Documentação: Sistema de Filtragem de Dados de Notas Fiscais Eletrônicas

Péricles dos Santos Leite
Projeto da construção das Tabelas de Recursos e de Usos de Sergipe
Universidade Federal de Sergipe (UFS)

1 Visão Geral

Esta documentação descreve o funcionamento e a estruturação do sistema de filtragem de dados desenvolvido para o projeto de pesquisa da construção das Tabelas de Recursos e de Usos (TRU) do Estado de Sergipe. A ferramenta foi projetada para receber e processar volumes expressivos de informações contidas em conjuntos de dados do tipo Notas Fiscais Eletrônicas (NFe), atuando como um motor de busca que extrai e organiza dados brutos em planilhas e áreas de síntese que melhor auxiliam na visualização da informação buscada.

O sistema opera como um filtro multidimensional que valida e normaliza códigos fundamentais para a análise econômica, como o CNAE (Classificação Nacional de Atividades Econômicas), o NCM (Nomenclatura Comum do Mercosul) e a classificação BEC (Grandes Categorias Econômicas). Além da tipificação técnica dos produtos e setores, a aplicação gerencia informações geográficas cruciais, permitindo o rastreamento de fluxos comerciais entre municípios sergipanos e diferentes Unidades Federativas de emissão ou destino.

Para garantir a integridade dos dados, o software executa rotinas automáticas de preenchimento de lacunas e padronização de caracteres, assegurando que as informações fiscais sejam convertidas em valores estatísticos confiáveis.

2 Especificação dos Filtros

O processamento de dados inicia-se obrigatoriamente pela definição do parâmetro "ANO", que instrui o sistema a carregar a base de dados correspondente ao período de emissão das Notas Fiscais Eletrônicas (NFe). A arquitetura de filtragem é regida predominantemente pela lógica AND, exigindo que todos os critérios definidos nos campos de preenchimento único sejam atendidos simultaneamente. Em contrapartida, campos que admitem seleção múltipla — como "Período"e "Produtos SCN - 75"— operam sob a lógica OR, permitindo a agregação de diversos itens em uma única consulta. Caso um campo não seja especificado pelo usuário, o sistema permite a saída de todos os resultados correspondentes sem restrições adicionais.

A interface do sistema é dividida em duas principais partes, de acordo com as seções de filtragem que a compõem: **Principal** e **Filtros Avançados**. Abaixo, são listados todos os campos de filtragem e suas respectivas especificações:

Parte Principal

Ano

Define o ano de emissão da NFe para o carregamento da base de dados correspondente. Possui o caráter de preenchimento obrigatório.

Código NCM

Busca textual por prefixo (1 a 7 dígitos). Permite filtrar desde grandes grupos de mercadorias até itens específicos.

UF (Dest/Orig)

Seleção da Unidade da Federação envolvida na operação comercial.

Município (Dest/Orig)

Seleção do município específico de destino ou de emissão da nota.

BEC

Classificação por Grandes Categorias Econômicas aplicada aos produtos.

CNAE (Emi/Dest)

Busca por atividade econômica (1 a 8 dígitos), seguindo a lógica de busca por início de código.

Período

Filtro de múltipla seleção para meses isolados ou agrupamentos temporais específicos, permitindo filtrar os dados por trimestres ou quaisquer outras formas de agrupamento de meses.

Filtros Avançados

Produto SCN - 75:

Caixa de múltipla seleção de produtos baseada na classificação SCN;

Setor SCN - 38:

A seleção deste campo (Emissão ou Destino) desbloqueia uma lista multi selecionável dos setores de interesse.

3 Carregamento de Informações

Após o preenchimento dos campos de filtragem com o que se deseja procurar, o usuário deve acionar o botão "Buscar", localizado à direita na última linha da área de seleção, destacado na cor verde, para assim dar início ao processamento dos dados. Com o acionamento, uma barra de progresso é exibida imediatamente abaixo dos campos, permitindo que o usuário acompanhe a execução das instruções requisitadas e estime o tempo que levará para concluir o processo por inteiro, dado que naturalmente arquivos maiores consomem um tempo maior de execução. O fluxo de processamento é segmentado em 8 etapas, cada uma responsável por tarefas específicas no código, explicadas a seguir junto do trecho de código responsável por elas:

Etapa 1: Leitura

É realizada a leitura dos csv's a partir do método `read_csv()` da biblioteca Pandas, além da inicialização da variável que armazena uma cópia do dataframe referente a Nota utilizada e em cujo todo processamento será realizado.

```
notas = pd.read_csv(NFE,delimiter=';',encoding=ENCODING,dtype={'NCM':str})
ncm_merg = pd.read_csv(NCM_MERGED,dtype={'NCM':str})
scn_cnae = pd.read_csv(CNAE_MERGED, dtype={f'CNAE_{TIPO}':str})

#Inicializa variavel como copia do csv notas
df = notas.copy()
```

