Análise e Proposta de Implementação para as Fases 4 e 5

Fase 4: Interface de Revisão Humana (Human-in-the-Loop)

Objetivo: Criar um ciclo de feedback onde especialistas fiscais possam validar, corrigir e auditar as classificações geradas pelo sistema, garantindo a máxima qualidade e conformidade.

Estratégia Conceitual

A base desta fase é conectar a saída do seu HybridRouter (o JSON detalhado de classificação) a uma interface amigável. O fluxo será:

1. O sistema classifica um lote de produtos.
2. Os resultados são armazenados em um banco de dados com um status inicial (ex: PENDENTE_REVISAO).
3. Uma API REST expõe esses resultados para uma aplicação web.
4. Especialistas acessam a interface, revisam as sugestões, aprovam ou corrigem.
5. As correções são salvas no banco de dados, fechando o ciclo de feedback.

Plano de Implementação Detalhado

1. Implementar API REST para Revisão de Classificações

- **Tecnologia Sugerida: FastAPI** em Python. É moderno, de alta performance, gera documentação interativa (Swagger UI) automaticamente e se integra perfeitamente com o seu ecossistema Python.
- **Estrutura de Dados (Banco de Dados):**
  - Recomendo estender sua tabela produto no PostgreSQL ou criar uma nova tabela classificacoes para armazenar os resultados e o status da revisão.
  - **Novas Colunas Necessárias:**
    - status_revisao (TEXT): PENDENTE_REVISAO, APROVADO, CORRIGIDO.
    - ncm_sugerido (VARCHAR): O NCM que o agente classificou.
    - cest_sugerido (VARCHAR): O CEST que o agente classificou.
    - confianca_sugerida (FLOAT): A confiança consolidada.
    - ncm_corrigido (VARCHAR, nullable): O NCM inserido pelo especialista.
    - cest_corrigido (VARCHAR, nullable): O CEST inserido pelo especialista.
    - revisado_por (VARCHAR, nullable): ID do especialista.
    - data_revisao (TIMESTAMP, nullable).
    - justificativa_correcao (TEXT, nullable).
    - dados_trace_json (JSONB): Para armazenar o objeto completo de traces e auditoria.
- **Endpoints da API (Exemplos):**
  - GET /api/v1/classificacoes:

        - **Função:** Lista as classificações para revisão.
        - **Filtros:** ?status=PENDENTE_REVISAO, ?confianca_min=0.7, ?page=1&limit=50.
        - **Retorno:** Uma lista de produtos com seus dados principais e a classificação sugerida.
    - GET /api/v1/classificacoes/{produto_id}:
        - **Função:** Retorna todos os detalhes de uma classificação específica, incluindo os traces completos dos agentes.
        - **Uso:** Essencial para a tela de detalhes, onde o especialista precisa entender *por que* o sistema tomou aquela decisão.
    - PUT /api/v1/classificacoes/{produto_id}/revisar:
        - **Função:** Endpoint principal para o especialista submeter a revisão.
        - **Corpo da Requisição (Body):**

```
{
  "acao": "APROVAR" // ou "CORRIGIR"
  "ncm_corrigido": "22021000", // Opcional, apenas se acao == "CORRIGIR"
  "cest_corrigido": "03.002.00", // Opcional
  "justificativa_correcao": "Produto é refrigerante, mas a embalagem é PET, não lata." // Opcional
}
```

        - **Lógica:** O backend atualiza o registro no banco de dados com os novos dados e muda o status_revisao.

2. Interface Web para Especialistas

- **Tecnologia Sugerida: React, Vue.js ou Svelte**. São frameworks modernos que permitem criar interfaces reativas e componentizadas.
- **Telas Principais:**
    - **Dashboard de Monitoramento:**
        - **Gráficos:**
            - Classificações por status (pizza ou barra).
            - Acurácia ao longo do tempo (% de classificações aprovadas sem correção).
            - Distribuição de Scores de Confiança (histograma).
            - Tempo médio de revisão.
        - **Dados:** Consumidos do endpoint GET /api/v1/dashboard/stats (que você também criaria).
    - **Fila de Revisão:**

        - Uma tabela ou lista de produtos com status=PENDENTE_REVISAO.
        - **Colunas:** Descrição do Produto, NCM Sugerido, Confiança, Data da Classificação.
        - **Funcionalidades:** Paginação, busca e filtros.
    - **Tela de Detalhe da Revisão (A mais importante):**
        - **Layout:** Dividido em seções claras.
        - **Seção 1: Dados do Produto:** Descrição original, código, etc.
        - **Seção 2: Classificação do Sistema:**
            - **NCM Sugerido:** 2202.10.00 (com a descrição oficial ao lado).
            - **Confiança:** 0.85.
            - **Justificativa do Agente:** Exibe a justificativa_final do ReconcilerAgent.
            - **Rastreabilidade (Traces):** Uma área colapsável/aba que mostra os traces de cada agente (ExpansionAgent, NCMAgent, etc.). Isso é **crucial** para a confiança do especialista no sistema.
        - **Seção 3: Ação do Especialista:**
            - Campos de input para NCM Corrigido e CEST Corrigido, pré-preenchidos com a sugestão do sistema.
            - Botões: **"Aprovar Sugestão"** e **"Salvar Correção"**.

