# RAG Corporativo com HyDE: Uma Experiência Prática de Chatbot B2B com Conhecimento Proprietário

Isaac Silva
NEOXS TECH
São Paulo/SP, Brazil
isaacorcine37@gmail.com

Heitor Costa
Universidade Federal de Lavras
Lavras/MG, Brazil
heitor@ufla.br

## Resumo

Neste artigo, é relatada a experiência de construção e implantação de um *chatbot B2B* baseado em *Retrieval-Augmented Generation* (RAG) para a empresa NEOXS TECH[1]. Nesse *chatbot*, foram utilizadas a API do Gemini para *embeddings* e geração de linguagem, a biblioteca LangChain para processamento de documentos e indexação vetorial com FAISS (*Facebook AI Similarity Search*). Além disso, adota-se a técnica de *Hypothetical Document Embeddings* (HyDE), combinada com *Maximum Marginal Relevance* (MMR), para mitigar a lacuna semântica entre as perguntas dos usuários e os documentos corporativos. Assim, são descritos o contexto de negócios, as decisões arquiteturais, os desafios enfrentados e os resultados qualitativos observados, com o objetivo de compartilhar lições aprendidas úteis para equipes que enfrentem cenários semelhantes.

## Palavras-chave

RAG, HyDE, Chatbot Corporativo, LangChain, Gemini, B2B, Conhecimento Proprietário

## 1 Introdução

Empresas que atuam em mercados B2B (*Business-to-Business*) acumulam grandes volumes de conhecimento proprietário, como manuais de treinamento, *playbooks* de vendas, políticas internas e conteúdos de capacitação. Tornar esse conhecimento acessível de forma conversacional e personalizada representa um relevante diferencial competitivo, pois reduz o tempo de *onboarding* de novos colaboradores e amplia a consistência das respostas comerciais.

A NEOXS TECH identificou essa oportunidade e demandou o desenvolvimento de um *chatbot* capaz de responder perguntas de negócio com base exclusivamente no acervo documental interno da empresa. O principal requisito funcional é que o *chatbot* não utilize conhecimento externo ao *corpus* corporativo: respostas genéricas ou inventadas seriam prejudiciais à credibilidade da solução junto aos usuários finais. Para atender a esses requisitos, adotou-se a abordagem de *Retrieval-Augmented Generation* (RAG) [3], que separa a etapa de recuperação de documentos relevantes da etapa de geração de linguagem natural. Contudo, a literatura aponta uma limitação conhecida nessa abordagem: a lacuna semântica entre perguntas curtas e situacionais dos usuários e os trechos de documentos longos e especializados [2]. Para endereçar esse problema, foi incorporada a técnica HyDE (*Hypothetical Document Embeddings*) [2], que gera um documento hipotético a partir da pergunta antes de realizar a busca vetorial, aproximando semanticamente a consulta dos documentos relevantes.

[1]https://neoxs.com.br/

Este artigo está organizado da seguinte forma. Na Seção 2, são apresentados o contexto de negócio e os requisitos do sistema. Na Seção 3, é descrita a arquitetura da solução. Na Seção 4, discutem-se os desafios e as lições aprendidas. Na Seção 5, são apresentados os resultados qualitativos. Na Seção 6, apresentam-se as conclusões do trabalho.

## 2 Contexto e Requisitos

A NEOXS TECH é uma empresa de tecnologia que atua no segmento B2B, fornecendo soluções e serviços a outras organizações. Seu processo de capacitação comercial envolve materiais extensos em formato PDF e em texto que descrevem estratégias de vendas, objeções comuns, propostas de valor e procedimentos internos. A consulta a esses materiais era manual e fragmentada, o que resultava em inconsistências nas respostas e na dependência de profissionais seniores.

Os requisitos centrais levantados para a solução foram (i) o *chatbot* deveria responder baseando-se exclusivamente nos documentos internos carregados, (ii) quando a base documental não contivesse informação suficiente, o *chatbot* deveria admitir a limitação, em vez de inventar uma resposta, (iii) o *chatbot* deveria manter contexto de conversa para suportar diálogos de múltiplos turnos, (iv) o acesso deveria ser controlado por autenticação de usuários e (v) a solução deveria ser implantável de forma gerenciada, com separação entre *frontend*, *backend* e banco de dados.

