BAB 2
LANDASAN TEORI

Bab ini memaparkan landasan teori dan penelitian terdahulu yang mendukung pengembangan sistem tanya jawab dokumen PDF multimodal berbasis *Retrieval-Augmented Generation* (RAG). Pembahasan mencakup konsep RAG, dokumen PDF multimodal, *preprocessing* adaptif dan OCR, representasi dokumen dan *embedding*, *Large Language Model* (LLM), *clustering*, RAPTOR, serta metrik evaluasi yang digunakan dalam penelitian. Selain itu, penelitian-penelitian terdahulu yang relevan juga dibahas untuk menunjukkan posisi dan kontribusi penelitian ini.

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) adalah pendekatan yang menggabungkan model bahasa generatif dengan mekanisme pencarian informasi dari sumber eksternal, dengan tujuan agar jawaban yang dihasilkan tidak hanya bergantung pada pengetahuan parametrik model, melainkan juga didukung oleh konteks yang diambil dari dokumen relevan [1]. Dalam tugas *question answering*, model generatif menghasilkan jawaban secara autoregresif berdasarkan masukan berupa pertanyaan dan konteks. Proses menghasilkan jawaban dapat dinyatakan sebagai berikut:

$$P(y \mid x) = \prod_{t=1}^{n} P(y_t \mid x, y_{<t}) \tag{2.1}$$

dengan x adalah masukan atau pertanyaan, y adalah jawaban yang dihasilkan, y_t adalah token ke-t, dan $y_{<t}$ adalah token-token yang telah dihasilkan sebelumnya. Pada model generatif murni, jawaban bergantung sepenuhnya pada pengetahuan parametrik yang diperoleh saat pelatihan, sehingga rentan menghasilkan informasi yang tidak akurat untuk topik spesifik berbasis dokumen [2, 4].

RAG mengatasi keterbatasan tersebut dengan menambahkan tahap *retrieval* sebelum menghasilkan jawaban. Secara matematis, pendekatan RAG dapat dinyatakan sebagai berikut:

$$P(y \mid x) \approx \sum_{z \in \mathscr{D}_k(x)} P_\eta(z \mid x) \cdot P_\theta(y \mid x, z) \tag{2.2}$$

dengan z adalah dokumen atau konteks yang diambil, $\mathscr{D}_k(x)$ adalah himpunan k dokumen teratas hasil *retrieval*, $P_\eta(z \mid x)$ adalah probabilitas relevansi dokumen berdasarkan *retriever*, dan $P_\theta(y \mid x, z)$ adalah probabilitas jawaban yang dihasilkan oleh *generator* berdasarkan pertanyaan dan konteks [1]. Kinerja sistem RAG sangat dipengaruhi oleh relevansi dan kelengkapan konteks yang diperoleh pada tahap *retrieval*. Apabila konteks yang diperoleh tidak relevan atau tidak memuat informasi yang diperlukan, jawaban yang dihasilkan berpotensi tidak akurat [3, 6].

Pipeline RAG secara umum terdiri atas pemrosesan dokumen, pembentukan indeks *embedding*, pencarian konteks, penyusunan *prompt*, dan menghasilkan jawaban. Secara konvensional, dokumen dibagi menjadi potongan teks datar (*flat chunks*) dengan ukuran tertentu, kemudian direpresentasikan sebagai vektor. Pendekatan ini memiliki keterbatasan pada dokumen panjang dan kompleks karena berisiko kehilangan konteks lintas bagian dokumen [7, 8]. Keterbatasan tersebut mendorong pengembangan berbagai pendekatan *retrieval* yang dirancang untuk mempertahankan konteks pada dokumen panjang dan kompleks, sebagaimana dibahas pada subbab-subbab berikutnya.

Kualitas jawaban yang dihasilkan *generator* juga dipengaruhi oleh bagaimana konteks hasil *retrieval* disusun dalam *prompt*. Dua instruksi yang umum diterapkan adalah *grounding*, yaitu instruksi eksplisit agar model hanya menjawab berdasarkan konteks yang diberikan untuk menekan *hallucination* [18], dan *Chain-of-Thought* (CoT), yaitu instruksi agar model bernalar bertahap sebelum menjawab [19]. Kedua teknik ini juga relevan pada tahap peringkasan klaster RAPTOR yang dibahas pada Subbab 2.7, di mana *prompt* perlu dirancang agar ringkasan tetap setia terhadap teks sumber [20].

