INSTITUTO DE PESQUISAS TECNOLÓGICAS

COMUNICAÇÃO TÉCNICA

N° 180011

Análise de estratégias avançadas de chunking para aprimoramento de RAG

Adriana Camargo Brito
Cassia de Souza

Resumo expandido apresentado no CONGRESSO DE MODELAGEM, SIMLAÇÃO COMPUTACIONAL E IA DO IPT, 1, 2025, São Paulo. 3 p.

Instituto de Pesquisas Tecnológicas do Estado de São Paulo S/A - IPT
Av. Prof. Almeida Prado, 532 | Cidade Universitária ou
Caixa Postal 0141 | CEP 01064-970
São Paulo | SP | Brasil | CEP 05508-901
Tel 11 3767 4374/4000 | Fax 11 3767-4099

www.ipt.br

ANÁLISE DE ESTRATÉGIAS AVANÇADAS DE *CHUNKING* PARA APRIMORAMENTO DE RAG

Adriana Camargo de Brito[1], Cassia de Souza Carvalho[2]

[1, 2] Instituto de Pesquisas Tecnológicas do Estado de São Paulo, Seção de Inteligência Artificial e Analytics
E-mail para contato: adrianab@ipt.br [1], cassiasouza@ipt.br [2]

RESUMO – Este estudo avalia o impacto de várias técnicas de chunking em arquiteturas de Retrieval-Augmented Generation (RAG). Foram comparadas quatro estratégias (Recursiva, Semântica, SDPM e LATE) aplicadas a um documento sobre cidades inteligentes. Os resultados mostram um trade-off: trechos menores reduziram a latência (≈2,5 s), mas comprometeram fidelidade e relevância; trechos maiores aumentaram a consistência (0,9), porém com maior tempo de resposta (≈14 s). Isso reforça a importância de equilibrar granularidade e desempenho em RAGs.

1 INTRODUÇÃO

Grandes Modelos de Linguagem (LLMs) possuem capacidade avançada de compreender e gerar texto, mas restringem-se ao conhecimento presente em seus dados de treinamento. Para suprir essa limitação, surgiu a técnica *Retrieval-Augmented Generation* (RAG) [1], que permite incorporar informações externas como contexto. Em RAGs, os documentos são segmentados, transformados em vetores densos e indexados em bancos de dados vetoriais, possibilitando a recuperação semântica de trechos relevantes a serem fornecidos a LLMs. Com o avanço das pesquisas, surgiram estratégias que ampliam a precisão das respostas, voltadas a: segmentação de documentos, reordenação de trechos recuperados, mecanismos híbridos de busca (que combinam vetores densos e esparsos) e grafos de conhecimento. Este trabalho analisa o efeito de técnicas avançadas de *chunking* no desempenho de RAG, para identificar até que ponto diferentes granularidades de segmentação influenciam as respostas.

2 METODOLOGIA

2.1 Construção da RAG

Foi desenvolvida uma RAG em Python com o *framework LangChain* [2], a partir de documento PDF com oito páginas A4, sobre conectividade em cidades inteligentes. Textos, tabelas e descrições de imagens foram extraídos com a biblioteca *PyMuPDF4LLM* [3] e segmentados em *chunks* com estratégia tradicional (pontuação e limite de caracteres) e três estratégias avançadas (item 2.1). Os *chunks* foram convertidos em vetores densos e armazenados no banco de dados *Qdrant* [4], que suporta busca híbrida, combinando vetores densos (similaridade semântica) e esparsos (palavras-chave). Os resultados são integrados por *Reciprocal Rank Fusion* [5]**,** que soma os *scores* recíprocos para a ordenação final *dos chunks.* A RAG foi configurada para retornar cinco *chunks* com maior similaridade à pergunta do usuário. O desempenho da RAG foi avaliado conforme item 2.2. Foram utilizados o modelo *paraphrase-multilingual-mpnet-base-v2* [6] para vetores densos e BM-25 [7] para vetores esparsos. Para gerar respostas foi usado o modelo *llama-3.1-8b-instant* [8] e para avalia-las, o

gpt-oss-120b [9], procedentes de famílias diferentes para minimizar o fenômeno conhecido como "narcisismo de LLM" [10].

