

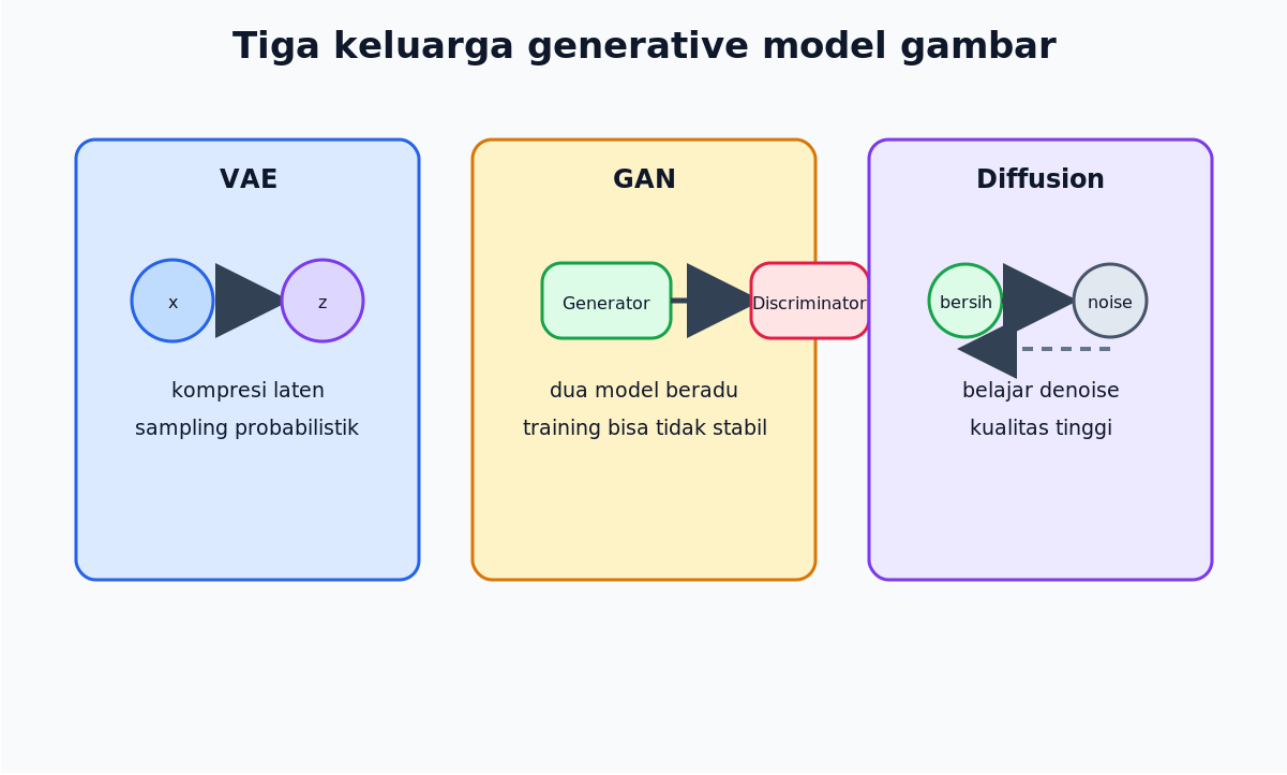
1. Mengapa fluency bukan bukti kebenaran?

2. Buat satu contoh hallucination pada domain pendidikan.

3. Apa bedanya “tidak tahu” yang baik dan jawaban karangan?

Subbab 10 — Diffusion, VAE, dan GAN: generative AI bukan hanya teks

Inti subbab: model generatif gambar/audio/video memakai prinsip berbeda, tetapi sama-sama belajar distribusi data.