2.2 Dokumen PDF Multimodal

Dokumen PDF multimodal dalam penelitian ini merujuk pada dokumen PDF yang tidak hanya memuat teks naratif, tetapi juga mengandung elemen semi-terstruktur dan visual, seperti tabel, daftar bernomor, struktur bab dan subbab, lampiran, gambar yang mengandung teks, serta halaman hasil pemindaian. Format PDF banyak digunakan sebagai format dokumen resmi karena mampu mempertahankan tata letak dan struktur tampilan dokumen secara konsisten.

Berbeda dengan dokumen teks biasa, PDF menyimpan informasi berdasarkan representasi tampilan halaman. Akibatnya, proses ekstraksi teks dari PDF dapat menghasilkan urutan teks yang tidak sesuai dengan struktur asli dokumen, terutama pada dokumen yang memiliki tabel, teks multi-kolom, *header*, *footer*, gambar, atau lampiran [11, 12]. *Tripathi et al.* [13] menunjukkan bahwa metode *chunking* berbasis teks konvensional gagal menangani tabel yang memanjang lintas halaman dan elemen visual pada dokumen PDF kompleks. Kesalahan pada tahap ekstraksi berpotensi menyebabkan hilangnya informasi atau perubahan konteks dari dokumen asli. Kondisi tersebut dapat memengaruhi kualitas *retrieval* karena informasi yang diindeks tidak lagi merepresentasikan isi dokumen secara utuh [17].

2.3 Preprocessing Adaptif dan OCR

Preprocessing adaptif adalah proses pemrosesan awal dokumen yang disesuaikan dengan karakteristik masing-masing halaman atau elemen dokumen. Pendekatan ini diperlukan karena dokumen PDF multimodal tidak selalu memiliki format yang seragam: sebagian halaman berupa teks digital, sebagian memuat tabel dengan *layout* kompleks, dan sebagian lainnya berupa hasil pemindaian [11, 12]. Keluaran dari proses *preprocessing* adaptif adalah himpunan unit dokumen yang dapat dinyatakan sebagai berikut:

$$U = \{u_1, u_2, \ldots, u_n\} \tag{2.3}$$

dengan setiap unit u_i memuat isi teks, metadata sumber, nomor halaman, struktur bab atau subbab, tipe elemen dokumen, serta informasi apakah teks diperoleh melalui OCR atau ekstraksi langsung.

Optical Character Recognition (OCR) merupakan bagian dari *preprocessing* yang diperlukan ketika halaman PDF tidak menyimpan teks dalam lapisan digital, melainkan berupa gambar hasil pemindaian. Akurasi OCR dipengaruhi oleh kualitas pemindaian, resolusi, kemiringan halaman, dan jenis font. Kesalahan pengenalan dapat menyebabkan istilah akademik, angka, atau nomor pasal terbaca secara keliru, sehingga berdampak pada kualitas informasi yang masuk ke basis pengetahuan [12]. Oleh karena itu, *preprocessing* adaptif perlu menangani setiap jenis halaman secara berbeda agar informasi yang diekstraksi tetap akurat dan terstruktur.

2.4 Representasi Dokumen dan Embedding

Representasi dokumen merupakan tahap penting dalam sistem RAG karena dokumen harus diubah menjadi bentuk yang dapat dicari secara efisien. Pada pendekatan *dense retrieval*, teks pertanyaan dan teks dokumen direpresentasikan sebagai vektor berdimensi tetap menggunakan model *embedding*. Vektor tersebut merepresentasikan makna semantik dari teks, sehingga dokumen yang relevan terhadap pertanyaan dapat ditemukan meskipun tidak memiliki kata kunci yang sama persis.

