Menulis praktik terbaik untuk mengoptimalkan aplikasi RAG

AWS Panduan Preskriptif

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Panduan Preskriptif: Menulis praktik terbaik untuk mengoptimalkan aplikasi RAG

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Menulis praktik terbaik untuk mengoptimalkan aplikasi RAG

Ivan Cui dan Samantha Stuart, Amazon Web Services

Juli 2025 (sejarah dokumen)

Model bahasa besar (LLMs) telah merevolusi bidang kecerdasan buatan dengan kemampuan luar biasa mereka untuk memahami dan menghasilkan teks mirip manusia. Namun, mereka menghadapi batasan yang signifikan: mereka hanya dapat bekerja dengan pengetahuan yang terkandung dalam data pelatihan mereka. Di sinilah Retrieval Augmented Generation (RAG) membantu. Ini menawarkan solusi yang menggabungkan LLMs dengan sumber pengetahuan eksternal, seperti data dan dokumen organisasi Anda. Melalui proses dua tahap yang melibatkan pengambilan informasi dan generasi respons, RAG memungkinkan sistem AI untuk mengakses dan menggabungkan up-to-date informasi dari berbagai sumber, menghasilkan respons yang lebih akurat dan terinformasi yang menjembatani kesenjangan antara pengetahuan model statis dan kebutuhan informasi dunia nyata yang dinamis.

Bagaimana Anda bisa mengoptimalkan konten untuk pengambilan dalam aplikasi berbasis RAG? Panduan ini memberikan praktik terbaik untuk membantu Anda mengoptimalkan gaya pemformatan dan penulisan konten berbasis teks di basis pengetahuan. Mengoptimalkan konten meningkatkan konteks yang membantu aplikasi RAG memahami informasi spesifik tugas dengan lebih akurat. Ketika sistem mengambil konten yang sangat relevan dan akurat, maka kualitas respons LLM meningkat. Mengoptimalkan proses penyampaian konteks pada tingkat sistem disebut rekayasa konteks, dan ini merupakan bagian penting dari arsitektur RAG agen. Dalam RAG agen, satu atau lebih LLMs alasan tambahan dan bertindak atas permintaan asupan sebelum eksekusi RAG. Ini memfasilitasi proses pengiriman informasi multi-langkah. Karena arsitektur RAG tumbuh semakin kompleks, optimasi konten sumber tetap menjadi cara paling langsung untuk memberikan konteks yang jelas. LLMs Praktik terbaik ini dirancang untuk membantu Anda memaksimalkan investasi organisasi Anda dalam aplikasi RAG.

Audiens yang dituju

Panduan ini ditujukan untuk insinyur AI, ilmuwan data, insinyur data, atau pengembang perangkat lunak yang sedang membangun aplikasi LLM dengan satu atau lebih komponen RAG. Untuk memahami konsep dan rekomendasi dalam panduan ini, Anda harus terbiasa dengan database vektor dan petunjuk untuk. LLMs

Tujuan

Rekomendasi dalam panduan ini dapat membantu Anda mencapai hal berikut:

- Tingkatkan akurasi dan relevansi respons yang dihasilkan oleh aplikasi RAG dengan menyediakan dokumen sumber yang terstruktur dengan baik dan kaya secara semantik, dioptimalkan untuk penggunaan token dan redundansi.
- Bantu aplikasi RAG untuk lebih memahami pengetahuan dan konteks khusus domain dengan memberikan definisi dan penjelasan yang jelas dalam dokumen sumber.
- Memfasilitasi pemeliharaan yang lebih mudah dan pembaruan basis pengetahuan untuk aplikasi RAG dengan mengikuti pedoman pemformatan dan penataan yang konsisten di seluruh dokumen sumber.
- Tingkatkan skalabilitas solusi RAG dengan memecah dokumen monolitik besar menjadi unit mandiri yang lebih kecil yang dapat diindeks dan diambil secara efisien.