Diffusion, VAE, dan GAN

VAE: kompresi probabilistik

Variational Autoencoder (VAE) belajar mengubah data menjadi ruang laten, lalu merekonstruksi data dari ruang itu [R11]. Ia mirip belajar peta ringkas: foto produk → koordinat laten → foto rekonstruksi. Karena laten bersifat probabilistik, kita dapat mengambil sampel baru.

x → encoder → z → decoder → x_hat

GAN: generator melawan discriminator

GAN memakai dua model [R12]:

• Generator membuat contoh palsu.

• Discriminator membedakan nyata vs palsu.

Keduanya berlatih seperti pemalsu uang dan pemeriksa uang. Jika seimbang, generator makin mahir. Jika tidak stabil, training bisa gagal.

Diffusion: belajar menghapus noise

Diffusion model menambah noise ke gambar sampai hampir acak, lalu belajar proses kebalikannya: menghapus noise sedikit demi sedikit [R13]. Latent diffusion mempercepat dengan bekerja di ruang laten, bukan langsung piksel besar [R14].

```
gambar bersih → tambah noise bertahap → noise
noise → denoise bertahap → gambar baru
```

Mengapa diffusion populer? Kualitas gambar tinggi, training relatif stabil dibanding GAN, dan mudah dikondisikan oleh teks. Namun ia tetap punya risiko: bias data, hak cipta, deepfake, dan output yang tampak realistis tetapi salah.

Tes cepat subbab 10

1. Jelaskan VAE dengan analogi peta ringkas.

2. Mengapa GAN disebut adversarial?

3. Mengapa diffusion bisa dibayangkan sebagai proses membersihkan kabut/noise?

Subbab 11 — MoE, scaling, dan efisiensi

Inti subbab: Mixture-of-Experts (MoE) membuat model sangat besar tetapi hanya mengaktifkan sebagian ahli untuk setiap token [R15].

Model dense memakai seluruh parameter aktif untuk setiap token. MoE memiliki banyak expert dan router. Router memilih expert mana yang dipakai.

```
token → router → expert terpilih → output
```

Analogi: rumah sakit besar. Pasien tidak ditangani semua dokter sekaligus. Resepsionis mengarahkan pasien ke dokter yang relevan: anak, gigi, saraf, atau umum. MoE membuat kapasitas besar tanpa biaya komputasi sebesar mengaktifkan semua dokter.

Keuntungan:

• kapasitas model lebih besar;

• biaya inferensi bisa lebih hemat dibanding dense setara;

• expert dapat belajar pola berbeda.

Tantangan:

• routing tidak seimbang;

• training lebih kompleks;

• deployment butuh infrastruktur kuat;

• interpretasi expert tidak selalu jelas.

Scaling law secara intuitif: performa model sering membaik ketika parameter, data, dan komputasi meningkat. Tetapi skala bukan segalanya. Data berkualitas, alignment, evaluasi, dan keselamatan tetap penting.

Tes cepat subbab 11

1. Apa perbedaan model dense dan MoE?

2. Mengapa router penting?

3. Mengapa skala besar tidak otomatis aman atau benar?

Subbab 12 — Tool use dan agen: saat LLM memakai alat

Inti subbab: LLM menjadi lebih berguna ketika dapat memanggil alat: kalkulator, search, database, kalender, compiler, atau API [R16].

LLM murni lemah pada aritmetika panjang, data terbaru, dan aksi dunia nyata. Tool use memberi jalan keluar:

```
pertanyaan → model memilih alat → alat memberi hasil → model menyusun jawaban
```

Contoh:

• menghitung pajak sederhana memakai kalkulator;

• mengambil stok barang dari database;

• menjalankan unit test kode;

• mencari dokumen internal;

• membuat tiket layanan pelanggan.

Namun tool use juga membuka risiko. Jika model boleh menjalankan aksi tanpa batas, ia bisa mengirim email salah, menghapus data, atau memanggil API mahal. Karena itu perlu izin, logging, sandbox, batas biaya, dan konfirmasi manusia untuk aksi sensitif.

