Relatório Técnico: Desenvolvimento do Assistente Virtual "VigIA" com Arquitetura RAG para Orientação em Situações de Incêndio

---

Lancelot Chagas Rodrigues / 554707
Ana Carolina Martins da Silva / 555762
Kauan Alves Batista / 555082

---

1. Descrição da Arquitetura Implementada (RAG para o VigIA)

Para o desenvolvimento do nosso assistente virtual VigIA, adotamos a arquitetura RAG (Retrieval-Augmented Generation). Optamos por esta abordagem devido a sua capacidade de combinar o poder de modelos de linguagem grandes (LLMs) com informações contextuais e atualizadas recuperadas de fontes externas. Isso permite que o VigIA forneca respostas mais precisas, relevantes e menos propensas a gerar informações incorretas, as chamadas "alucinações", do que um LLM utilizado isoladamente, especialmente em um dominio crítico como o de desastres naturais.

Nosso pipeline RAG para o VigIA foi estruturado em etapas sequenciais, executadas dinamicamente a cada interação do usuário:

- **a. Coleta de Consulta do Usuário e Identificação de Perfil:**
  A interação se inicia com o VigIA identificando o perfil do usuário (Vítima, Morador ou Familiar) atraves de um diálogo inicial. Essa identificação é crucial para adaptar o tom e o foco da resposta.
- **b. Recuperação Dinâmica de Informação (Retrieval):**
  Quando o usuário submete uma pergunta, o sistema primeiro realiza uma busca na web em tempo real. Utilizamos a biblioteca duckduckgo-search para consultar a internet por informações que possam responder à dúvida do usuário, focando em encontrar fontes potencialmente relevantes sobre incendios. Das URLs retornadas, selecionamos um número limitado das mais promissoras para processamento.
- **c. Extração e Processamento de Conteúdo Web:**
  O conteúdo textual principal de cada URL selecionada é extraído utilizando a biblioteca newspaper3k. Realizamos uma limpeza básica e garantimos que apenas conteúdo substancial seja considerado. Este texto extraído de cada fonte é então dividido em pedaços menores (chunks) utilizando o RecursiveCharacterTextSplitter da biblioteca LangChain, visando criar segmentos de texto semanticamente coerentes e de tamanho gerenciável.

- **d. Geração de Embeddings e Busca Vetorial em Memória:**
  Para cada chunk de texto obtido das fontes web, geramos um vetor de embedding utilizando o modelo models/embedding-001 da API Google GenAI, com task_type="RETRIEVAL_DOCUMENT". Estes embeddings são indexados em memória usando faiss-cpu (especificamente, IndexFlatIP com normalização L2). A pergunta do usuário também é convertida em um embedding (com task_type="RETRIEVAL_QUERY"), e realizamos uma busca por similaridade para identificar os N chunks mais relevantes.
- **e. Geração Aumentada da Resposta (Augmented Generation):**
  Os textos dos chunks recuperados formam o contexto. Este contexto, junto com a pergunta, o perfil do usuário e um prompt de sistema detalhado, é enviado ao modelo gemini-1.5-pro-latest da API Google GenAI. O prompt instrui o VigIA sobre sua persona, foco, uso estrito do contexto e nuances de tom.

2. Fontes de Informação Utilizadas

O VigIA utiliza um sistema de recuperação de informação dinâmico, buscando na web em tempo real a cada consulta. Isso garante que as informações base sejam potencialmente as mais atuais.

- **Mecanismo de Busca:** Empregamos a biblioteca duckduckgo-search, configurada para priorizar resultados em português do Brasil.
- **Extração de Conteúdo:** O conteúdo das URLs é processado pelo newspaper3k para focar no texto principal.
- **Limitações:** Reconhecemos a variabilidade na qualidade da informação online. O prompt do VigIA o instrui a ser cauteloso e direcionar para autoridades. Futuramente, poderíamos incorporar uma lista de dominios confiáveis prioritários.