## 3 Arquitetura da Solução

### 3.1 Visão Geral

A solução foi desenvolvida como uma aplicação web composta por três serviços orquestrados via Docker Compose[2]: i) um banco de dados PostgreSQL[3] para persistência de usuários e histórico de mensagens; ii) um *backend* em FastAPI[4] (Python[5]) responsável pela autenticação, lógica de conversação e motor de RAG; e iii) um *frontend* em Next.js[6] que expõe a interface de *chat* e as telas de registro e *login*.

O *backend* implementa autenticação baseada em JWT[7] (*JSON Web Token*), gerencia sessões de usuário com SQLAlchemy[8]/ PostgreSQL e expõe dois *endpoints* principais: `POST/chat/query`, que processa mensagens e retorna respostas do chatbot, e `GET/chat/`

[2]https://docs.docker.com/compose/
[3]https://www.postgresql.org/
[4]https://fastapi.tiangolo.com/
[5]https://www.python.org/
[6]https://nextjs.org/
[7]https://www.jwt.io/
[8]https://www.sqlalchemy.org/

`history`, que recupera o histórico de conversas do usuário autenticado. A fim de proporcionar uma compreensão visual e holística do ecossistema, a solução foi estruturada em quatro camadas lógicas interdependentes, cujas interações e fluxos programáticos de dados estão detalhados na Figura 1: i) Apresentação (*Presentation*); ii) Lógica de Negócio (*Business Logic*); iii) Acesso a Dados (*Data Access*); e iv) Infraestrutura (*Infrastructure*). Esse modelo de isolamento garante que as requisições originadas na interface do usuário transitem de forma segura e auditável pelas regras de negócio e pela persistência de dados.

## 3.2 Motor de RAG com HyDE e MMR

O motor de RAG é o componente central da solução. Na inicialização, ele percorre a pasta de documentos corporativos aceitando arquivos `.pdf` e `.txt`, extrai o texto com a biblioteca pypdf[9], divide o conteúdo em *chunks* com o `RecursiveCharacterTextSplitter` do LangChain[10] (tamanho de 1.200 caracteres, sobreposição de 200 caracteres) e constrói um índice vetorial FAISS[11] utilizando o modelo de *embeddings* text-embedding-004 do Google[12].

No processamento de cada pergunta, o *chatbot* executa um *pipeline* de recuperação em duas etapas, ambas baseadas na abordagem de *Maximum Marginal Relevance* (MMR), para garantir diversidade nos resultados retornados. A primeira etapa realiza uma busca direta com base na pergunta original do usuário. A segunda etapa implementa a técnica HyDE [2]: o modelo de linguagem é solicitado a gerar um parágrafo hipotético plausível que responda à pergunta. Esse texto é utilizado como nova consulta de busca. Os resultados de ambas as etapas são combinados e deduplicados, formando um conjunto de até oito *chunks* relevantes.

A motivação para adotar HyDE está diretamente relacionada à lacuna semântica descrita por Gao et al. [2]: perguntas situacionais e curtas ("Como responder a uma objeção de preço?") têm representação vetorial muito diferente dos trechos documentais que as respondem, que tendem a ser mais longos e técnicos. Ao transformar a pergunta em um documento hipotético, com linguagem semelhante à dos materiais corporativos, o HyDE reduz essa distância semântica e melhora o *recall* na etapa de recuperação. Se o índice vetorial não retornar resultados por causa de uma falha no serviço de *embeddings* ou da ausência de *matches*, o *chatbot* recorre a uma busca lexical simples nos *chunks* armazenados em memória, garantindo resiliência operacional.

Para integrar esse motor ao ecossistema da aplicação, o fluxo de comunicação entre os componentes é assíncrono e orquestrado. Quando o usuário submete uma pergunta via interface (*Frontend*), a requisição HTTP (*Hypertext Transfer Protocol*) é encaminhada ao roteador de *chat* no *Backend* (FastAPI). Antes de acionar o motor de RAG, uma dependência de segurança intercepta a requisição para validar o *token* JWT e recuperar o registro do usuário no banco de dados relacional (PostgreSQL). Simultaneamente, as últimas interações daquela sessão são extraídas do banco de dados para compor o histórico conversacional. O roteador repassa a consulta e o histórico ao motor de RAG, que realiza as chamadas à API do Gemini para a geração de *embeddings* e síntese da resposta. Por fim, a resposta consolidada é persistida no modelo de dados da conversa e devolvida ao *Frontend*, completando o ciclo de fluxo de dados de ponta a ponta.

