Introdução

A preparação de datasets é uma etapa fundamental em qualquer projeto de análise de dados ou aprendizado de máquina. Este documento descreve, de forma detalhada, o procedimento adotado para a preparação do dataset a partir de um arquivo JSON Lines, incluindo a limpeza, filtragem, remoção de duplicatas e a divisão do conjunto de dados para facilitar seu uso em etapas subsequentes.

Objetivos

- Processar e extrair informações relevantes do arquivo JSON Lines.
- Limpar e padronizar os dados para garantir a qualidade e consistência.
- Remover entradas inválidas ou redundantes que possam comprometer análises futuras.
- Dividir o dataset em partes menores para facilitar o gerenciamento e processamento.

Descrição dos Dados

O dataset inicial está armazenado em um arquivo no formato JSON Lines (trn.json), onde cada linha representa um objeto JSON contendo, pelo menos, os campos title (título) e content (conteúdo). Este formato permite o processamento eficiente de grandes volumes de dados, uma linha por vez.

Etapas de Processamento

1. Inicialização e Preparação do Ambiente

Primeiramente, é criado uma lista vazia json_new que armazenará os dados processados. O arquivo JSON Lines é aberto em modo de leitura com codificação UTF-8 para garantir a correta interpretação dos caracteres.

```
json_new = []

with open(r'/content/drive/MyDrive/Colab Notebooks/trn.json', 'r',
encoding='utf-8') as file:
    json_new = []
```

2. Leitura e Extração dos Dados

Cada linha do arquivo é lida sequencialmente. Utilizando a biblioteca json, cada linha é convertida em um dicionário Python. Dos dados extraídos, apenas os campos title e content são preservados e adicionados à lista json_new. Linhas que

não puderem ser decodificadas corretamente são ignoradas, e uma mensagem de erro é exibida para facilitar a identificação de possíveis problemas nos dados.

```
for line in file:
    try:
        item = json.loads(line)
        new_item = {
            "title": item["title"],
            "content": item["content"]
        }
        json_new.append(new_item)
    except json.JSONDecodeError as e:
        print(f"Skipping invalid line: {line.strip()} due to error: {e}")
```

3. Conversão para DataFrame

Após a extração, a lista json_new é convertida em um DataFrame do pandas, permitindo operações mais eficientes de manipulação e análise dos dados.

```
df = pd.DataFrame(json_new)
print(df.head())
```

4. Remoção de Linhas com Valores Nulos ou Vazios

É fundamental garantir que as colunas title e content contenham informações válidas. Portanto, são removidas as linhas onde qualquer uma dessas colunas esteja vazia ou contenha apenas espaços em branco.

```
df = df[(df['title'].str.strip() != '') & (df['content'].str.strip() != '')]
print('Removida as linhas com valores nulos em title ou content')
```

5. Eliminação de Entradas Duplicadas

Para evitar redundâncias, entradas duplicadas com base nos campos title e content são removidas. Isso assegura que cada par título-conteúdo seja único no dataset final.

```
df = df.drop_duplicates(subset=['title', 'content'])
print('Removida entradas duplicadas')
```

6. Limpeza e Padronização dos Textos

Os textos das colunas title e content são padronizados para melhorar a consistência dos dados. Isso inclui a remoção de caracteres especiais e a conversão de todo o texto para minúsculas. A função clean_text é definida para realizar essas operações utilizando expressões regulares.

```
def clean_text(text):
    text = re.sub(r'[^\w\s]', '', text.strip().lower())
    return text

print('Removido os caracteres especiais')

df['title'] = df['title'].apply(clean_text)
df['content'] = df['content'].apply(clean_text)
print('Aplicada função de limpeza')
```

7. Filtragem de Descrições Curta

Descrições muito curtas podem não fornecer informações suficientes para análises futuras. Portanto, são removidas as entradas onde o conteúdo (content) possui cinco ou menos palavras.

```
df = df[df['content'].str.split().str.len() > 5]
print('Removida descrições muito curtas')
```

8. Divisão do Dataset

Para facilitar o gerenciamento e o processamento paralelo, o DataFrame resultante é dividido em quatro partes iguais utilizando a função np.array_split da biblioteca NumPy.

```
import numpy as np

df_split = np.array_split(df, 4)
```

9. Salvamento dos Dados Processados

Cada uma das quatro partes do dataset é salva em um arquivo JSON separado, nomeado de forma sequencial (json_parte_1.json, json_parte_2.json, etc.). O parâmetro force_ascii=False garante que caracteres especiais sejam preservados no arquivo final.

```
for i, df_part in enumerate(df_split, start=1):
    output_filename = f'./json_parte_{i}.json'
```

```
    df_part.to_json(output_filename, orient='records', lines=True,
force_ascii=False)
    print(f"Parte {i} dos dados foi salva em '{output_filename}'.")
```

Resultados

Após a execução do procedimento descrito, o dataset inicial foi transformado em um conjunto de dados limpo e padronizado, livre de duplicatas e entradas inválidas. A divisão em quatro partes facilita o armazenamento e o processamento subsequente, permitindo que cada parte seja utilizada de forma independente conforme a necessidade do projeto.

Considerações Finais

A preparação adequada dos dados é crucial para o sucesso de qualquer projeto de análise ou modelagem. Este procedimento assegura que os dados utilizados sejam de alta qualidade, consistentes e prontos para serem explorados em etapas posteriores. A implementação meticulosa das etapas de limpeza, filtragem e padronização contribui significativamente para a eficácia e precisão das análises futuras.