2.3 Estratégias de *chunking*

Foram usadas as seguintes estratégias: (i) Caractere Recursivo (referência), divisão por pontuação e caracteres, com limite de tamanho fixo; (ii) Semântico, que cria *chunks* com o mesmo assunto; (iii) *Semantic Double-Pass Merging* (SDPM) [11], que identifica mudanças de assunto pela queda de similaridade semântica entre sentenças e (iv) LATE [12], que agrupa sentenças até atingir um limite de similaridade ou tamanho do *chunk.* Os parâmetros empregados nas técnicas foram: em (i), 512 caracteres de tamanho e 75 caracteres de sobreposição; em (ii, iii, iv), 512 *tokens* de tamanho e o modelo de vetores densos do item 2.2; em (iii, iv), 0.6 de similaridade.

2.4 Especificações do ambiente de execução dos experimentos

Os experimentos foram executados em ambiente Google Colab [13], com a seguinte configuração: CPU Intel(R) Xeon(R) CPU @ 2.20GHz e 14 GB de memória RAM. A inferência dos modelos de LLM foi realizada no servidor em nuvem Groq [14].

2.5 Mecanismos de avaliação

Para a avaliação foram consideradas cinco perguntas referentes ao assunto do contexto, escritas por usuário da RAG. As perguntas foram encaminhadas para a RAG, que teve seu desempenho avaliado com base nas seguintes métricas: (i) Similaridade, que considera o valor da distância do cosseno entre a pergunta e o *chunk (*é apresentada a pontuação média de todos os *chunks* e a pontuação do primeiro *chunk*); (ii) Fidelidade, o quanto a resposta gerada é fiel ao contexto, (iii) Relevância, se o *chunk* é relevante à resposta. As métricas (ii) e (iii) empregam um LLM para o julgamento via *prompt zero-shot.* As métricas (i, ii, iii) variam entre 0 e 1, com escala crescente de desempenho. Além disso, são apresentadas estatísticas dos *chunks*, como latência para fornecer a resposta, quantidade máxima de caracteres do *chunk* e tamanho médio.

RESULTADOS E DISCUSSÃO

Na Tabela 1 são apresentadas as estatísticas dos *chunks* e as métricas de avaliação do desempenho da RAG.

Tabela 1: Métricas por tipo de chunker

Chunker	Num. *chunks*	Tamanho máximo de *chunk*	Tamanho médio de *chunks*	Latência média (s)	Similaridade		Fidelidade média	Relevância média
					Média	Média primeiro *chunk*		
Recursivo	41	510	429	2,9	0,6	0,9	0,7	0,7
SDPM	15	2219	1707	13,8	0,6	0,9	0,9	0,9
LATE	10	2381	1138	14,1	0,6	0,8	0,9	0,9
Semântico	71	1057	240	2,4	0,5	0,8	0,7	0,8

Os métodos SDPM e LATE alcançaram maior fidelidade e relevância (0,9), indicando que trechos mais extensos (da ordem de 2000 caracteres) favoreceram a recuperação de contexto consistente. Em contraste, com os métodos Recursivo e Semântico foram produzidos mais *chunks* (41 a 71), representando trechos menores (510 a 1057 caracteres), proporcionando uma latência significativamente reduzida (entre 2,4 e 2,0 s). Porém, com tais métodos, houve redução de 10 a 20% na fidelidade e relevância, possivelmente em decorrência de uma fragmentação excessiva do contexto.

A similaridade média manteve-se estável (entre 0,5 e 0,6), refletindo a diversidade semântica dos trechos recuperados. Entretanto, a análise do primeiro *chunk* sugere que a segmentação SDPM e a recursiva preservam maior proximidade imediata com a consulta, embora isso não se traduza em maior fidelidade global, no caso da recursiva.