Misalkan q adalah pertanyaan pengguna dan d_i adalah unit dokumen ke-i. Jika fungsi *embedding* dilambangkan sebagai $E(\cdot)$, maka vektor pertanyaan dan dokumen dinyatakan sebagai berikut:

$$v_q = E(q), \quad v_i = E(d_i) \tag{2.4}$$

Kedekatan antara pertanyaan dan dokumen dihitung menggunakan *cosine similarity*:

$$\text{sim}(q, d_i) = \frac{v_q \cdot v_i}{\|v_q\| \cdot \|v_i\|} \tag{2.5}$$

Nilai kemiripan yang lebih tinggi menunjukkan relevansi semantik yang lebih besar. Dalam sistem RAG, unit dokumen dengan nilai kemiripan tertinggi dipilih sebagai konteks yang dimasukkan ke dalam *prompt generator*. Selain digunakan untuk pencarian, *embedding* dalam konteks RAPTOR juga menjadi dasar pengelompokan (*clustering*) dan peringkasan rekursif dalam pembangunan struktur hierarkis.

Model *embedding* untuk *dense retrieval* umumnya menggunakan arsitektur *bi-encoder*, yaitu pertanyaan dan dokumen dikodekan menjadi vektor secara terpisah sehingga vektor dokumen dapat dihitung sekali di awal dan disimpan dalam indeks [21]. Karakteristik ini menjadikan *bi-encoder* efisien untuk pencarian berskala besar, sehingga sesuai digunakan pada tahap *retrieval* utama dalam sistem RAG.

Untuk dokumen berbahasa Indonesia, diperlukan model *embedding* yang mendukung representasi lintas bahasa (*multilingual*). Salah satu model yang banyak digunakan adalah *multilingual-E5* [22], yang dilatih secara kontrastif pada pasangan teks dari banyak bahasa dan menghasilkan representasi semantik yang kuat pada

tugas *retrieval*. Model keluarga E5 menggunakan konvensi penambahan prefiks `query:` pada pertanyaan dan `passage:` pada dokumen sebelum proses *embedding*, yang berpengaruh signifikan terhadap kualitas representasi [22]. Berdasarkan karakteristik tersebut, *multilingual-E5* dipilih sebagai model *embedding* pada penelitian ini karena mampu merepresentasikan dokumen yang memuat campuran bahasa Indonesia dan istilah teknis berbahasa Inggris secara efektif.

2.5 Large Language Model

Large Language Model (LLM) adalah model bahasa berukuran besar yang dilatih pada korpus teks dalam jumlah sangat besar untuk mempelajari pola bahasa dan pengetahuan umum. Sebagian besar LLM modern dibangun di atas arsitektur *Transformer* [23], yang memanfaatkan mekanisme *self-attention* untuk menangkap hubungan antar-token dalam sebuah urutan secara paralel. Dalam sistem RAG, LLM berperan sebagai komponen generatif yang menghasilkan jawaban berdasarkan konteks hasil *retrieval*; pada pendekatan RAPTOR, LLM juga digunakan untuk meringkas klaster pada proses pembangunan struktur hierarkis [14].

Model LLM dapat bersifat tertutup, seperti GPT [24] yang umumnya diakses melalui API berbayar tanpa dapat dijalankan secara lokal, atau bersifat terbuka (*open-source*) sehingga dapat diunduh dan dijalankan pada perangkat sendiri. Penelitian ini menggunakan model terbuka karena seluruh proses inferensi dijalankan secara lokal sehingga dokumen tidak perlu dikirim ke layanan pihak ketiga. Pendekatan ini sesuai dengan karakteristik dokumen institusional yang menuntut kerahasiaan dan pengendalian data.

Agar LLM berukuran besar dapat dijalankan pada perangkat keras dengan keterbatasan memori, digunakan teknik *quantization*, yaitu menurunkan presisi bobot model (misalnya dari 16-bit menjadi 4-bit) sehingga kebutuhan memori berkurang signifikan dengan penurunan kualitas yang relatif kecil [25]. Dengan teknik ini, model seperti Qwen2.5 versi 7B parameter dapat dijalankan pada GPU kelas konsumen. Qwen2.5 dipilih sebagai *generator* pada penelitian ini karena mendukung pemrosesan multibahasa, tersedia dalam varian terkuantisasi, dan dapat dijalankan secara lokal pada perangkat penelitian, sesuai dengan kebutuhan sistem RAG yang dikembangkan. Kebutuhan akan model penilai (*judge*) yang independen dari model *generator* untuk menghindari bias *self-preference* dibahas pada Subbab 2.8 bersama metrik evaluasi yang digunakan.