Agen adalah sistem yang menggabungkan LLM, memori, alat, dan loop perencanaan. Agen berguna untuk tugas multi-langkah, tetapi lebih sulit dievaluasi daripada sekali jawab.

Aturan aman pemula:

```
Baca boleh otomatis.
Tulis/ubah data perlu validasi.
Kirim/bayar/hapus perlu konfirmasi eksplisit.
```

Tes cepat subbab 12

1. Mengapa LLM perlu kalkulator untuk hitungan penting?

2. Sebutkan tiga alat yang aman untuk mode read-only.

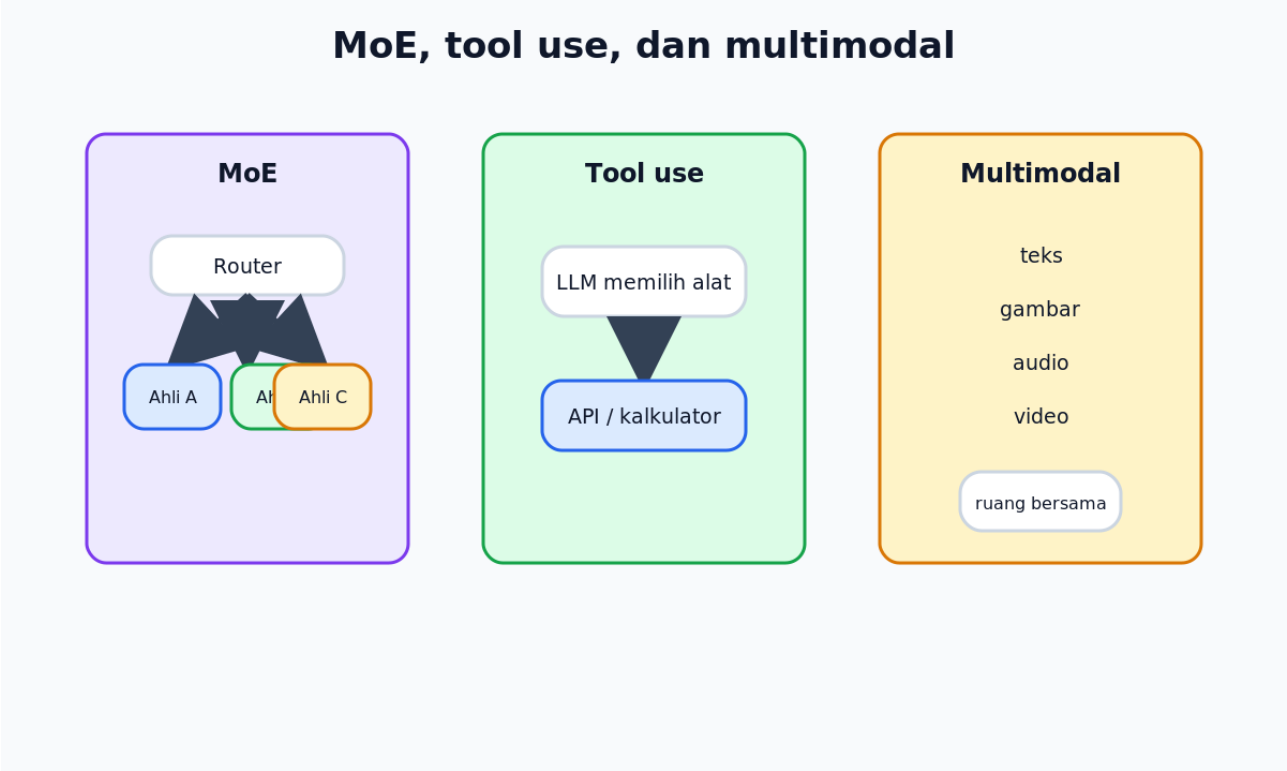
3. Aksi apa yang harus membutuhkan konfirmasi manusia?