Etapa 2: Normalização

Aplicação das funções `normalizar_ncm()` e `normalizar_cnae()` do conjunto de funções do módulo `app_components` para padronização dos códigos econômicos, fazendo a remoção de espaços vazios ao inicio e ao final das strings e de pontos flutuantes, e passando o tamanho de todos os códigos para o padrão econômico, 8 dígitos para NCM e 7 dígitos para o CNAE.

```
#Padronizacao da coluna NCM
df['NCM'] = ui.normalizar_ncm(df['NCM'])
ncm_merg['NCM'] = ui.normalizar_ncm(ncm_merg['NCM'])
ncm_merg['NCM'] = ncm_merg['NCM'].str.strip()

#Padronizacao da coluna CNAE
df[f'CNAE_{TIPO}'] = ui.normalizar_cnae(df[f'CNAE_{TIPO}'])
scn_cnae[f'CNAE_{TIPO}'] = ui.normalizar_cnae(scn_cnae[f'CNAE_{TIPO}'])
scn_cnae['SETORES SCN'] = scn_cnae['SETORES SCN'].str.strip()
```

Etapa 3: Merges

Mescla do conjunto de dados das Notas com informações adicionais sobre os códigos CNAE e NCM a partir do método `merge()` da biblioteca Pandas, priorizando a conservação de todos os valores do dataframe da Nota com o uso do formato "left". Os arquivos *ncm_merg* e *scn_cnae* possuem informações extras que complementam os dados das Notas, como colunas de descrição, novos códigos como BEC que se associam com o NCM e a quais setores a CNAE está relacionada no padrão do Sistema de Contas Nacionais (SCN).

```
#Merges
df = pd.merge(df, ncm_merg, on='NCM', how='left')
df = pd.merge(df, scn_cnae, on=f'CNAE_{TIPO}', how='left')
```

Etapa 4: Ajuste NCM incompleto

Substitui NCM's com quantidade de dígitos inferior a 7 por "NAN", dada a impossibilidade de associa-lo a um produto especifico. Esta etapa foi adicionada por robustez a bugs, mesmo os códigos tendo sido normalizados na etapa anterior. Apesar disso, a complexidade de tempo nesta etapa é muito baixo, não fazendo diferença no tempo total de processamento.

```
df.loc[df['NCM'].str.len() < 7, 'NCM'] = ('NA')
```

Etapa 5: Filtros Avançados

Aplica o filtro sobre os campos da seção de Filtros Avançados, controlada pela mudança da variável de estado *session_state.switch* para 0, que acontece ao apertar o botão de Filtros Avançados. Nesta parte do sistema, encontram-se, além das novas opções de campos "Produto SCN - 75"e "Setor SCN - 38", campos já existentes no primeiro layout Principal, como Ano, UF e Município.

```
if st.session_state.switch == 0:
#Filtragem dos produtos por SCN
if len(produto) > 0:
    df['PRODUTOS SCN'] = df['PRODUTOS SCN'].str.strip()
    df = df[df['PRODUTOS SCN'].isin(produto)]
if len(scnae) > 0:
    df['SETORES SCN'] = df['SETORES SCN'].str.strip()
    df = df[df['SETORES SCN'].isin(scnae)]
```

Etapa 6: Filtros Gerais

Nessa etapa, são aplicados os filtros sobre todos os campos em que houve uma decisão do usuário por escolher entre buscar uma informação ou outra, de acordo com o conteúdo de cada campo.

```
#Filtragem dos meses
if len(meses_corrigidos) > 0:
    condicao = " | ".join([f"MES_EMI == ' {mes},'" for mes in meses_corrigidos
])
    df = df.query(condicao)

#Filtragem da CNAE
if cnae_e != '':
    df = df[df['CNAE_EMIT'].str.startswith(cnae_e)]
if cnae_d != '':
    df = df[df['CNAE_DEST'].str.startswith(cnae_d)]

#Filtragem da NCM
if ncm != '':
    df = df[df['NCM'].astype(str).str.startswith(ncm)]

#Filtragem da UF
if uf_e != '':
    df = df[df['UF_EMIT'] == uf_e]
if uf_d != '':
    df = df[df['UF_DEST'] == uf_d]

#Filtragem do Municipio
if mun_d != '':
    df = df[df['XMUN_DEST'] == mun_d]
if mun_e != '':
    df = df[df['XMUN_EMIT'] == mun_e]

#Filtragem BEC (NOVO)
if dicB:
    codigos_filtrar = dic_bec[dicB]
df = df[df['BEC'].isin(codigos_filtrar)]
```