Memahami LLM dan RAG

Untuk memahami bagaimana meningkatkan kualitas dokumen sumber meningkatkan kualitas respons RAG, Anda harus memahami cara kerja internal LLM. Kekuatan sebenarnya dari LLM terletak pada kemampuan mereka untuk menggunakan mekanisme perhatian diri dan arsitektur transformator. Teknik-teknik canggih ini memungkinkan model untuk secara efektif memproses dan menghubungkan bagian-bagian yang berbeda dari urutan input, terlepas dari posisi atau jarak mereka dalam teks. Kemampuan ini sangat kontras dengan model bahasa tradisional, yang sering berjuang dengan ketergantungan jangka panjang dan pemahaman konteks. Selanjutnya, LLM dilatih pada skala yang belum pernah terjadi sebelumnya. Beberapa model terbesar terdiri dari triliunan parameter dan telah menelan terabyte data tekstual dari berbagai sumber. Skala besar ini memungkinkan LLM untuk mengembangkan pemahaman bahasa yang kaya, menangkap nuansa halus, idiom, dan isyarat kontekstual yang sebelumnya menantang untuk sistem AI. Hasilnya adalah kelas model yang dapat menghasilkan teks yang koheren dan lancar dan menunjukkan kemampuan luar biasa dalam tugas-tugas seperti menjawab pertanyaan, meringkas teks, dan bahkan pembuatan kode.

Untuk menggunakan model ini, kita dapat beralih ke layanan seperti Amazon Bedrock, yang menyediakan akses ke berbagai model pondasi dari Amazon dan penyedia pihak ketiga, termasuk Anthropic, Cohere, dan Meta. Anda dapat menggunakan Amazon Bedrock untuk bereksperimen dengan model canggih, menyesuaikan dan menyempurnakannya, atau memasukkannya ke dalam solusi generatif Anda melalui satu API. AI-powered

Meskipun LLM unggul dalam menangkap pola dan menghasilkan teks yang koheren, mereka sering tidak memiliki akses ke informasi terbaru atau khusus. RAG menggabungkan kekuatan generatif LLM dengan komponen pengambilan yang dapat mengakses dan menggabungkan informasi yang relevan dari sumber eksternal, sebagai bagian dari prompt LLM yang terwujud. Contoh sumber eksternal termasuk Pangkalan Pengetahuan untuk Amazon Bedrock, sistem pencarian cerdas seperti Amazon Kendra, atau database vektor seperti Amazon Service. OpenSearch

