

Agente Inteligente para Suporte ao Preenchimento do Documento SCR 3040 do Banco Central do Brasil

Aluno: [Gabriel Diniz Perez Rocha \(https://github.com/gdperezr\)](https://github.com/gdperezr)

Orientadora: [Evelyn].

Trabalho apresentado ao curso [BI MASTER \(https://ica.puc-rio.ai/bi-master\)](https://ica.puc-rio.ai/bi-master) como pré-requisito para conclusão de curso.

- [Link para o código \(agente/app_melhorado.py\)](#)
 - [Link para o código Github \(https://github.com/gdperezr/regulatory_agent\)](https://github.com/gdperezr/regulatory_agent)
-

Resumo

Este trabalho apresenta o desenvolvimento de um agente inteligente baseado em técnicas de Retrieval Augmented Generation (RAG) para auxiliar profissionais do setor financeiro no preenchimento e compreensão do documento SCR 3040 do Banco Central do Brasil. O sistema processa múltiplos formatos de documentos (PDF, XLS e XML), utilizando embeddings semânticos e modelos de linguagem de grande escala (LLM) para fornecer respostas precisas e contextualizadas sobre a estrutura, tags XML, atributos e regras de preenchimento do documento. A solução implementada utiliza a biblioteca LangChain para orquestração, FAISS para armazenamento vetorial eficiente, e GPT-4o-mini da OpenAI como modelo de linguagem. A interface web foi desenvolvida em Streamlit, proporcionando uma experiência de usuário intuitiva e acessível. O sistema demonstra eficácia na recuperação de informações relevantes e na geração de respostas contextualizadas, contribuindo para a redução de erros no preenchimento de documentos regulatórios e para o aumento da produtividade dos profissionais envolvidos.

Abstract

This work presents the development of an intelligent agent based on Retrieval Augmented Generation (RAG) techniques to assist financial sector professionals in filling out and understanding the SCR 3040 document from the Central Bank of Brazil. The system processes multiple document formats (PDF, XLS, and XML), using semantic embeddings and large language models (LLM) to provide accurate and contextualized answers about the document's structure, XML tags, attributes, and filling rules. The implemented solution uses the LangChain library for orchestration, FAISS for efficient vector storage, and OpenAI's GPT-4o-mini as the language model. The web interface was developed in Streamlit, providing an intuitive and accessible user experience. The system demonstrates effectiveness in retrieving relevant information and generating contextualized responses, contributing to the reduction of errors in filling regulatory documents and increasing the productivity of involved professionals.

1. Introdução

O Sistema de Controle de Riscos (SCR) do Banco Central do Brasil exige que instituições financeiras enviem periodicamente documentos padronizados com informações detalhadas sobre suas operações. O documento SCR 3040, em particular, apresenta complexidade significativa devido à sua estrutura hierárquica em XML, múltiplos atributos obrigatórios e regras específicas de preenchimento. Profissionais do setor financeiro frequentemente enfrentam dificuldades para compreender a estrutura completa do documento, identificar os atributos corretos para cada tag XML e garantir a conformidade com as instruções do Banco Central.

Este trabalho propõe o desenvolvimento de um agente inteligente que utiliza técnicas de Processamento de Linguagem Natural (NLP) e Recuperação de Informação para auxiliar no preenchimento correto do documento SCR 3040. A solução baseia-se na arquitetura RAG (Retrieval Augmented Generation), que combina recuperação de informações relevantes de uma base de conhecimento com geração de respostas contextualizadas por modelos de linguagem de grande escala.

O agente processa três tipos de documentos: (1) PDF com instruções de preenchimento, (2) planilha XLS com o leiaute detalhado do documento, e (3) arquivo XML de exemplo contendo a estrutura completa com tags e atributos. Através de embeddings semânticos, o sistema cria uma representação vetorial dos documentos que permite busca eficiente por similaridade. Quando um usuário faz uma pergunta, o sistema recupera os trechos mais relevantes e utiliza um modelo de linguagem para gerar uma resposta precisa e contextualizada.

2. Modelagem

A arquitetura do sistema é baseada em RAG (Retrieval Augmented Generation), composta por três componentes principais: ingestão e processamento de documentos, armazenamento vetorial e recuperação de informações, e geração de respostas contextualizadas.