## 3.3 Construção do *Prompt* e Controle de *Persona*

Os *chunks* recuperados, junto com as últimas cinco mensagens do histórico de conversa, são inseridos em um *prompt*[13] estruturado que instrui o Gemini[14] a assumir a *persona* de consultor de vendas sênior e a utilizar exclusivamente o conhecimento presente no contexto fornecido. O *prompt* proíbe explicitamente o uso de conhecimento geral externo aos documentos, instrui o modelo a reproduzir termos e bordões presentes nos materiais de treinamento e estabelece que, na ausência de informação suficiente, o modelo deve declinar a resposta de forma transparente, em vez de inventar conteúdo. Essa estratégia de controle de *persona* por meio de instruções de *chatbot* é fundamental para atender ao requisito de fidelidade documental: o modelo atua como recuperador e sintetizador do *corpus* corporativo, não como um assistente generalista.

# 4 Desafios e Lições Aprendidas

A seguir, são apresentados os principais desafios técnicos e as decisões de projeto observados ao longo do desenvolvimento da solução. Esses aspectos refletem limitações práticas identificadas durante a implementação e a experimentação do *chatbot*, bem como as estratégias adotadas para mitigá-las. Os itens destacam questões relacionadas à recuperação semântica de informações, definição da estratégia de fragmentação dos documentos, adaptação à evolução dos modelos de linguagem, controle de custos e limites de contexto, além de mecanismos para garantir a fidelidade das respostas ao *corpus* corporativo:

- **Lacuna Semântica na Recuperação**. O principal desafio técnico enfrentado foi exatamente a lacuna semântica descrita na literatura [2]. Nas primeiras versões do *chatbot*, perguntas situacionais curtas retornavam *chunks* pouco relevantes, resultando em respostas genéricas ou na ativação indevida do *fallback* de "informação não encontrada". A adoção de HyDE reduziu significativamente esse problema, pois o texto hipotético gerado pelo próprio modelo de linguagem incorpora vocabulário semelhante ao dos documentos corporativos;
- **Estratégia de *Chunking***. O tamanho e a sobreposição dos *chunks* impactam diretamente a qualidade das respostas. *Chunks* muito pequenos perdem contexto local e *chunks* muito grandes reduzem a precisão da recuperação. Após a experimentação, chegou-se ao valor de 1.200 caracteres, com sobreposição de 200 caracteres, o que preserva frases-chave e bordões completos sem comprometer o *recall*. A sobreposição é especialmente importante para trechos onde conceitos fundamentais aparecem no final de um *chunk* e são desenvolvidos no início do próximo;
- **Seleção Dinâmica de Modelo**. A API (*Application Programming Interface*) do Gemini evolui com relativa frequência,

[9] https://pypi.org/project/pypdf/
[10] https://www.langchain.com/
[11] https://faiss.ai/index.html
[12] https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/tune-embeddings?hl=pt-br
[13] Infelizmente, não será possível apresentar o *prompt* em decorrência da confidencialidade
[14] https://gemini.google.com/app