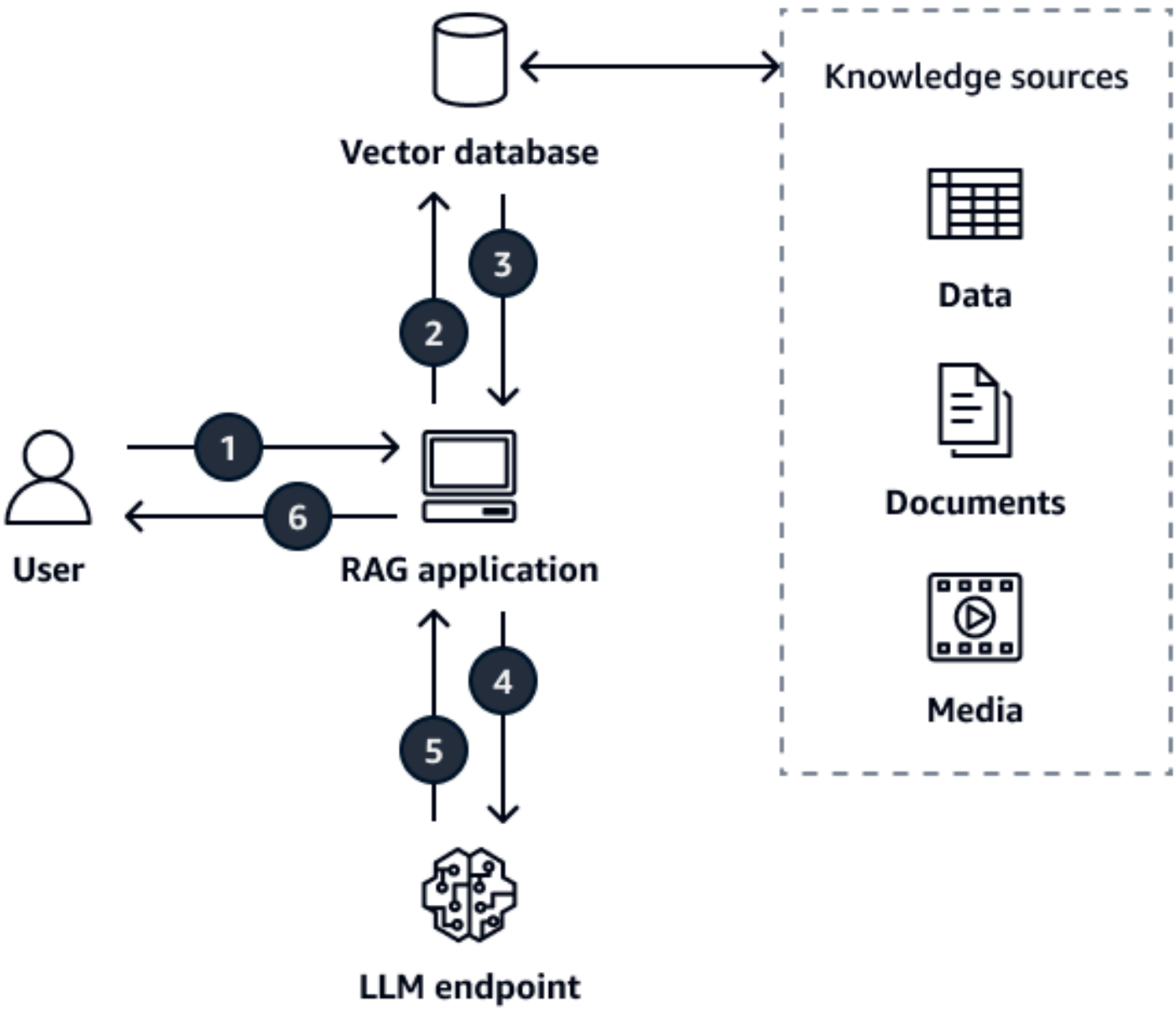

Diagram menjelaskan alur kerja berikut:

1. Pengguna mengirimkan kueri ke aplikasi RAG.
2. Aplikasi RAG menanyakan database vektor yang berisi sumber pengetahuan, seperti dokumen, data, atau media.
3. Aplikasi RAG mengambil informasi yang relevan dari database vektor berdasarkan kesamaan semantik antara kueri dan dokumen yang disimpan.
4. Aplikasi RAG menambah prompt asli dengan konteks yang diambil dan mengirimkannya ke titik akhir LLM.

5. Titik akhir LLM menghasilkan respons dan mengembalikannya ke aplikasi RAG.
6. Aplikasi RAG mengembalikan respons yang dihasilkan kepada pengguna.

Pada intinya, RAG menggunakan proses dua tahap. Pada tahap pertama, model pengambilan mengidentifikasi dan mengambil dokumen atau bagian yang relevan berdasarkan kueri masukan. Model pengambilan ini dapat berupa sistem pengambilan informasi tradisional, model pengambilan yang padat, atau kombinasi keduanya. Pada tahap kedua, informasi yang diambil dan kueri asli dimasukkan ke dalam LLM sebagai template prompt yang sepenuhnya terwujud. LLM sangat bergantung pada kualitas konten sumber yang dikirimkan oleh komponen retriever. Mereka menerapkan mekanisme perhatian diri untuk mengkodekan secara matematis bagaimana konten yang diambil berhubungan dengan tugas. LLM kemudian menghasilkan respons berdasarkan kueri dan informasi yang diambil. Dalam RAG, mengendalikan kualitas dokumen sumber yang diambil merupakan cara langsung untuk meningkatkan representasi internal LLM dari suatu tugas. RAG secara efektif menambah data pelatihan LLM dengan data eksternal yang relevan. Pendekatan ini memungkinkan RAG untuk mengambil keuntungan dari kekuatan LLM dan sistem pengambilan, memungkinkan menghasilkan tanggapan yang lebih akurat dan terinformasi yang menggabungkan pengetahuan saat ini dan khusus.

Vektor dan embeddings

Vektor dan embeddings adalah konsep dasar dalam pembelajaran mesin dan pemrosesan bahasa alami. Vektor adalah objek matematika yang mewakili besaran yang memiliki magnitudo dan arah. Dalam konteks natural language processing (NLP), kata, kalimat, atau dokumen sering direpresentasikan sebagai vektor dalam ruang vektor berdimensi tinggi. Embeddings, di sisi lain, adalah cara untuk mewakili objek seperti kata atau dokumen dalam ruang vektor dimensi yang lebih rendah di mana hubungan antara vektor menangkap kesamaan semantik atau sintaksis. Penyematan kata, misalnya, memungkinkan kata-kata dengan arti yang sama memiliki representasi vektor yang serupa. Ini membantu algoritma untuk memahami dan memproses bahasa secara lebih efektif.