Fase 5: Aprendizagem Contínua

Objetivo: Utilizar o feedback valioso dos especialistas para retroalimentar e melhorar a precisão do sistema de forma automática e periódica.

Estratégia Conceitual

As correções humanas são a fonte da verdade ("golden source"). Em vez de um complexo e caro retreinamento do modelo de linguagem base (llama3), a abordagem mais eficaz para sistemas RAG como o seu é **aprimorar a etapa de recuperação de informação (Retrieval)**.

1. As classificações validadas (APROVADO ou CORRIGIDO) formam um "Golden Set".
2. Este Golden Set é usado para criar um novo índice vetorial, menor e de altíssima qualidade.
3. Na hora de classificar um novo produto, o sistema busca por exemplos similares tanto no índice original (geral) quanto no índice do Golden Set (especializado).
4. Os exemplos do Golden Set, por serem validados, recebem um peso maior no contexto enviado ao LLM, guiando-o para uma decisão mais precisa.

Plano de Implementação Detalhado

1. Salvar Correções Humanas como "Golden Set"

- Isso já foi resolvido pelo modelo de dados da Fase 4. O seu "Golden Set" é simplesmente uma consulta no banco de dados:
  SELECT descricao_produto, ncm_corrigido, cest_corrigido
  FROM classificacoes
  WHERE status_revisao = 'CORRIGIDO' OR status_revisao = 'APROVADO';

2. Retreinamento Periódico (Augmented Retrieval)

- **Processo:**
  1. **Criar um novo script:** scripts/update_golden_index.py.
  2. **Agendamento:** Este script pode ser executado periodicamente (ex: toda noite) via cron ou um agendador similar.
  3. **Lógica do Script:**
     - Conecta ao banco e executa a query do "Golden Set".
     - Para cada produto no Golden Set, gera um embedding da descricao_produto usando o mesmo modelo sentence-transformers/all-MiniLM-L6-v2.
     - Salva esses embeddings em um **novo índice FAISS separado** (ex: data/knowledge_base/golden_set_index.faiss) e seus metadados em um golden_metadata.db.
- **Modificação no HybridRouter:**
  - No método _get_semantic_context, em vez de buscar apenas no vector_store principal, ele agora fará duas buscas:
    1. Busca no índice principal (faiss_index.faiss) por k=3 exemplos.
    2. Busca no índice do Golden Set (golden_set_index.faiss) por k=2 exemplos.
  - Os 5 exemplos (3 gerais + 2 validados) são combinados. Você pode até prefixar os exemplos do Golden Set com um marcador, como "[Exemplo Validado]: ..." no prompt para dar mais peso a eles na decisão do LLM.

3. Análise de Drift na Qualidade

- **Conceito:** Drift ocorre quando as características dos novos produtos mudam com o tempo, e a performance do modelo começa a degradar.
- **Implementação:**
  - **Monitoramento no Dashboard:** A forma mais simples e eficaz é monitorar as métricas da Fase 4 ao longo do tempo.
    - Crie gráficos de linha no dashboard para acompanhar a **"Acurácia (%)"** (classificações aprovadas / total revisado) semana a semana.
    - Monitore o **"Score de Confiança Médio"** das novas classificações.

- **Detecção:** Se você observar uma tendência de queda na acurácia ou na confiança média por várias semanas consecutivas, isso é um forte indicador de drift.
- **Ação:** A análise de drift aciona uma investigação. Pode ser necessário:
  - Analisar os produtos que estão sendo classificados incorretamente para identificar novos padrões.
  - Atualizar a base de conhecimento estruturada (ncm_mapping.json) com novas regras ou exemplos.
  - O próprio sistema de aprendizagem contínua já ajuda a mitigar o drift, pois novos padrões corrigidos pelos especialistas são incorporados ao Golden Set.

Resumo da Proposta

Fase	Componente	Tecnologia/Estratégia Proposta
Fase 4	API REST	**FastAPI** com endpoints para GET e PUT de classificações.
	Banco de Dados	**PostgreSQL** com colunas adicionais para status e correções.
	Interface Web	**React/Vue** com telas para Dashboard, Fila e Detalhe da Revisão.
	Monitoramento	**Chart.js/Recharts** no frontend para visualização de métricas.
Fase 5	Golden Set	**Query no PostgreSQL** buscando classificações revisadas.
	Retreinamento	**Augmented Retrieval**: um segundo índice FAISS para o Golden Set.
	Análise de Drift	**Monitoramento contínuo** da acurácia e confiança no Dashboard.

Este plano constrói sobre a excelente base que você já tem, criando um ciclo de

melhoria contínua que une a inteligência artificial dos seus agentes com a expertise indispensável dos especialistas humanos.