2.6 Clustering: Reduksi Dimensi dan Gaussian Mixture Model

Pembangunan struktur hierarkis pada RAPTOR memerlukan proses pengelompokan (*clustering*) terhadap *embedding* unit dokumen. Namun, *embedding* dokumen umumnya memiliki dimensi yang sangat tinggi (misalnya 1024 dimensi), sehingga proses *clustering* secara langsung menjadi tidak efektif akibat fenomena *curse of dimensionality*. Untuk mengatasi hal ini, dilakukan reduksi dimensi terlebih dahulu menggunakan *Uniform Manifold Approximation and Projection* (UMAP) [26]. UMAP memetakan data berdimensi tinggi ke ruang berdimensi lebih rendah dengan tetap mempertahankan struktur lokal dan, hingga batas tertentu, struktur global data, sehingga klaster semantik menjadi lebih mudah dipisahkan.

Setelah reduksi dimensi, pengelompokan dilakukan menggunakan *Gaussian Mixture Model* (GMM). GMM adalah model probabilistik yang mengasumsikan data berasal dari campuran beberapa distribusi Gaussian, dan menghitung probabilitas keanggotaan setiap titik terhadap setiap klaster (*soft clustering*) [27]. Pendekatan ini dipilih pada RAPTOR karena sebuah unit dokumen dapat memiliki keterkaitan dengan lebih dari satu topik, sehingga keanggotaan probabilistik lebih sesuai dibandingkan pengelompokan tegas (*hard clustering*). Kombinasi UMAP dan GMM inilah yang digunakan *Sarthi et al.* [14] dalam pembangunan pohon RAPTOR dan diadopsi pada penelitian ini.

Jumlah komponen (klaster) pada GMM tidak ditentukan secara manual, melainkan dipilih secara otomatis menggunakan *Bayesian Information Criterion* (BIC) [28], yaitu kriteria pemilihan model yang menyeimbangkan kecocokan model terhadap data (*likelihood*) dengan kompleksitasnya melalui pemberian penalti terhadap jumlah parameter:

$$\mathrm{BIC} = -2\ln\hat{L} + p\ln n \tag{2.6}$$

dengan $\hat{L}$ adalah *likelihood* maksimum model, p jumlah parameter, dan n jumlah data. Model dengan nilai BIC terendah dipilih, sehingga jumlah klaster yang terbentuk menyesuaikan struktur data tanpa kecenderungan berlebihan (*overfitting*). Pada RAPTOR, GMM dengan pemilihan jumlah komponen via BIC diterapkan pada tahap pengelompokan global maupun lokal [14].

2.7 Hierarchical Retrieval dan RAPTOR

Hierarchical retrieval adalah pendekatan *retrieval* yang merepresentasikan dokumen dalam beberapa tingkat granularitas. Berbeda dengan *retrieval* datar yang hanya mencari pada satu level, *hierarchical retrieval* memungkinkan sistem menemukan informasi pada level detail maupun level ringkasan secara bersamaan [14, 7]. Pendekatan ini sesuai untuk dokumen panjang seperti handbook kampus, di mana pertanyaan sederhana dapat dijawab dari satu unit teks, sedangkan pertanyaan mengenai prosedur atau aturan dapat memerlukan gabungan beberapa bagian dokumen dari bab yang berbeda.