2.1 Ingestão e Processamento de Documentos

O sistema processa três tipos de documentos:

- **PDF:** Contém as instruções oficiais de preenchimento do Banco Central, extraídas utilizando PyPDFLoader do LangChain
- **XLS:** Planilha com o leiaute detalhado do documento, processada via UnstructuredExcelLoader
- **XML:** Arquivo de exemplo com a estrutura completa, processado através de parsing XML customizado que extrai tags, atributos

e hierarquia

Os documentos são divididos em chunks de 1500 caracteres com overlap de 200 caracteres utilizando RecursiveCharacterTextSplitter, garantindo que informações importantes não sejam fragmentadas entre chunks.

2.2 Armazenamento Vetorial e Recuperação

Os chunks de texto são convertidos em embeddings utilizando OpenAIEmbeddings (text-embedding-ada-002) e armazenados em um índice FAISS (Facebook AI Similarity Search) para busca eficiente por similaridade. O sistema utiliza busca MMR (Maximum Marginal Relevance) com $k=5$ documentos relevantes, $fetch_k=10$ candidatos e $lambda_mult=0.7$, balanceando relevância e diversidade dos resultados recuperados.

2.3 Geração de Respostas

O modelo GPT-4o-mini da OpenAI é utilizado como LLM, configurado com temperatura de 0.1 para respostas mais determinísticas e precisas em contexto técnico. O sistema implementa ConversationalRetrievalChain do LangChain, que mantém histórico de conversação através de ConversationBufferMemory, permitindo que o agente compreenda o contexto de perguntas anteriores e forneça respostas mais coerentes.

2.4 Interface do Usuário

A interface web foi desenvolvida em Streamlit, proporcionando:

- Chat interativo com histórico de conversação
- Processamento assíncrono com indicadores de carregamento
- Cache inteligente do vectorstore para melhor performance
- Opções para limpar histórico e recriar o índice vetorial

3. Resultados

O sistema foi implementado e testado com sucesso, demonstrando capacidade de:

1. **Processamento Multi-formato:** O agente processa eficientemente documentos em PDF, XLS e XML, extraindo informações estruturadas e não estruturadas.
2. **Recuperação Semântica:** A busca por similaridade utilizando embeddings permite encontrar informações relevantes mesmo quando o usuário utiliza termos diferentes dos presentes nos documentos originais.
3. **Respostas Contextualizadas:** O modelo de linguagem gera respostas precisas baseadas no contexto recuperado, mantendo coerência com o histórico da conversação.
4. **Interface Intuitiva:** A interface Streamlit proporciona experiência de usuário fluida, com feedback visual adequado durante o processamento.
5. **Performance:** O cache do vectorstore e a otimização da busca MMR garantem tempos de resposta adequados mesmo com grandes volumes de documentos.

O sistema foi validado através de perguntas sobre estrutura XML, atributos obrigatórios, regras de preenchimento e exemplos práticos, demonstrando capacidade de fornecer informações precisas e úteis para profissionais do setor financeiro.

