Projeto 2

Introdução

Neste projeto trabalharemos conceitos relacionados à instrumentalização do código para detecção e correção de possíveis gargalos. O objetivo central é detectar funções e estruturas de dados que consomem a maioria dos recursos (tempo e memória) e propor alternativas para mitigar esses problemas.

Serão utilizadas ferramentas de profiling para obter estatísticas sobre a utilização de recursos pelo programa. Depois, o código será alterado para usar estruturas de dados mais eficientes. O código será novamente avaliado e as possíveis melhorias serão confirmadas (ou não).

Do ponto de vista de estrutura de dados, trabalharemos os conceitos de referências e árvores de pesquisa. Especificamente, utilizaremos árvores Tries para amenizar o custo das comparações, e referências para reduzir o impacto de memória no armazenamento dos n-gramas do projeto anterior. Dessa forma, também trabalharemos conceitos de manutenção de software e busca por melhorias de implementações anteriores.

O que fazer?

Como dito, o objetivo é aprimorar a implementação do projeto 1. Dessa forma, você deverá:

- Instrumentar o código, através do uso de bibliotecas de profiling disponíveis para Python, para obter estatísticas sobre o uso de recursos do seu software. Você deverá utilizar: cProfile de Python para obter estatísticas de tempo das funções do seu programa (https://docs.python.org/3/library/profile.html#module-cProfile); e memory profiler (https://pypi.org/project/memory_profiler/) para obter estatísticas de uso de memória de suas estruturas de dados.
- Reduzir o custo de armazenamento dos n-gramas. Provavelmente, sua implementação atual dessa classe armazena de forma explícita todas as palavras que compõem o documento. Isso resulta em um custo de O(n^2) por documento. Contudo, repetir as palavras do documento é desnecessário, uma vez que precisamos apenas guardar de qual documento esse n-grama foi gerado e a sequência de termos/palavras que ele representa. Dessa forma, podemos apenas armazenar uma referência para o documento de origem, e um par (início,fim) demarcando o intervalo correspondente ao n-grama. O custo dessa representação é O(1), tornando-se mais atrativa para um corpus com muito documento.
- Reduzir o tempo de processamento para comparar um documento suspeito a todos os documentos do corpus. A classe Corpus deverá conter uma árvore Trie construída sobre os n-gramas dos documentos que compõem o corpus. Em outras

palavras, a chave dos elementos da árvore Trie são os n-gramas, enquanto os valores armazenados são os documentos em si. Observe que, como vários documentos podem compartilhar um n-grama, os valores armazenados devem ser listas de documentos. Tendo essa estrutura como base, você deverá modificar seu código para acelerar o cálculo da contenção do documento suspeito em todos os documentos do corpus.
- Finalmente, você deverá instrumentar novamente o código, conforme feito no primeiro item, e avaliar se houve (e de quanto foi) a melhoria em relação ao código inicial.

O que entregar?

Você deve entregar todas as classes implementadas, bem como os arquivos com os resultados da instrumentalização do código. **Não comprima os arquivos de forma alguma**. O uso de qualquer ferramenta de compressão acarretará em perdas de pontos, em particular, se a ferramenta for **rar**.

Não coloque no classroom os dados. Somente o código deve ser entregue.

Como será a avaliação?

Os seguintes pontos serão observados:

1. Completude (foram implementadas todas as funcionalidades especificadas)
2. Correção (as funcionalidade foram implementadas corretamente)
3. Organização (o código está bem organizado -- evite colocar todas as funções em um único arquivo, organize seu código em módulos https://docs.python.org/3/tutorial/modules.html)

Trabalhos plagiados da internet ou colegas serão prontamente anulados. Como não é possível saber quem fez e quem copiou, todos os envolvidos serão penalizados. Isso não significa que você não possa discutir uma solução com o colega, contudo, existe uma diferença bem grande entre pedir ajuda/discutir e copiar trechos ou o todo do outro.

Datas importantes

- Divulgação da proposta: 15/10/18
- Entrega da instrumentalização do código anterior (com análise): 24/10/18
- Entrega da alteração da classe n-grama (código funcionando): 28/10/18
- Entrega da versão final: 04/11/18

A entrega do projeto será dividida em etapas. A razão é fazer com que haja um desenvolvimento constante do projeto. A intenção é evitar que você deixe para construir o projeto nos últimos dias e, com isso, não consiga executá-lo. Dessa forma, atrasos parciais contarão negativamente para a nota final. Haverá penalização de 10% por dia para as etapas iniciais. Não serão aceitas versões finais entregues com atraso.