Database vektor

Dalam AI generatif, database vektor adalah database yang menyimpan dan mengelola representasi vektor dokumen, kueri, atau objek lainnya. Ini dirancang untuk menyimpan dan mengambil vektor secara efisien. Ini mendukung operasi yang cepat dan terukur seperti pencarian semantik dan pencocokan kesamaan. Database vektor mengindeks vektor dengan menggunakan struktur data khusus, seperti grafik Hierarchical Navigable Small World (HNSW) atau algoritma Neighbors (KNN).

K-Nearest Struktur data ini memungkinkan pencarian tetangga terdekat yang cepat, sehingga memungkinkan untuk dengan cepat menemukan vektor serupa dalam database.

Pencarian semantik

Pencarian semantik adalah teknik yang meningkatkan relevansi hasil pencarian dengan memahami maksud dan konteks kueri, bukan hanya mencocokkan kata kunci. Dalam istilah teknis, pencarian semantik melibatkan membandingkan representasi vektor dari kueri dan dokumen dalam database untuk menemukan kecocokan yang paling relevan. Strategi pengambilan yang berbeda dapat digunakan untuk pencarian semantik, termasuk namun tidak terbatas pada:

- HNSW — Struktur data berbasis grafik yang mengatur vektor dengan cara yang membuatnya efisien untuk mencari tetangga terdekat.
- KNN — Algoritma yang menemukan K vektor terdekat dengan vektor kueri berdasarkan metrik jarak, seperti kesamaan kosinus.
- Kesamaan kosinus — Ukuran kesamaan antara dua vektor bukan nol yang mengukur kosinus sudut di antara mereka. Hal ini sering digunakan dalam pencarian semantik untuk membandingkan arah vektor dalam ruang dimensi tinggi.
- Locality-sensitive hashing (LSH) — Teknik yang melakukan hash vektor serupa ke ember yang sama atau terdekat dengan probabilitas tinggi. Ini memungkinkan perkiraan pencarian tetangga terdekat, yang bisa lebih cepat daripada pencarian yang tepat di ruang dimensi tinggi.

Tantangan dalam data sumber yang memengaruhi aplikasi RAG

Salah satu tantangan signifikan dalam mengembangkan aplikasi Retrieval-Augmented Generation (RAG) yang optimal terletak pada sifat data mentah atau dokumen yang digunakan. Seringkali, perusahaan menggunakan dokumen yang ada yang dibuat untuk referensi manusia. Dokumen-dokumen ini sering menyertakan hyperlink dan tangkapan layar gambar untuk mempromosikan pemahaman. Namun, elemen-elemen ini menghalangi pengambilan semantik karena batas token kutipan. Ini menghasilkan kinerja retriever yang buruk.

Berikut ini adalah tantangan dokumen mentah yang paling umum untuk aplikasi RAG yang optimal:

- Kurangnya pemformatan dan metadata terstruktur — Dokumen mentah dapat kekurangan judul bagian, subjudul, atau metadata yang jelas. Hal ini membuat sulit untuk mengidentifikasi dan mengekstrak informasi yang relevan. Misalnya, dokumen yang panjang tanpa judul yang jelas dapat menyulitkan untuk menentukan konteks informasi tertentu.
- Bahasa informal dan tidak konsisten — Dokumen mentah sering mengandung bahasa informal atau terminologi yang tidak konsisten. Ini dapat membingungkan model RAG. Misalnya, singkatan yang tidak didefinisikan dalam dokumen atau sudah dikenal oleh LLM dapat digunakan di seluruh dokumen.
- Verbositas dan redundansi — Dokumen mentah mungkin bertele-tele dan berisi informasi yang tidak perlu atau berlebihan. Ini dapat membanjiri model RAG, yang mengarah ke respons yang kurang ringkas dan relevan. Contohnya termasuk dokumen yang mengulangi informasi yang sama beberapa kali atau beberapa dokumen yang berisi informasi serupa atau kontradiktif.
- Istilah dan frasa ambigu — Dokumen mentah dapat berisi istilah atau frasa ambigu yang dapat ditafsirkan dalam berbagai cara. Ambiguitas ini dapat menyebabkan salah tafsir oleh model RAG dan tanggapan yang tidak akurat. Misalnya, dokumen yang menggunakan istilah dengan banyak arti dapat menghasilkan respons yang tidak selaras dengan makna yang dimaksudkan.
- Injeksi elemen grafis dan hyperlink — Dokumen mentah yang berisi grafik dan informasi hyperlink bekerja dengan baik untuk konsumsi manusia. Namun, elemen-elemen ini dapat menggunakan batas token pengambilan. Hasilnya adalah kutipan mungkin tidak lengkap. Misalnya, grafik dan hyperlink URLs dikembalikan sebagai bagian dari pengambilan, yang menggunakan token pengambilan, dan informasi kunci dari paragraf berikutnya tidak ada.
- Kurangnya pengetahuan atau konteks khusus domain - Dokumen mentah dapat kekurangan pengetahuan atau konteks khusus domain yang diperlukan untuk pembuatan yang akurat. Ini