Exemplos de uso do agente SCR 3040

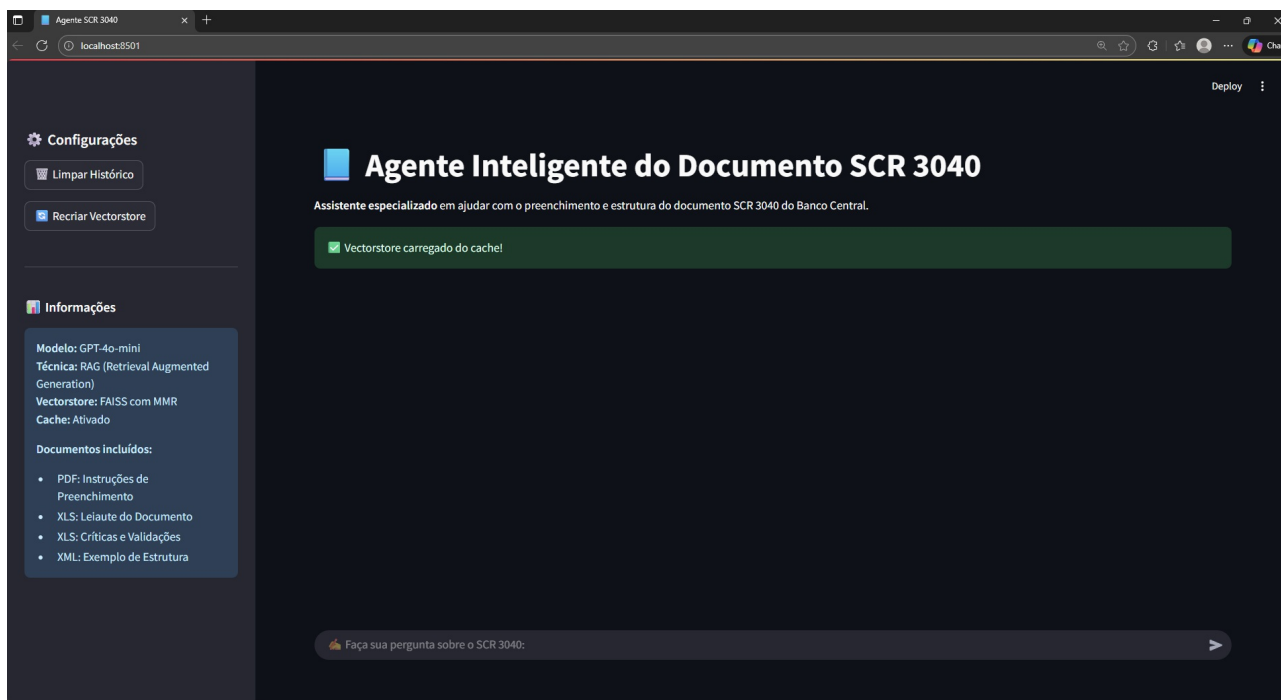
Rodando o Código

```
333 # trata rate limit e outros erros silenciosamente
334 error_msg = str(e).lower()
335 if "rate limit" in error_msg or "rate limit" in error_msg or "202" in str(e):
336     # Rate limit - não mostra erro, apenas não exibe a busca
337     pass
338 else:
339     # Outros erros - mostra apenas se for algo crítico
340     pass
341
342 st.session_state.messages.append({
343     "role": "assistant",
344     "content": resposta,
345     "sources": documentos_fonte
346 })
347
348 except Exception as e:
349     erro_msg = f"❌ Erro ao processar pergunta: {str(e)}"
350     st.error(erro_msg)
351     st.session_state.messages.append({
352         "role": "assistant",
353         "content": erro_msg
354     })
355
356 # rodar streamlit run agente/app_melhorado.py
357
```