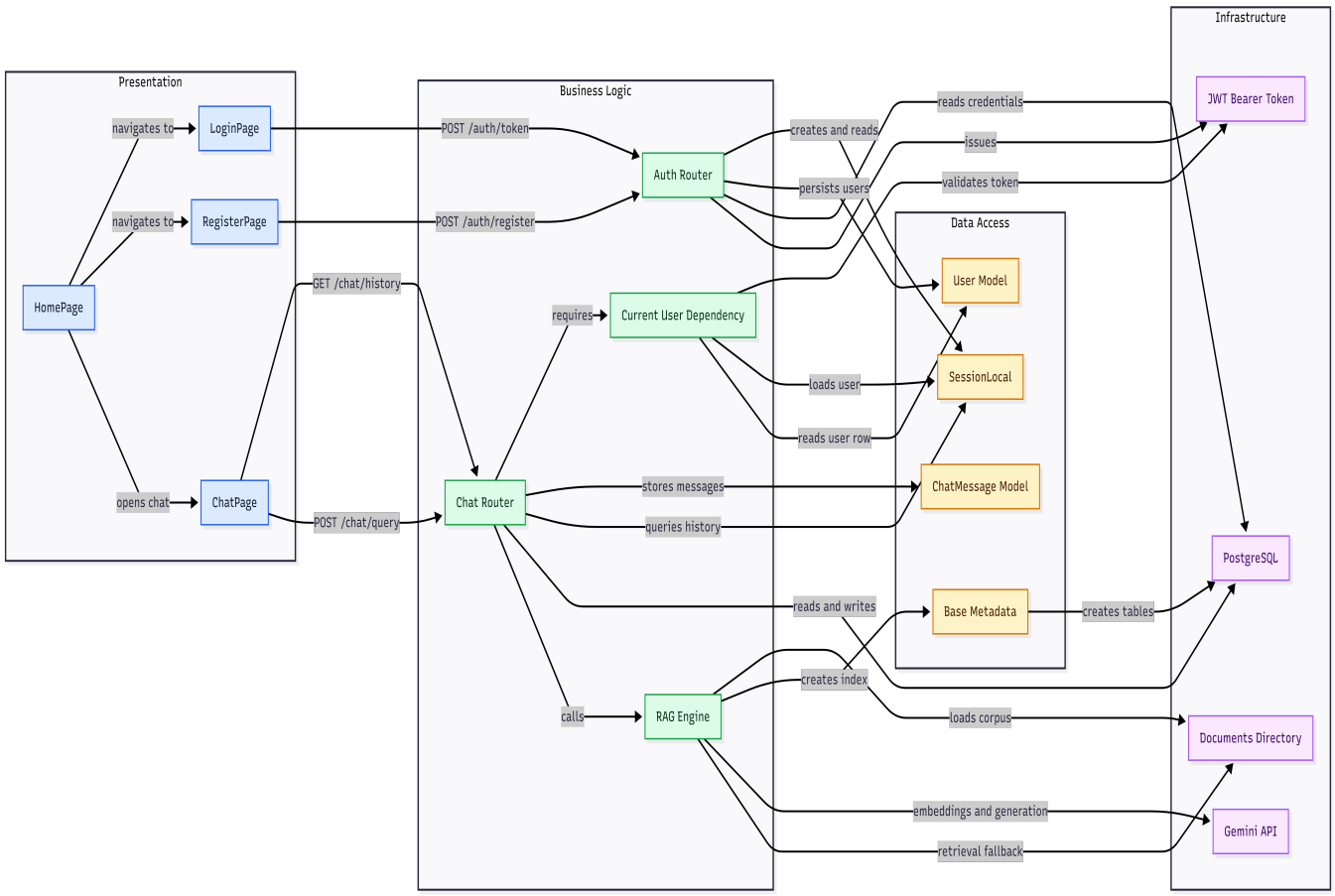


**Figura 1: Diagrama arquitetural e fluxo de comunicação de dados do sistema.**

podendo desativar modelos ou alterar os nomes dos *endpoints*. Para tornar a solução mais resiliente, foi implementado um mecanismo de seleção dinâmica de modelo: o *chatbot* lista os modelos disponíveis via API e seleciona automaticamente o primeiro que suporte o método `generateContent`, com *fallback* para um nome estático configurável por meio de variável de ambiente. Essa abordagem evitou quebras de serviço durante atualizações da API ao longo do desenvolvimento;

- **Controle de Custos e Limites de Contexto**. A API do Gemini é cobrada por *tokens* processados. Para controlar custos e respeitar os limites de contexto, foi implementado um limitador de caracteres no contexto enviado ao modelo (máximo de 9.000 caracteres por requisição). Quando o conjunto de *chunks* ultrapassa esse limite, os *chunks* são truncados em ordem de relevância decrescente. Adicionalmente, o histórico de conversa é limitado às últimas cinco mensagens, suficiente para manter coerência em diálogos encadeados sem comprometer a janela de contexto;
- **Fidelidade ao *Corpus* Corporativo**. Um risco inerente a modelos de linguagem de grande escala é a tendência de preencher lacunas de informação com conhecimento geral, mesmo quando instruídos a não o fazer. O controle foi realizado em duas camadas: i) instrução explícita no *prompt*, proibindo o uso de conhecimento externo e estabelecendo um comportamento de declínio transparente na ausência de informação; e ii) validação qualitativa das respostas por usuários especialistas do domínio, que identificavam respostas "fabricadas" pelo modelo. Essa combinação de instrução e validação humana mostrou-se mais eficaz do que abordagens puramente técnicas.