Etapa 7: Tratamento de Valores Faltantes

Preenchimento de valores faltantes com o rótulo NAN.

```
#Preenche os valores faltantes com a string 'NA'
df = df.fillna('NA')
```

Etapa 8: Geração dos dados

Gera o dataframe de resultado com os dados filtrados pelo usuário, destacando uma mensagem de aviso caso não existam correspondências para a busca realizada. Abaixo da pre-visualização dos dados gerada após o carregamento total, é exibido também uma área de síntese do valor de produção de cada produto, que surge como um menu do tipo sanfona.

```
#Se o dataframe for vazio
if  df.empty:
    st.write(f'Nao ha correspondentes para essa busca.')
else:
    st.write(f'Notas fiscais {ano}')
    st.dataframe(df.head(),use_container_width=True)

    with st.expander('Total VPROD'):

        ui.normalizar_vprod(df)
        total = df['VPROD'].sum()
        df['PRODUTOS SCN'] = df['PRODUTOS SCN'].fillna('').str.strip().str.
upper()

        # Sintese do valor de cada produto
        st.write("Vprod Total:",total)
        for produto in scn:
            if produto in importantes:
                vprod = df.loc[df['PRODUTOS SCN'] == produto, 'VPROD'].sum(
skipna=True)

                st.write(f"{produto}: {vprod}")
```

4 Alternância entre Layouts

A troca de Layouts das opções "Principal"e de "Filtros Avançados"ocorre em pagina única, a partir de uma lógica da troca de estados de uma variável de controle apelidada de "switch", que é controlada com o auxílio do método `session_state` da biblioteca Streamlit, que permite dar dinamicidade e simular a troca de interfaces ao associar a mudança de estado da variável switch ao botão "Filtros Avançados"caso o usuário se encontre na aba Principal, e "Voltar"caso o usuário esteja na aba de filtros avançados. Abaixo segue como essa lógica é aplicada em código no contexto do Sistema de Filtragem:

No inicio do código fonte, primeiro é inicializada a variável switch caso a mesma ainda não exista, e mantém seu valor a cada nova iteração que é feita.

```
if 'switch' not in st.session_state:
     st.session_state.switch = 1
```

Ao clicar no botão rotulado por "Filtros avançados"o estado da variável switch é alterada para 0, fazendo com que na próxima execução o Layout da aba "Principal"seja limpada com o método *empty()* da biblioteca Streamlit.

```
#Aplicados ao segundo container:
    with c2.container():
    #Botao de Filtros Avancados
        with col4:
            ui.botao_padrao()
            if st.button('Filtros avancados'):
                st.session_state.switch = 0
```

Ao clicar no botão "Voltar", o estado da variável switch é alterada para 1, fazendo com que na próxima execução o layout da aba de "Filtros Avançadas"seja limpada e apareça de novo os elementos associados ao layout da aba "Principal".

```
if st.button('Voltar'):
    st.session_state.switch = 1
```

5 Instalação e Dependências(Requirements)

Para o desenvolvimento do sistema de Filtros de Notas Fiscais Eletrônicas (NFe) com foco na análise econômica necessária para o projeto da Construção das Tabelas de Recursos e de Usos (TRU), foram utilizadas apenas as bibliotecas Pandas, para o tratamento e visualização dos dados com os quais foram trabalhados, e Streamlit, para criar uma interface de usuário e trazer mais clareza aos processos realizados. Abaixo, segue o passo a passo para a execução correta do projeto:

5.1 Clone o repositório:

```
git clone https://github.com/periclessleite/gui_notas_fiscais.git
```

5.2 Instale as dependências:

```
pip install -r requirements.txt
```

Ao concluir a instalação das dependências, se faz necessário também colocar os dados das Notas Fiscais escrita no formato `"pib_{ano}_NFe.csv"`, e dos arquivo `"NCM_MERGED_LEFT.csv"`, `"SCN_CNAE_DEST.csv"` e `"SCN_CNAE_DEST.csv"`, todos em formato `csv` dentro da pasta "data", para que o Filtro funcione corretamente. Após isso, seguir com a última etapa:

5.3 Execute o programa:

```
streamlit run ./Python/app.py
```

Resumo das dependências

Pandas

Responsável pela leitura de CSVs e operações de merge.

Streamlit

Responsável pela interface, barra de progresso e gestão de estado via session_state.