RAPTOR (*Recursive Abstractive Processing for Tree-Organized Retrieval*) adalah implementasi *hierarchical retrieval* yang mengorganisasi dokumen ke dalam struktur pohon melalui proses pengelompokan dan peringkasan rekursif [14]. Pembangunan pohon RAPTOR dimulai dari himpunan *chunks* awal pada level dasar:

$$C^{(0)} = \{c_1^{(0)}, c_2^{(0)}, \ldots, c_n^{(0)}\} \tag{2.7}$$

Pada setiap level l, unit-unit dikelompokkan berdasarkan kemiripan semantik menjadi klaster $G_j^{(l)}$, kemudian setiap klaster diringkas menggunakan model generatif untuk menghasilkan node baru pada level berikutnya:

$$c_j^{(l+1)} = \text{Summarize}(G_j^{(l)}) \tag{2.8}$$

Proses rekursif tersebut menghasilkan struktur pohon retrieval:

$$T = \bigcup_{l=0}^{L} C^{(l)} \tag{2.9}$$

dengan T adalah struktur pohon dan L adalah level tertinggi dalam hierarki. Pada saat *retrieval*, sistem dapat mengambil node relevan dari berbagai level: node level bawah menyediakan konteks detail, sedangkan node level atas menyediakan ringkasan konteks yang lebih luas. *Sarthi et al.* [14] menunjukkan bahwa RAPTOR meningkatkan performa secara signifikan pada tugas *question answering* yang memerlukan pemahaman multitingkat dibandingkan RAG konvensional. Dalam

penelitian ini, RAPTOR diadaptasi untuk dokumen PDF multimodal yang telah melalui *preprocessing* adaptif, sehingga node-node pohon dibangun dari unit informasi yang berasal dari teks digital, tabel yang dinormalisasi, dan hasil OCR.

Karena setiap node pada level $l+1$ dibangun dari ringkasan node-node di level l (Persamaan 2.8), kesalahan yang muncul pada tahap *preprocessing* adaptif, misalnya teks OCR yang salah dikenali atau struktur tabel yang tidak terbaca utuh, berisiko ikut terbawa dan membesar saat diringkas secara rekursif ke level-level berikutnya. Oleh karena itu, *preprocessing* adaptif dan RAPTOR diterapkan dalam satu *pipeline*. Kualitas unit dokumen yang dihasilkan pada level dasar menentukan kualitas ringkasan yang dibentuk pada level-level berikutnya. Dengan demikian, keberhasilan RAPTOR tidak hanya bergantung pada proses *retrieval* hierarkis, tetapi juga pada kualitas informasi yang diperoleh selama tahap *preprocessing*.

Salah satu konsekuensi penerapan RAPTOR adalah meningkatnya kebutuhan komputasi karena proses peringkasan dilakukan pada setiap klaster dan setiap level hierarki. Semakin besar jumlah dokumen yang diproses dan semakin dalam struktur pohon yang dibangun, semakin besar pula biaya komputasi yang diperlukan. Oleh sebab itu, konfigurasi *chunking* dan jumlah level pohon perlu ditentukan dengan mempertimbangkan karakteristik dataset yang digunakan.

2.8 Metrik Evaluasi Sistem RAG

Evaluasi sistem RAG dilakukan pada dua aspek utama: kualitas *retrieval* dan kualitas jawaban [3, 6]. Metrik yang digunakan untuk mengevaluasi kualitas *retrieval* adalah *Precision@k*, *Recall@k*, dan *Mean Reciprocal Rank* (MRR).

Precision@k mengukur proporsi dokumen relevan dari k hasil teratas yang dikembalikan oleh sistem. Jika $\text{Rel}_k(q)$ adalah jumlah dokumen relevan pada k hasil teratas untuk pertanyaan q, maka:

$$\text{Precision@}k = \frac{\text{Rel}_k(q)}{k} \tag{2.10}$$

Recall@k mengukur proporsi dokumen relevan yang berhasil ditemukan dari seluruh dokumen relevan yang ada:

$$\text{Recall@}k = \frac{\text{Rel}_k(q)}{|\text{Rel}(q)|} \tag{2.11}$$

dengan $|\text{Rel}(q)|$ adalah jumlah seluruh dokumen relevan untuk pertanyaan q. MRR mengukur posisi dokumen relevan pertama dalam hasil *retrieval*:

$$\text{MRR} = \frac{1}{|Q|} \sum_{q=1}^{|Q|} \frac{1}{\text{rank}_q} \tag{2.12}$$

dengan rank_q adalah peringkat dokumen relevan pertama untuk pertanyaan q dan $|Q|$ adalah jumlah pertanyaan evaluasi. Nilai MRR yang lebih tinggi menunjukkan bahwa dokumen relevan ditemukan pada posisi yang lebih awal dalam hasil *retrieval*.