## 5 Resultados Observados

A avaliação do *chatbot* foi predominantemente qualitativa, realizada em parceria com usuários internos da NEOXS TECH durante um período de uso controlado. O processo corporativo de capacitação e consulta de materiais não contava com qualquer método automatizado, sendo completamente dependente da disponibilidade e da interação direta dos colaboradores com os profissionais mais experientes da organização. Para mostrar a eficácia e a confiabilidade da ferramenta diante desse cenário legado, o processo de validação envolveu especialistas seniores em vendas da empresa. Esses profissionais estressaram "estressaram" intencionalmente o modelo por meio de consultas complexas, ambíguas e situacionais, averiguando

minuciosamente a acurácia das respostas, a aderência às metodologias internas e o comportamento do sistema em cenários de alta exigência. Os principais resultados observados foram:

- A incorporação de HyDE resultou em uma melhora perceptível na relevância dos *chunks* recuperados para perguntas situacionais, exatamente o tipo mais comum no contexto de vendas B2B. Usuários relataram que as respostas passaram a citar estratégias e termos presentes nos materiais de treinamento de forma mais precisa e contextualizada;
- O mecanismo de *fallback*, que permite a limitação do sistema quando nenhum trecho relevante é encontrado, foi bem recebido pelos usuários, que o perceberam como um sinal de confiabilidade. A experiência confirma a literatura que aponta a transparência sobre incertezas como fator de confiança em sistemas conversacionais [2];
- O suporte ao histórico de conversa permitiu fluxos de diálogo mais naturais, em que o usuário podia refinar sua pergunta com base em respostas anteriores sem precisar recontextualizar todo o cenário a cada turno. Esse recurso foi especialmente valorizado em simulações de objeções de vendas, que frequentemente se desenvolvem em múltiplas etapas.
- O tempo médio de resposta ficou entre 3 e 7 segundos por consulta, considerado aceitável pelos usuários para um *chatbot* de suporte à decisão, e não para uma consulta instantânea. A latência é dominada pela chamada à API do Gemini, com impacto menor da busca vetorial FAISS, que opera em memória. A respeito do comportamento do sistema frente à escala, percebeu-se que, quanto maior o volume de dados inseridos no *corpus*, mais demorada e pesada se torna a requisição. Esse acréscimo no tempo de resposta e no peso computacional ocorre por causa da expansão do espaço de busca vetorial mapeado em memória e, prioritariamente, da sobrecarga no volume de texto recuperado, processado e transmitido no *payload* para a API geradora externa, evidenciando um limite de escala a ser considerado por outras equipes.

## 6 Conclusão

Neste artigo, relatou-se a experiência de desenvolvimento de um *chatbot B2B* baseado em RAG para a NEOXS TECH, descrevendo as decisões arquiteturais, os desafios enfrentados e as lições aprendidas. A principal contribuição prática é a combinação de HyDE e MMR para mitigar a lacuna semântica em *corpus* corporativo, mostrando que essa técnica é aplicável e eficaz em cenários reais de empresas de médio porte com documentação proprietária.

As lições aprendidas mais relevantes são: (i) o controle de *persona* por meio de instruções de sistema é eficaz, mas exige validação humana contínua; (ii) a estratégia de *chunking* deve ser calibrada empiricamente para cada *corpus*; (iii) mecanismos de resiliência, como *fallback* lexical e seleção dinâmica de modelo, são essenciais para sistemas em produção; e (iv) a transparência sobre as limitações do sistema aumenta a confiança dos usuários.

Como trabalho futuro, planeja-se a implementação de avaliação quantitativa com medidas de RAG, como RAGAS[15] [1], para mensurar *precision* e *recall* de forma sistemática, além da exploração de *reranking* baseado em *cross-encoders* para refinar a seleção dos *chunks* após a etapa de recuperação vetorial. Considerando o impacto direto da ferramenta na rotina dos usuários, projeta-se expandir a validação sob a perspectiva de Interação Humano-Computador (IHC). Pretende-se aplicar metodologias de IHC e conduzir estudos experimentais com usuários para mensurar o valor centrado no ser humano, avaliando a eficácia percebida, a usabilidade da interface conversacional e a calibração do nível de confiança do usuário no sistema quando os mecanismos de contingência (*fallback*) são acionados.

[15] https://www.ragas.io/

## Referências

[1] James J. Espinosa Anke L. Schockaert S. Es, S. 2024. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, Nikolaos Aletras and Orphee De Clercq (Eds.). Association for Computational Linguistics, St. Julians, Malta, 150–158. doi:10.18653/v1/2024.eacl-demo.16

[2] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise Zero-Shot Dense Retrieval without Relevance Labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 1762–1777. doi:10.18653/v1/2023.acl-long.99

[3] Perez E. Piktus A.. Petroni F. Karpukhin V. Goyal N. Küttler H. Lewis M. Yih W. Rocktäschel T. Riedel S. Kiela D. Lewis, P. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.