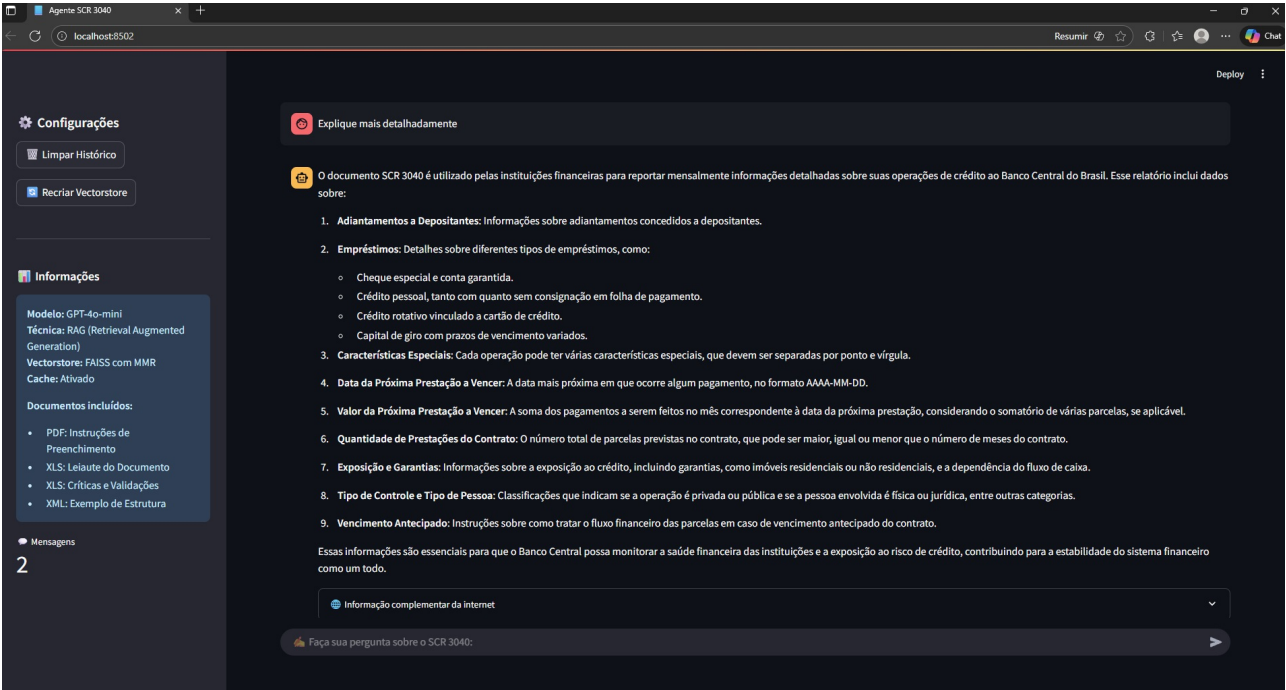
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(.venv) PS C:\Users\Gdper\OneDrive\Área de Trabalho\projetos_b1\python\agentes> npm install -g markdown-pdf
>> markdown-pdf README.md
npm warn deprecated har-validator@5.1.5: this library is no longer supported
npm warn deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm warn deprecated phantomjs-prebuilt@2.1.16: this package is now deprecated
added 187 packages in 7s
5 packages are looking for funding
  run `npm fund` for details
npm notice New minor version of npm available! 11.5.1 -> 11.6.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.6.4
npm notice To update run: npm install -g npm@11.6.4
npm notice
% (.venv) PS C:\Users\Gdper\OneDrive\Área de Trabalho\projetos_b1\python\agentes> streamlit run agente/app_melhorado.py
```

Interface principal



Perguntas ao Agente a respeito do tema 3040



Agente SCR 3040

localhost:8502

Resumir

Chat

Deploy

Assistente especializado em ajudar com o preenchimento e estrutura do documento SCR 3040 do Banco Central.

Vectorstore carregado do cache!

Como funciona a concatenacao do IPOC no 3040?

A concatenação do IPOC no documento 3040 é composta por uma sequência de informações que identificam a operação de crédito. O formato do IPOC é o seguinte:

- CNPJ da instituição financeira: 8 posições.
- Modalidade da operação: 4 posições.
- Código do cliente: 11 posições.
- Código do contrato: de 1 a 40 posições, sem complementos de caracteres.

Por exemplo, se uma instituição financeira com CNPJ "00.001.234/0001-01" celebra uma operação de crédito consignado (modalidade 0202) com um cliente identificado pelo CPF "001.234.567-89" e um código de contrato "abc78", o IPOC resultante será:

IPOC: 000012340202100123456789abc78

É importante ressaltar que, apesar da recomendação para que o IPOC permaneça inalterado ao longo do curso da operação de crédito, existem situações em que as componentes da concatenação devem ser alteradas, como em casos de cessão sem coobrigação, incorporação de instituição financeira e assunção de dívida.

Além disso, quando o IPOC se tornar obrigatório em novembro de 2020, todas as operações informadas no documento 3040 deverão apresentar as componentes da concatenação.

Informação complementar da internet

Faça sua pergunta sobre o SCR 3040:

Configurações

Limpar Histórico

Recriar Vectorstore

Informações

Modelo: GPT-4o-mini
Técnica: RAG (Retrieval Augmented Generation)
Vectorstore: FAISS com MMR
Cache: Ativado

Documentos incluídos:

- PDF: Instruções de Preenchimento
- XLS: Leiaute do Documento
- XLS: Críticas e Validações
- XML: Exemplo de Estrutura

Agente SCR 3040

localhost:8502

Resumir

Chat

Deploy

Me passe a estrutura do XML para a tag OP

