Trabalho 2 – versão 2.1 – publicada em 25/Maio/2023

Rede do Metro de Londres: visualização e estudo da rede usando um grafo

Atualizações deste enunciado em 25/Maio/2023:

- retificação substituindo em 2 e 2d) “curto” por “rápido”
- a data limite de entrega do trabalho passa para 4ª feira, 31/Maio/2023 às 23h59
- o número máximo de páginas do relatório é 15, incluindo anexos.

Figura 1 Diagrama que representa a rede do Metro de Londres (https://tfl.gov.uk/maps/track/tube).

Este trabalho tem como objetivo representar, visualizar e analisar informação sobre a rede do metro de Londres utilizando um grafo.

Este trabalho deve utilizar os conjuntos de dados (*datasets*) fornecidos em Stations.csv e Connections.csv. A implementação do grafo que representa a rede do metro de Londres deve ser feita em Python e deve possuir uma funcionalidade de visualização do grafo utilizando bibliotecas adequadas a visualização de grafos (como por exemplo a NetworkX). Esta implementação do grafo que representa a rede de metro de Londres será utilizada para fazer uma simulação em Python que permita estudar caminhos mais curtos entre duas estações através do cálculo destes caminhos e sua visualização.

O trabalho desenvolvido deve ser descrito num **relatório** que inclua:

A. Introdução
B. Desenvolvimento: Métodos e Resultados
C. Discussão e Conclusões

O número máximo de páginas do relatório é 15, incluindo anexos. Para mais detalhe sobre o que pode/deve incluir em cada uma das partes do relatório consultar o anexo “Conteúdo do relatório”.

No desenvolvimento devem ser executadas as duas fases seguintes:

1. Implementação de uma classe LondonNetworkGraph com visualização usando bibliotecas de visualização de grafos

A classe **LondonNetworkGraph:**

a) deve ser implementada em Python, com base na implementação do TAD Grafo fornecida em Graph.zip que se encontra no material relativo à semana 8 (24 a 28/Abr);
b) o seu atributo principal deve ser um grafo direcionado do NetworkX (https://networkx.org/);
c) deve ter as seguintes funcionalidades implementadas como métodos da classe:
    1. __init__(): inicializar o grafo direcionado do NetworkX
    2. stations(file_path): importar estações para o grafo (com todas os dados associados)
    3. connections(file_path): importar ligações para o grafo (com todas os dados associados)
    4. n_stations(): devolver o número de total de estações
    5. n_stations_zone(): devolver o número de estações de cada zona
    6. n_edges(): devolver o número total de edges
    7. n_edges_line(): devolver o número de edges por linha
    8. mean_degree(): devolver o grau médio das estações
    9. mean_weigth(weight): devolver o peso médio das conexões (dado o tipo de peso)
    10. visualize(): visualização do grafo, usando
        i) NetworkX (https://networkx.org/)
        ii) Matplotlib (https://matplotlib.org/)
        iii) Folium (https://python-visualization.github.io/folium/#).

Aconselhamos os alunos a procurarem ter esta 1ª fase do desenvolvimento pronta na semana de 15 a 19/maio/2023, bem como a parte respetiva do relatório.

2. Desenvolver uma simulação em Python para calcular e visualizar o caminho mais rápido entre duas estações, usando o algoritmo de Dijkstra

O objetivo da 2ª fase do trabalho é desenvolver uma simulação que tem os seguintes passos:

a) Gerar aleatoriamente dois pontos (start, end) entre x1 e x2, y1 e y2
b) Gerar aleatoriamente uma hora do dia
c) Para cada um dos pontos, calcular qual a estação mais próxima
d) Calcular qual o caminho mais rápido entre essas duas estações dada a hora do dia
    i. Usando o método do NetworkX que implementa o algoritmo de Dijkstra
    ii. Implementar em Python o algoritmo de Dijkstra (4 valores)
        - Atenção que para esta implementação é necessário definir na classe **LondonNetworkGraph** atributos para guardar toda a informação sobre o grafo necessária. Isto é para esta implementação não é possível utilizar apenas o atributo que guarda a instância do grafo criada chamando o NetworkX.
        - Mesmo que não seja possível colocar o algoritmo Dijkstra feito em Python a funcionar deve entregar todo o seu código que fez (original) comentado. Esse código, desde que seja original e contiver comentários explicativos esclarecedores do que as principais instruções fazem será valorizado
e) Desenvolver um método de visualização que mostra todo o grafo, os dois pontos gerados e ainda pinta o caminho ótimo entre o start e o end.

Qualidade do código Python

O desenvolvimento do código Python de respeitar as normas PEP 8.

Submissão do trabalho incluindo relatório

O trabalho deve ser distribuído entre os 2 elementos do grupo, devendo estar equitativamente distribuído. Cada grupo deve submeter o resultado do seu trabalho até à data e hora limite de entrega: 31 de maio de 2023, 23h59m. A submissão do trabalho deve ser feita no Moodle na Atividade do Trabalho 2, fazendo o carregamento (upload) do ficheiro ZIP com o conteúdo descrito de seguida.