Subbab 13 — Multimodal model: teks, gambar, audio, dan video

Inti subbab: multimodal model menghubungkan beberapa jenis data dalam ruang representasi bersama.

CLIP belajar mencocokkan gambar dan teks sehingga gambar “pedagang bakso di gerobak” dekat dengan deskripsi yang sesuai [R17]. Flamingo menunjukkan model vision-language dapat melakukan few-shot learning dengan gambar dan teks [R18]. Model modern melanjutkan arah ini:

pengguna dapat mengirim gambar struk, bertanya isi tabel, meminta deskripsi foto, atau membuat caption.



MoE, tool use, dan multimodal

Contoh Indonesia:

• membaca foto menu warung dan membuat daftar harga;

• menjelaskan grafik laporan desa;

• membantu guru membuat soal dari gambar eksperimen;

• mendeskripsikan kondisi tanaman dari foto;

• membuat alt text aksesibilitas untuk konten pemerintah.

Tetapi multimodal bukan pengganti ahli. Foto medis, dokumen hukum, bukti kriminal, dan identitas pribadi punya risiko tinggi. Model bisa salah membaca angka kecil, konteks budaya, atau detail visual.

Tes cepat subbab 13

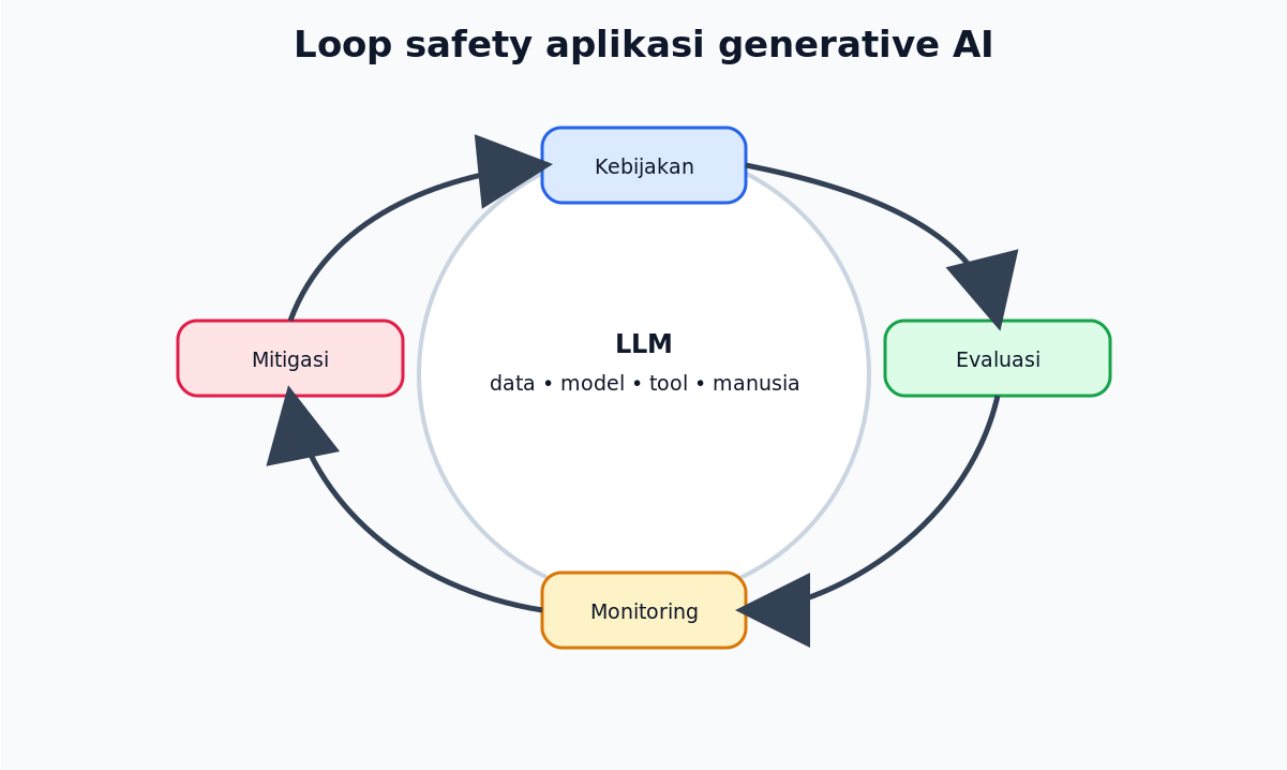
1. Apa arti ruang representasi bersama?