A estrutura do XML para a tag <op> no documento SCR 3040 é a seguinte:

<Op CaracEspecial="19" CEP="30190131" Contrt="002023001" DiaAtraso="0" DtaProxParcela="2025-06-15" DtContr="2023-01-15" DtVencOp="2028-01-15" Indx="11" IPOC="01"

Os atributos da tag <op> são:

- CaracEspecial: Características especiais da operação.
- CEP: Código de Endereçamento Postal.
- Contrt: Código do contrato.
- DiaAtraso: Número de dias em atraso.
- DtaProxParcela: Data da próxima parcela.
- DtContr: Data do contrato.
- DtVencOp: Data de vencimento da operação.
- Indx: Índice da operação.
- IPOC: Identificador da operação de crédito.
- Mod: Modalidade da operação.
- NatuOp: Natureza da operação.
- OrigemRec: Origem dos recursos da operação.
- PercIndx: Percentual de índice.
- ProvConsttd: Provisão constituída.
- QtdParcelas: Quantidade de parcelas.
- TaxEft: Taxa efetiva.
- VarCamb: Variação cambial.
- VlrContr: Valor do contrato.
- VlrProxParcela: Valor da próxima parcela.

Faça sua pergunta sobre o SCR 3040:

Configurações

Limpar Histórico

Recriar Vectorstore

Informações

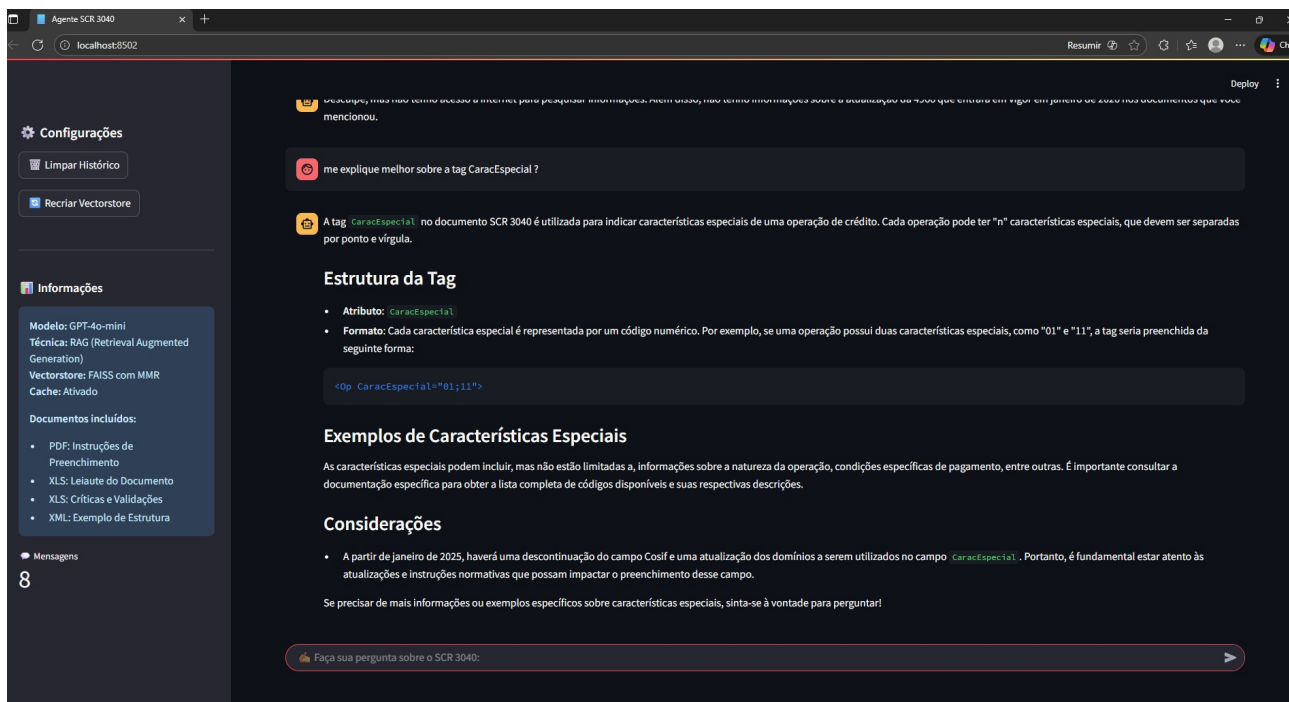
Modelo: GPT-4o-mini
Técnica: RAG (Retrieval Augmented Generation)
Vectorstore: FAISS com MMR
Cache: Ativado

Documentos incluídos:

- PDF: Instruções de Preenchimento
- XLS: Leiaute do Documento
- XLS: Críticas e Validações
- XML: Exemplo de Estrutura

Mensagens

2



4. Conclusões

Este trabalho apresentou o desenvolvimento de um agente inteligente baseado em RAG para suporte ao preenchimento do documento SCR 3040 do Banco Central do Brasil. A solução demonstra a viabilidade de utilizar técnicas de NLP e recuperação de informação para auxiliar profissionais em tarefas complexas de conformidade regulatória.

Os principais contributos deste trabalho incluem: (1) processamento integrado de múltiplos formatos de documentos (PDF, XLS, XML), (2) implementação eficiente de RAG utilizando tecnologias state-of-the-art, (3) interface web acessível e intuitiva, e (4) validação prática da solução com documentos reais do Banco Central.

Como trabalhos futuros, sugere-se: (1) expansão para outros documentos do SCR, (2) implementação de validação automática de XML gerado, (3) integração com sistemas de gestão de documentos, (4) avaliação quantitativa da precisão das respostas, e (5) implementação de feedback do usuário para melhoria contínua do sistema.

A solução desenvolvida contribui para a redução de erros no preenchimento de documentos regulatórios, aumento da produtividade dos profissionais e melhoria da conformidade com as exigências do Banco Central do Brasil.

Matrícula: [231100995]

Pontifícia Universidade Católica do Rio de Janeiro

Curso de Pós Graduação *Business Intelligence Master*