Para a submissão, devem entregar um zip com:

- o relatório num formato digital que vos pareça adequado, que deve incluir o código Python desenvolvido (ou, pelo menos, as partes importantes) e os restantes elementos pedidos acima; o relatório deve seguir a estrutura indicada na seção sobre estrutura do relatório;
- um ficheiro com o código Python desenvolvido, em formato (.py) ou no formato (.ipynb) pronto a ser utilizado/testado.

A partilha de código em trabalhos de grupos diferentes será devidamente penalizada. Serão usados os meios habituais de verificação de plágio. O código entregue deve ser integralmente da autoria dos membros do grupo.

Anexo: Conteúdo do relatório

De seguida são descritos alguns aspetos que podem/devem (conforme pertinente) ser incluídos no relatório. Este conteúdo foi adaptado do enunciado do Projeto Final para Avaliação da UC de Análise de Redes Avançada (Prof. Jorge Louçã, Prof. António Fonseca) do ano 2022/2023.

A. Introdução

A.1. Descrição/Definição do Problema

É verdade que definir o problema, o que queremos alcançar e/ou que questões pretendemos responder não é fácil. Nesta fase é essencial pensarmos no contexto que vamos analisar, se estamos à vontade com ele, que informação já temos, e como podemos obter mais informação através das ferramentas e do conhecimento de que dispomos. Devemos abordar problemas que consigamos resolver, mas sem colocar soluções fáceis em detrimento da relevância das respostas que podemos obter.

A.2. Análise do que já foi estudado sobre o problema

Um passo importante é o levantamento daquilo que já se sabe sobre o problema. Se as soluções são triviais, ou o problema já foi muito estudado, o nosso contributo pode não ser muito grande e o problema pode até já estar resolvido. É importante ler e aprender com o trabalho que já foi desenvolvido em torno do problema.

B. Desenvolvimento: Métodos e Resultados

B.1. Aquisição de Dados

Embora isso possa parecer um passo simples para aqueles que têm experiência anterior com dados (ou que trabalham com eles diariamente), pode ser um desafio significativo para quem quer investigar e resolver um problema específico que envolve de recolha de informação. Neste projeto os dados a utilizar já estão disponíveis em *datasets* normalmente bem formatados. No entanto, dependendo do problema colocado e da forma como o vão resolver, podem ter de recolher informação adicional em *datasets* adicionais e de ajustar formatos ou conciliar os dados entre *datasets*.

B.2. Limpeza dos dados

Este processo consiste em limpar, manipular e transformar dados considerados brutos, em dados que podem ser lidos/processados por, por exemplo, scripts em Python. É um processo necessário quando queremos 'filtrar' nossos dados para servir de base à resolução do problema, eliminando possíveis dados incorretos e/ou nulos, formatos incorretos e/ou inválidos, selecionando apenas colunas interessantes e

descartando aqueles que não são interessantes, detetando possíveis *outliers*, entre outros. É o resultado deste processo que nos permite obter visualizações limpas dos dados, bons modelos e também o contrário: pode ser a razão pela qual os algoritmos não têm os resultados esperados.

B.3. Análise dos dados

É na sequência desta fase que podemos tirar conclusões sobre os nossos dados. Nesta fase são utilizados algoritmos e ferramentas que são aplicados sobre os dados através de ferramentas de *software*, ou através de *scripts* programados para o efeito. É nesta fase que são calculadas métricas e estatísticas sobre os dados que vão suportar a relevância estatística dos resultados. Uma análise bem feita dos dados não se deve basear unicamente sobre resultados quantitativos, mas também na análise das possíveis relações qualitativas existentes nos mesmos através do exame criterioso daquilo que os dados representam.

Muitas vezes nesta fase volta-se a estágios anteriores, para recolher mais dados, ou outros dados ou mais informação sobre os mesmos. Os métodos também são revistos em função dos resultados parcelares obtidos.

B.4. Recolha de resultados

Quando se tratam dados em grande quantidade a recolha de resultados pode constituir uma tarefa morosa. A recolha de resultados nestes casos pode constitui um passo importante. É também nesta fase que são agregados resultados para visualização gráfica.

B.5. Descrição da análise de resultados e Visualização

Este procedimento vem completar o nosso processo de análise com a descrição textual da análise de resultados e a elaboração de gráficos e outras visualizações. Esta fase é muito importante porque a qualidade desta descrição (textual e visual) tem um grande impacto naquilo que o vosso trabalho pode representar. Devemos procurar ser claros, objetivos e ler o nosso relatório através do olhar dos outros procurando ter em conta o seu sentido subjetivo e a sua facilidade de apreensão.

C. Discussão e conclusões

O processo culmina na extração de conclusões sobre os resultados da nossa análise. Se tivermos muitas informações sobre os dados, agora podemos passar para a etapa que nos permite apresentar a solução final para o nosso problema, que pode dar resposta às questões nele colocadas. É fundamental lembrar que os algoritmos não são varinhas mágicas. Nós, como cientistas de dados, temos de saber interpretar os resultados dos nossos algoritmos e métodos.

Nesta fase também se colocam mais hipóteses sobre outras alternativas de trabalho, discute-se a validade das conclusões em função das circunstâncias em que elas foram tomadas e desenha-se um trabalho futuro.