Evaluasi kualitas jawaban menggunakan dua metrik: *faithfulness* dan *answer relevancy*. *Faithfulness* mengukur sejauh mana klaim dalam jawaban didukung oleh konteks yang diambil, sedangkan *answer relevancy* mengukur kesesuaian jawaban terhadap pertanyaan pengguna. Kedua metrik tersebut digunakan untuk memastikan bahwa jawaban yang dihasilkan tidak hanya relevan terhadap pertanyaan pengguna, tetapi juga didukung oleh konteks yang diperoleh pada tahap *retrieval*. Dengan demikian, kualitas jawaban dapat dievaluasi dari aspek ketepatan informasi maupun kesesuaiannya terhadap pertanyaan [3].

Berbeda dengan metrik *retrieval* yang dapat dihitung langsung dari *ground truth*, *faithfulness* dan *answer relevancy* bersifat kualitatif sehingga sulit diukur secara otomatis. Salah satu pendekatan yang banyak digunakan untuk mengatasi hal ini adalah *LLM-as-judge*, yaitu memanfaatkan LLM sebagai penilai yang diberi rubrik dan skala skor tertentu untuk menilai keluaran sistem secara konsisten. *Liu et al.* [29] memformalkan pendekatan ini melalui kerangka G-Eval, yang menggunakan LLM dengan panduan kriteria untuk mengevaluasi kualitas teks dan menunjukkan korelasi yang lebih tinggi terhadap penilaian manusia dibandingkan metrik berbasis kecocokan kata. Khusus untuk sistem RAG, *Es et al.* [30] mengusulkan kerangka RAGAS (*Retrieval-Augmented Generation Assessment*) yang mendefinisikan metrik *faithfulness* dan *answer relevancy* serta menilainya secara otomatis tanpa memerlukan jawaban acuan. Pendekatan *LLM-as-judge* inilah yang diadopsi pada penelitian ini untuk mengevaluasi kualitas jawaban.

Untuk menghindari kecenderungan sebuah LLM menilai keluarannya sendiri lebih tinggi, model *judge* dipilih dari model yang berbeda dari model *generator*. Penelitian ini menggunakan google/gemma-4-27b-it [31] sebagai *judge*, yaitu model *Mixture-of-Experts* dengan 26 miliar parameter aktif yang mendukung penalaran bertahap (*chain-of-thought*) dan dapat dijalankan secara

lokal. Pemisahan model *generator* dari model penilai mengikuti rekomendasi *Zheng et al.* [32], yang menunjukkan bahwa *LLM-as-judge* rentan terhadap bias *self-preference* apabila penilai menilai jawaban dari model yang sama.

2.9 Penelitian Terdahulu

Beberapa penelitian terdahulu telah mengeksplorasi RAG, *hierarchical retrieval*, pemrosesan dokumen PDF, dan sistem tanya jawab kampus. Penelitian-penelitian tersebut dirangkum pada Tabel 2.1 sebagai dasar konseptual dan bukti empiris posisi penelitian ini.

Tabel 2.1. Ringkasan penelitian terdahulu

No	Tahun	Penelitian	Latar Belakang dan Hasil	Celah Penelitian
1	2020	*Lewis et al.* [1]	Memperkenalkan RAG yang menggabungkan *retriever* berbasis dokumen eksternal dengan *generator*. Menunjukkan bahwa pengambilan pengetahuan eksternal mengurangi *hallucination* dan meningkatkan akurasi faktual.	Menggunakan representasi teks datar tanpa mempertimbangkan struktur hierarkis dokumen, sehingga belum mampu menangani dokumen panjang dan kompleks secara optimal.
2	2024	*Sarthi et al.* [14]	Mengusulkan RAPTOR yang mengorganisasi dokumen ke dalam struktur pohon hierarkis melalui *clustering* dan peringkasan rekursif. Menunjukkan peningkatan performa pada *benchmark* QuALITY dan NarrativeQA dibandingkan RAG konvensional.	RAPTOR diujikan pada korpus teks bersih dan belum mempertimbangkan tantangan ekstraksi dari dokumen PDF multimodal dengan tabel, gambar, dan halaman pemindaian.
Lanjut pada halaman berikutnya				