dapat membatasi kemampuan model RAG untuk menghasilkan respons yang relevan dan akurat. Contohnya adalah dokumen yang mereferensikan konsep khusus tanpa memberikan konteks. Ini mungkin mengarah pada tanggapan yang tidak berarti dalam domain yang diberikan.

Meskipun daftar ini tidak komprehensif, ini memberikan titik awal bagi perusahaan untuk memikirkan apa yang tidak berfungsi dan mengapa. Dokumen mungkin memiliki satu atau lebih dari tantangan ini. Kunci untuk mengoptimalkan aplikasi RAG adalah dengan menggunakan satu set dokumen yang mematuhi praktik terbaik penulisan yang mengoptimalkan pengambilan.

Praktik terbaik dokumentasi untuk aplikasi RAG

Mengembangkan aplikasi Retrieval-Augmented Generation (RAG) yang sukses memerlukan pertimbangan yang cermat dari berbagai faktor terkait dokumen untuk mengoptimalkan kinerjanya. Praktik terbaik di bagian ini dikuratori berdasarkan pengalaman membangun sistem RAG dengan banyak pemimpin organisasi. Berikut ini adalah beberapa praktik terbaik utama untuk dokumen untuk meningkatkan efektivitas aplikasi RAG Anda:

- Gunakan judul dan subpos dengan benar — Mengatur konten Anda dengan judul dan subpos yang jelas meningkatkan keterbacaan dan membantu model RAG memahami struktur dokumen Anda. Praktik ini memungkinkan model untuk menavigasi dan mengekstrak informasi dari dokumen dengan lebih baik, yang meningkatkan kualitas respons yang dihasilkan.
- Pastikan penomoran berurutan — Saat menggunakan daftar bernomor, penting untuk mempertahankan penomoran yang tepat untuk menghindari kebingungan. Pastikan bahwa setiap item daftar diberi nomor secara berurutan tanpa melewatkan angka. Ini membantu menjaga kejelasan dan koherensi dalam konten Anda.
- Tambahkan transisi antara item daftar - Menyediakan transisi antara item dalam daftar berpoin atau bernomor membantu memandu LLM melalui konten. Misalnya, Anda dapat menggunakan frasa seperti “Setelah menyelesaikan langkah 2, lakukan...” untuk menghubungkan ide dan meningkatkan arus informasi.
- Ganti tabel — Hindari menggunakan tabel. Format informasi ini dalam daftar berpoin multi-level atau dalam sintaks tingkat datar. Sintaks tingkat datar adalah mengatur elemen atau item pada tingkat hierarki yang sama, tanpa tingkat subordinasi bersarang. Struktur ini LLMs membantu mencerna informasi. Karena sebagian besar dokumen yang diindeks dibaca dari kiri ke kanan, sintaks tingkat datar memungkinkan informasi untuk mengikuti lebih koheren tanpa perlu referensi dimensi tambahan. Format ini lebih kondusif untuk aplikasi RAG karena menyajikan informasi secara terstruktur dan mudah dicerna.
- Informasi grafis pra-proses untuk efisiensi — Multi-modal LLMs dapat menyerap gambar dan teks. Kurangi resolusi gambar, hapus gambar yang berlebihan, dan jelaskan konten elemen grafis dalam format teks. Langkah-langkah ini meningkatkan konteks yang bermakna, menghindari penggunaan token yang tidak perlu, dan meningkatkan aksesibilitas untuk model RAG.
- Tambahkan pembuka sesi untuk pertanyaan umum — Saat menangani pertanyaan atau tugas umum, seperti “Bagaimana cara memesan perangkat lunak?” , tambahkan starter sesi yang mentransisikan pembaca ke dalam proses. Misalnya, Anda dapat menambahkan “Jika Anda ingin