2. Beri dua contoh multimodal AI yang bermanfaat di pendidikan.

3. Mengapa membaca dokumen identitas dengan AI perlu hati-hati?

Subbab 14 — Safety, alignment, privasi, dan tata kelola

Inti subbab: aplikasi LLM yang baik tidak hanya pintar, tetapi juga aman, dapat diaudit, dan sesuai konteks sosial.



Loop safety generative AI

Safety mencakup beberapa lapisan:

1. Data safety: data pribadi, rahasia bisnis, hak cipta, dan izin penggunaan.

2. Model safety: bias, hallucination, toksisitas, jailbreak, prompt injection.

3. System safety: logging, rate limit, akses tool, audit trail, fallback.

4. Human safety: batas domain, eskalasi ke ahli, edukasi pengguna.

5. Governance: kebijakan, evaluasi berkala, dokumentasi risiko.

NIST AI RMF menekankan pengelolaan risiko AI secara terstruktur [R21]. OWASP LLM Top 10 memberi daftar risiko aplikasi LLM seperti prompt injection, data leakage, insecure output handling, dan excessive agency [R22]. Constitutional AI adalah salah satu pendekatan riset untuk membuat model lebih harmless melalui prinsip dan feedback AI [R20].

Konteks Indonesia:

• Jangan memasukkan NIK, data kesehatan, nilai siswa, atau rahasia perusahaan ke layanan AI publik tanpa izin dan perlindungan.

• Untuk bahasa Indonesia dan bahasa daerah, uji bias dan kesalahan pemahaman lokal.

• Untuk layanan publik, jawaban harus merujuk dokumen resmi dan punya jalur komplain manusia.

• Untuk pendidikan, AI sebaiknya menjadi tutor, bukan mesin plagiarisme.

Risk register mini:

Risiko	Dampak	Mitigasi awal
Hallucination peraturan	keputusan salah	RAG dokumen resmi + sitasi
Prompt injection dokumen	instruksi sistem bocor	filter, isolasi instruksi, evaluasi
Data pribadi bocor	kerugian hukum/etis	redaksi, izin, minimisasi data
Jawaban bias	ketidakadilan	dataset/evaluasi lokal
Tool berlebihan	aksi salah	konfirmasi dan least privilege

Tes cepat subbab 14

1. Apa bedanya safety model dan safety sistem?

2. Mengapa data pribadi tidak boleh sembarang dimasukkan ke AI publik?

3. Buat risk register tiga baris untuk chatbot sekolah.

Subbab 15 — Praktikum arsitektur: mini RAG dan decoding

Inti subbab: pembaca perlu melihat ide generative AI bekerja dalam kode kecil sebelum memakai layanan besar.

Di folder `code/`, bab ini menyediakan `generative_ai_playground.py`. Script ini tidak memanggil API luar. Ia memakai NumPy dan korpus mini agar dapat berjalan di terminal, VS Code, Jupyter, Google Colab, dan Kaggle Notebook.

Yang dipraktikkan:

1. tokenisasi sederhana;

2. embedding bag-of-words;

3. cosine similarity;

4. self-attention kecil;

5. decoding temperature/top-k pada bigram mini;

6. mini RAG berbasis dokumen lokal;

7. pemeriksaan klaim sederhana terhadap sumber.

