Tópicos em Engenharia de Software: Relatório Técnico

Vítor José Lara Bastos

https://github.com/vitorjoseph/Pr-tica-4-T-picos-em-Engenharia-de-Software

Introdução

Este relatório técnico aborda a importância dos testes no desenvolvimento de código que lida com grandes volumes de dados, bem como a otimização de desempenho usando PySpark. A atividade proposta tinha como objetivo principal a implementação de testes unitários para funções de processamento de dados, com foco na função avgAgeCountry e também a introdução de funções adicionais para processar e analisar os dados gerados.

1 Desenvolvimento de Testes Unitários

Os testes unitários desempenham um papel crucial no desenvolvimento de software, especialmente quando se trata de processamento de dados, eles ajudam a garantir que as funções estejam funcionando conforme o esperado, o que é fundamental ao lidar com grandes volumes de dados. A seguir, discutiremos a importância de diferentes tipos de testes e seus objetivos.

1.1 Função avgAgeCountry

A função avgAgeCountry foi o foco dos testes unitários e os cenários considerados incluem:

- **Arquivo JSON vazio:** Verificamos se a função lida adequadamente com dados ausentes, retornando um dicionário vazio quando nenhum dado está disponível.
- **Valores de idade ausentes ou nulos:** Testamos a função em dados em que a idade está ausente ou nula, garantindo que esses casos sejam tratados sem causar erros.
- **Campo 'country' ausente ou nulo:** Verificamos como a função se comporta quando o campo 'country' não está presente ou é nulo nos dados, assegurando que esses casos não causem falhas.

1.2 Outras Funções de Processamento

Além dos testes unitários, introduzimos funções adicionais para processar e analisar os dados e as funções adicionais incluem:

- **countPeopleByCountry:** Essa função conta o número de pessoas em cada país nos dados, o que é útil para entender a distribuição de pessoas por país.
- **avgAgeByCountry:** Calcula a média de idade das pessoas em cada país, o que é valioso para análises demográficas.
- **findOldestAndYoungest:** Identifica a pessoa mais velha e a mais jovem nos dados, permitindo análises sobre faixas etárias.

A importância dessas funções adicionais está em permitir análises mais profundas dos dados gerados, fornecendo insights valiosos.

1.3 Função de Transformação

A função avgAgeCountry foi modificada para aceitar uma função de transformação como segundo argumento, permitindo a aplicação de transformações à idade antes de calcular a média e um exemplo dado foi a conversão de idade de anos para meses onde a função de transformação foi testada para garantir que as transformações sejam aplicadas corretamente e que os resultados sejam precisos.

2 Objetivos dos Testes

Os objetivos dos testes desenvolvidos incluem:

- **Tipos de problemas que estavam sendo prevenidos:** Exceções não tratadas, comportamento inadequado em casos especiais, imprecisão do cálculo e conformidade com a função de transformação
- **Relevância do teste unitário e a liberdade para criar diferentes asserts:** A relevância do teste unitário reside em várias áreas, como a detecção precoce de erros, suporte a mudanças e liberdade para criar diferentes asserts.
- **A intenção de cada teste:**

  test_avgAgeCountry_empty_data: Verificar se a função lida adequadamente com dados vazios, retornando um dicionário vazio.

  test_avgAgeCountry_missing_age: Garantir que a função trate adequadamente idades ausentes ou nulas nos dados.

  test_avgAgeCountry_missing_country: Verificar o comportamento da função quando o campo "country" está ausente ou nulo nos dados.

  test_avgAgeCountry_with_transform_func: Avaliar se a função de transformação personalizada aplicada à idade antes do cálculo da média funciona corretamente.

test_read_json_file_success: Garantir que a função read_json_file funcione corretamente para arquivos válidos.

test_read_json_file_file_not_found: Confirmar que a função lide adequadamente com a ausência do arquivo.

test_read_json_file_invalid_json: Verificar como a função trata arquivos JSON inválidos.

3 Reflexão sobre Testes em Big Data

Nesta seção, discorra sobre os desafios enfrentados ao escrever testes para ambientes de Big Data. Aborde a importância de testar funções que processam grandes volumes de dados e considere a otimização de performance com PySpark. Além disso, discuta como você faria um teste de desempenho para o cenário avaliado.

4 Conclusão

Os testes unitários desempenham um papel crítico na validação e garantia da qualidade do código que lida com grandes volumes de dados. Eles ajudam a identificar erros e problemas de lógica que podem ser caros em termos de recursos de processamento quando aplicados a grandes conjuntos de dados. Além disso, funções adicionais podem extrair informações valiosas dos dados, o que é fundamental para análises e tomada de decisões.

Integrar PySpark em ambientes de big data é uma extensão natural dessa abordagem, permitindo o processamento eficiente de grandes conjuntos de dados.

Em resumo, a combinação de testes unitários e funções de processamento de dados é essencial para a criação de soluções robustas e eficientes em cenários de big data. A otimização de desempenho e a análise de dados bem-sucedidas dependem de testes sólidos e da funcionalidade correta das funções de processamento.