memesan perangkat lunak, ikuti langkah-langkah di bawah ini...". Ini membantu menciptakan pencocokan semantik yang tinggi, yang membantu LLM membangun respons yang kohesif.

- Tambahkan ringkasan ke setiap bagian — Setelah setiap judul atau subjudul, tambahkan ringkasan singkat dan ringkas dari konten di bagian itu. Ini dapat meningkatkan cakupan semantik dan memperkuat poin-poin penting. Hal ini meningkatkan akurasi pencarian kesamaan dalam ruang embedding, sehingga meningkatkan kinerja aplikasi RAG. Ini sangat membantu jika dokumen dimaksudkan untuk LLM dan konsumsi manusia atau jika tabel dan elemen grafis diperlukan.
- Disambiguasi — Dokumen harus ringkas dan fokus. LLMs menghasilkan tanggapan berdasarkan kutipan yang diambil, sehingga disambiguasi membantu model menggunakan informasi yang jelas dan relevan. Ini menghasilkan tanggapan yang lebih akurat dan informatif.
- Mendefinisikan singkatan dan mengatur konteks — LLMs dilatih pada sejumlah besar data internet, dan sebagian besar waktu, mereka tidak memiliki konteks dokumen internal perusahaan. Oleh karena itu, menetapkan konteks, mendefinisikan singkatan, dan menghindari atau mendefinisikan terminologi khusus perusahaan membantu LLM memahami data perusahaan Anda. Ini membantu LLM untuk menjawab pertanyaan dengan lebih akurat dan dapat membantu mencegah halusinasi.
- Merestrukturisasi dokumen besar menjadi dokumen yang lebih kecil untuk penandaan dan pengindeksan yang efisien — Hindari pengindeksan dokumen besar yang berisi beberapa subtopik. Pertimbangkan untuk membagi dokumen besar menjadi dokumen yang lebih kecil dan mandiri yang memiliki judul yang jelas. Ini meningkatkan pengindeksan dan penandaan.

Pertanyaan yang Sering Diajukan

Mengapa penting untuk mengoptimalkan dokumen untuk aplikasi RAG?

Dokumen mentah sering ditulis untuk konsumsi manusia tanpa mempertimbangkan persyaratan sistem AI canggih, seperti aplikasi Retrieval Augmented Generation (RAG). Mengoptimalkan dokumen dengan mengikuti praktik terbaik dapat secara signifikan meningkatkan kinerja dan akurasi aplikasi RAG dengan memberikan informasi yang terstruktur, tidak ambigu, dan relevan dengan model.

Apa saja tantangan umum dengan dokumen mentah yang dapat menghambat kinerja RAG?

Beberapa tantangan utama termasuk kurangnya pemformatan dan metadata terstruktur, bahasa informal atau tidak konsisten, verbositas dan redundansi, istilah dan frasa yang ambigu, penyertaan elemen hyperlink, dan kurangnya konteks khusus domain. Masalah-masalah ini dapat membingungkan model RAG dan menyebabkan tanggapan yang tidak akurat atau tidak relevan. Untuk informasi selengkapnya, lihat Tantangan dalam data sumber yang memengaruhi aplikasi RAG dalam panduan ini.

Bagaimana penggunaan judul dan subpos dapat meningkatkan kinerja RAG?

Judul dan subjudul yang jelas membantu model RAG memahami struktur dan konteks konten. Hal ini memungkinkan mereka untuk lebih menavigasi dan mengekstrak informasi yang relevan dari dokumen dan meningkatkan kualitas tanggapan yang dihasilkan. Untuk informasi selengkapnya, lihat Praktik terbaik Dokumentasi untuk aplikasi RAG dalam panduan ini.

Mengapa disarankan untuk mengganti informasi tabel dengan sintaks tingkat datar?