Esses achados evidenciam um *trade-off* entre granularidade e desempenho, ou seja, com *chunking* mais fino, com trechos curtos, ocorre uma otimização da velocidade de resposta, mas ocorre um comprometimento da consistência semântica. Por outro lado, com o *chunking* mais grosso, com trechos mais longos, há uma maior preservação do contexto, porém aumenta latência. Essa dinâmica corrobora estudos anteriores que destacam a importância do equilíbrio entre tamanho do *chunk* e eficiência de recuperação [15–16].

CONCLUSÃO

Este trabalho analisou o impacto de diferentes estratégias de *chunking* no desempenho de uma arquitetura RAG. Os resultados demonstraram que as abordagens SDPM e LATE, ao produzirem *chunks* mais extensos, alcançaram maiores níveis de fidelidade e relevância (0,9), indicando que a preservação de contextos amplos favorece respostas mais consistentes. Em contrapartida, as estratégias Recursiva e Semântica, ao fragmentarem o documento em maior número de trechos menores, reduziram a latência de resposta, mas perderam de 10 a 20% em fidelidade e relevância. Há um trade-off entre granularidade e qualidade das respostas em RAGs: *chunking* mais fino otimiza velocidade, *chunking* mais grosso preserva contexto e consistência semântica.

REFERÊNCIAS

[1] LEWIS P, PEREZ E, PIKTUS A, PETRONI F, KARPUKHIN V, GOYAL N, KÜTTLER H, LEWIS M, YIH W, ROCKTÄSCHEL T, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv*, 2021.
[2] LANGCHAIN, INC. *LangChain*. 2025.
[3] LANGCHAIN, INC. *PyMuPDF4LLM Documentation*. 2025.
[4] QDRANT, INC. *Qdrant Documentation*. 2025.
[5] COR M, CLARKE C, BUETTCHER S. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. *Proc. SIGIR*, p. 758-759, 2009.
[6] WOLF T, DEBUT L, SANH V, CHAUMOND J, DELANGUE C, MOI A, CISTAC P, RAULT T, LOUF R, FUNTOWICZ M, HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv*, 2020.
[7] ROBERTSON S E, WALKER S, HANCOCK-BEAULIEU M M, GATFORD M, PAYNE A, LI B. Automatic ad hoc, filtering, VLC, and interactive tracks. *Proc. TREC*, 1999.
[8] TOUVRON H, LAVRIL T, IZACARD G, MARTINET X, LACHAUX M, LACROIX T, ROZIÈRE B, GOYAL N, HAMBRO E, AZHAR F, LLaMA: Open and Efficient Foundation Language Models. *arXiv*, 2023.
[9] BROWN T B, MANN B, RYDER N, SUBBIAH M, KAPLAN J, DHARIWAL P, NEELAKANTAN A, SHYAM P, SESHADRI G, KHAN C, Language Models are Few-Shot Learners. *Neural Inf. Process. Syst.*, v. 33, p. 1877-1901, 2020.
[10] LIU Y, MOOSAVI N S, LIN C. LLMs as Narcissistic Evaluators: When Ego Inflates Evaluation Scores. *arXiv*, 2024.
[11] MINHAS B, NIGAM S. *Chonkie: A no-nonsense fast, lightweight, and efficient text chunking library*. 2025.
[12] GÜNTHER M, MOHR I, WILLIAMS D J, WANG B, XIAO H. Late Chunking: Contextual Chunk Embeddings Using Long-Context Embedding Models. *arXiv*, 2025.
[13] GOOGLE, INC. *Google Colaboratory*. 2025.
[14] GROQ, INC. *Groq*. 2025
[15] KUSH, J. et al. Introducing a new hyper-parameter for RAG: Context Window Utilization. *ArXiv*, 2024.
[16] MEROLA, C. et al. Reconstructing Context: Evaluating Advanced Chunking Strategies for Retrieval-Augmented Generation. *ArXiv*, 2025.