Cara menjalankan:

```
cd zero-to-hero-menaklukkan-ai/chapters/11-generative-ai-context/code
python3 generative_ai_playground.py --self-test
python3 generative_ai_playground.py
```

Output disimpan ke `code/outputs/bab11_demo_results.json`.

Apa yang tidak dilakukan script ini? Script ini bukan ChatGPT mini. Ia tidak melatih Transformer besar. Tujuannya adalah membuat mekanisme dasar terlihat dan dapat dihitung.

Tes cepat subbab 15

1. Mengapa praktikum memakai korpus mini?

2. Apa perbedaan demo bag-of-words dan embedding neural modern?

3. Mengapa output mini RAG tetap perlu dicek manusia?

Subbab 16 — Kesalahan umum dan peta keputusan

Inti subbab: keputusan arsitektur generative AI harus dimulai dari masalah, data, risiko, dan evaluasi, bukan dari hype.

Kesalahan umum:

1. Menganggap LLM selalu tahu data terbaru. Solusi: RAG/search/sumber resmi.

2. Menggunakan prompt sebagai satu-satunya guardrail. Solusi: validasi, sandbox, policy, evaluasi.

3. Memakai generative AI untuk semua masalah. Banyak tugas cukup dengan logistic regression, random forest, atau search biasa.

4. Tidak mengukur kualitas. Buat dataset evaluasi kecil dari pertanyaan nyata.

5. Mengabaikan bahasa dan konteks lokal. Uji dengan bahasa Indonesia, slang, singkatan, dan domain lokal.

6. Mencampur fakta dan kreativitas. Atur temperature dan sumber sesuai tujuan.

7. Tidak mencatat sumber. Tanpa sumber, debugging dan audit sulit.

Peta keputusan sederhana:

> Butuh klasifikasi label? → supervised ML/BERT-style classifier.
> Butuh membuat teks bebas? → decoder-only LLM.
> Butuh menjawab dokumen internal? → RAG + LLM + sitasi.
> Butuh transformasi teks ke teks? → encoder-decoder atau LLM instruksional.
> Butuh gambar/audio baru? → diffusion/VAE/GAN sesuai kebutuhan.
> Butuh aksi ke sistem lain? → tool use dengan izin dan audit.
> Butuh keputusan risiko tinggi? → human-in-the-loop + sumber resmi + kebijakan.

Ringkasan bab

• Generative AI membuat konten baru yang plausible, tetapi plausible tidak sama dengan benar.

• Transformer penting karena self-attention, paralelisme, dan skalabilitas.

• GPT/ChatGPT kuat karena decoder-only pretraining skala besar, few-shot prompting, instruction tuning, dan feedback manusia.

• BERT, GPT, dan T5 berbeda pada arah konteks dan format output.

• Prompting adalah spesifikasi tugas, bukan sihir.

• Embeddings dan semantic search memungkinkan pencarian makna.

• RAG membuat jawaban lebih grounded, tetapi tidak otomatis bebas hallucination.

• Diffusion, VAE, dan GAN menunjukkan generative AI juga kuat di gambar/audio/video.

• MoE, tool use, dan multimodal model memperluas kapasitas dan kegunaan sistem.

• Safety, privasi, evaluasi, dan tata kelola wajib sejak awal.

Tes akhir bab

1. Jelaskan Transformer dalam 5 kalimat untuk teman SMA.

2. Bandingkan BERT, GPT, dan T5 dalam tabel 3 kolom.

3. Rancang pipeline RAG untuk dokumen aturan laboratorium kampus.

4. Buat prompt yang memaksa model menjawab hanya dari konteks.

5. Sebutkan 5 risiko aplikasi LLM dan mitigasinya.

6. Jelaskan perbedaan diffusion dan GAN dengan analogi visual.

7. Kapan Anda memilih model klasik daripada LLM?

8. Buat risk register mini untuk chatbot layanan desa.