Tabel 2.1 Ringkasan penelitian terdahulu (lanjutan)

No	Tahun	Penelitian	Latar Belakang dan Hasil	Celah Penelitian
3	2025	*Tripathi et al.* [13]	Mengusulkan *vision-guided chunking* yang mempertimbangkan struktur visual halaman untuk mengatasi keterbatasan *chunking* teks pada PDF kompleks. Menunjukkan peningkatan akurasi *retrieval* pada dokumen dengan tabel lintas halaman dan elemen visual.	Berfokus pada dokumen umum tanpa *hierarchical retrieval*; belum diterapkan pada domain dokumen institusional berbahasa Indonesia.
4	2025	*Drushchak et al.* [15]	Mengembangkan *pipeline* RAG yang memproses teks, gambar, tabel, dan video secara terintegrasi. Menunjukkan bahwa integrasi multimodal meningkatkan cakupan informasi yang dapat diambil oleh sistem.	Berfokus pada domain umum dan tidak mempertimbangkan kebutuhan *preprocessing* adaptif untuk PDF institusional dengan struktur bab-subbab dan tabel regulasi.

Lanjut pada halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu (lanjutan)

No	Tahun	Penelitian	Latar Belakang dan Hasil	Celah Penelitian
5	2026	Yasuno [16]	Mengusulkan RAPTOR-AI yang mengintegrasikan pemahaman gambar ke dalam struktur hierarkis RAPTOR untuk kebutuhan respons bencana. Melaporkan peningkatan presisi *retrieval* dibandingkan pendekatan *baseline* pada dataset respons bencana yang digunakan dalam studi tersebut.	Dirancang untuk domain bencana dengan dokumen laporan; belum diterapkan pada dokumen administratif institusional yang memiliki struktur regulasi dan prosedur.
6	2024	*Tribber et al.* [9]	Mengembangkan chatbot RAG untuk layanan informasi kampus di Indonesia (LangChain, ChromaDB, Gemini). Menunjukkan bahwa RAG efektif untuk pertanyaan informasi akademik yang bersifat langsung.	Memecah dokumen menjadi potongan teks datar tanpa representasi hierarkis; belum dirancang untuk menangani ketergantungan konteks lintas bagian dokumen PDF multimodal.

Lanjut pada halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu (lanjutan)

No	Tahun	Penelitian	Latar Belakang dan Hasil	Celah Penelitian
7	2026	*Bustomy dan Rosid* [10]	Mengembangkan chatbot RAG akademik kampus menggunakan LLaMA 3.1 (LangGraph, ChromaDB; *preprint*). Dievaluasi dengan kerangka RAGAS dan menghasilkan *Context Precision* 0,86 serta *Faithfulness* 0,77.	Tidak menerapkan representasi hierarkis; hasil menunjukkan ruang peningkatan terutama pada pertanyaan yang membutuhkan konteks lintas bagian, serta belum mengintegrasikan *preprocessing* adaptif.

Berdasarkan Tabel 2.1, penelitian terdahulu dapat dikelompokkan menjadi dua kategori utama. Kategori pertama berfokus pada pengembangan *hierarchical retrieval*, seperti RAPTOR [14] dan RAPTOR-AI [16], namun diujikan pada korpus teks bersih atau domain non-institusional. Kategori kedua berfokus pada pemrosesan dokumen PDF multimodal atau pengembangan sistem tanya jawab kampus [13, 15, 9, 10], tetapi belum memanfaatkan representasi hierarkis.

Sejauh penelusuran pustaka yang dilakukan, belum ditemukan penelitian yang mengintegrasikan *preprocessing* adaptif untuk dokumen PDF multimodal dengan representasi hierarkis berbasis RAPTOR pada dokumen handbook institusional berbahasa Indonesia. Oleh karena itu, penelitian ini berupaya mengisi celah tersebut melalui penggabungan kedua pendekatan dalam satu sistem tanya jawab berbasis RAG.