Ini bisa menjadi tantangan bagi model RAG untuk menafsirkan tabel karena mereka membutuhkan pemahaman tentang struktur dua dimensi. Menyajikan informasi tabel dalam sintaks tingkat datar

atau daftar berpoin membantu model untuk lebih mudah memproses informasi, yang mengarah ke kinerja yang lebih baik. Untuk informasi selengkapnya, lihat Praktik terbaik Dokumentasi untuk aplikasi RAG dalam panduan ini.

Bagaimana menambahkan ringkasan dapat meningkatkan kinerja RAG?

Menyertakan ringkasan singkat di awal setiap bagian atau subbagian dapat meningkatkan cakupan semantik dan memperkuat poin-poin penting. Ini meningkatkan akurasi pencarian kesamaan dalam ruang embedding, yang pada akhirnya meningkatkan kinerja aplikasi RAG. Untuk informasi selengkapnya, lihat Praktik terbaik Dokumentasi untuk aplikasi RAG dalam panduan ini.

Mengapa penting untuk mendefinisikan singkatan dan mengatur konteks untuk LLMs?

LLMs Dilatih pada berbagai data, tetapi mereka tidak memiliki konteks untuk singkatan atau terminologi khusus perusahaan. Mendefinisikan singkatan dan memberikan konteks membantu LLMs memahami dan merespons dengan lebih akurat. Ini dapat membantu mencegah halusinasi atau salah tafsir. Untuk informasi selengkapnya, lihat Praktik terbaik Dokumentasi untuk aplikasi RAG dalam panduan ini.

Langkah dan sumber daya selanjutnya

Panduan ini membahas serangkaian tantangan tingkat dokumen untuk aplikasi RAG dan praktik terbaik untuk menguranginya. Pembelajaran ini dikuratori dari wawancara dan diskusi dengan para pemimpin industri, dan mereka didukung oleh kasus penggunaan perusahaan.

Untuk mulai mengoptimalkan dokumen Anda untuk aplikasi RAG, kami sarankan untuk melakukan audit terhadap dokumen Anda yang ada. Identifikasi area yang menimbulkan tantangan bagi aplikasi RAG. Contohnya termasuk kurangnya struktur, bahasa yang ambigu, atau penggunaan elemen grafis yang berlebihan. Prioritaskan dokumen yang sering diakses atau penting untuk operasi bisnis Anda. Berkolaborasi dengan ahli materi pelajaran untuk menerapkan praktik terbaik dalam panduan ini. Pastikan dokumen direstrukturisasi dengan judul yang jelas, bahasa yang ringkas, dan elemen pengaturan konteks. Untuk dokumen baru, buat pedoman dan templat yang memastikan konsistensi dan membantu penulis mematuhi praktik terbaik. Selain itu, pertimbangkan untuk berinvestasi dalam alat atau layanan yang dapat mengotomatiskan aspek proses pengoptimalan dokumen, seperti menggunakan AI generatif untuk merestrukturisasi dokumen. Dengan mengambil pendekatan proaktif untuk pengoptimalan dokumen, Anda dapat membuka potensi penuh aplikasi RAG dan mendorong hasil yang lebih akurat dan berwawasan luas di seluruh organisasi Anda.

Sumber daya berikut dapat membantu Anda memahami dan membangun aplikasi RAG di organisasi Anda.

Sumber daya

AWS dokumentasi

- Memilih database AWS vektor untuk kasus penggunaan RAG (Panduan AWS Preskriptif)
- Terapkan kasus penggunaan RAG AWS dengan menggunakan Terraform dan Amazon Bedrock AWS (Panduan Preskriptif)
- Kembangkan asisten berbasis obrolan AI generatif tingkat lanjut dengan menggunakan RAG dan prompt (Prescriptive Guidance) ReAct AWS
- Ambil data dan hasilkan respons AI dengan Pangkalan Pengetahuan Amazon Bedrock (dokumentasi Amazon Bedrock)
- Retrieval Augmented Generation (dokumentasi Amazon SageMaker AI)
- Pengambilan opsi dan arsitektur Augmented Generation pada AWS(Panduan Preskriptif)AWS

AWS Sumber daya lainnya

- Buat asisten multimodal dengan RAG canggih dan Amazon Bedrock (AWS posting blog)

Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan umpan RSS.

Perubahan	Deskripsi	Tanggal
Publikasi awal	—	Juli 24, 2025

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.