3. Exemplos de Prompts e Respostas

O prompt enviado ao modelo Gemini é construído dinamicamente, incluindo: instrução de sistema (persona VigIA, foco em incêndios, uso estrito de contexto, etc.), perfil do usuário, contexto recuperado da web, e a pergunta do usuário. Detalhes específicos de tom para cada perfil também são fornecidos. Um exemplo completo do template do prompt está documentado no código-fonte (rag_assistant_tab.py).

A seguir, alguns exemplos de interações observadas durante os testes:

- **Exemplo 1 (Vítima - Queimaduras):**

    - *Perfil:* Vítima
    - *Pergunta:* Como cuidar de queimaduras e identificar o grau?
    - *Contexto Web (Exemplo Simplificado):* Documentos medicos indicam resfriamento com água para queimaduras de 1º grau e atendimento urgente para 2º e 3º grau.
    - *Resposta VigIA:* (Similar à resposta obtida nos testes: "Se você se queimou, avalie a situação: Queimadura de 1º grau: Pele vermelha... Lembre-se: estas informações são baseadas no contexto fornecido...")
- **Exemplo 2 (Familiar - Causa dos Incêndios):**
    - *Perfil:* Familiar
    - *Pergunta:* É verdade que o governo está escondendo a real causa dos incêndios?
    - *Contexto Web (Exemplo Simplificado):* Artigos de notícias citam declarações do presidente do Ibama sobre causas humanas e investigações.
    - *Resposta VigIA:* (Similar à resposta obtida nos testes: "Querido familiar, entendo sua preocupação. De acordo com as informações que tenho acesso, Rodrigo Agostinho, presidente do Ibama, afirmou...")
- **Exemplo 3 (Morador - Prevenção Doméstica):**
    - *Perfil:* Morador
    - *Pergunta:* Como proteger minha casa de incêndios?
    - *Contexto Web (Exemplo Simplificado):* Guias de defesa civil sugerem limpeza de terreno e materiais resistentes; manuais específicos são recomendados.
    - *Resposta VigIA:* (Similar à resposta obtida, onde ele admite contexto limitado e dá dicas gerais rotuladas como tal).

4. Reflexão Crítica sobre o Sistema

- **Pontos Fortes:**
  Nossa equipe observou que a arquitetura RAG com busca web dinamica permitiu ao VigIA fornecer respostas contextualmente relevantes e atuais. A adaptação do tom com base no perfil do usuário foi bem-sucedida, e o sistema demonstrou capacidade de admitir limitações quando o contexto recuperado

era insuficiente, evitando a fabricação de respostas. A flexibilidade da busca web é um ponto positivo para informações recentes.

- **Desafios e Limitações:**
  A eficácia do VigIA depende crucialmente da qualidade dos resultados da busca DuckDuckGo e da extração do Newspaper3k. Para algumas perguntas específicas, o contexto recuperado nem sempre foi ótimo, resultando em respostas mais genericas. A latência inerente à busca web e processamento em tempo real é uma consideração. Observamos tambem que, mesmo instruído a usar apenas o contexto, o LLM pode, em raras ocasiões, inferir cenários um pouco além do estritamente fornecido se o contexto for vago. A veracidade das fontes da web é um desafio contínuo, mitigado parcialmente pelas instruções de cautela no prompt do VigIA.

- **Melhorias Futuras:**
  Sugerimos para futuras iterações a implementação de uma lista de "domínios confiáveis" para a busca web, refinamento das estratégias de chunking e re-ranking dos resultados de busca. A criação de uma pequena base de conhecimento estatica local para informações críticas de segurança, consultada antes da busca web, também seria um acréscimo valioso.

- **Impactos Sociais e Éticos:**
  Tivemos a responsabilidade de lidar com informações críticas em mente. O risco de informação incorreta, mesmo com RAG, existe. Por isso, o VigIA foi projetado para ser cauteloso, admitir quando não sabe e sempre incentivar o contato com serviços de emergência. A linguagem empática foi considerada essencial. O uso de IA para este fim é promissor, mas exige monitoramento contínuo.