

MPEI – Trabalho Prático

INFORMAÇÕES

Este projecto está a ser realizado por Mário Alexandre Lopes Liberato (NMEC: 84917) e Jorge Miguel Aires de Matos Oliveira (NMEC: 84983). O projecto pretende utilizar alguns módulos desenvolvidos no âmbito dos guiões práticos da unidade curricular de Métodos Probabilísticos para Engenharia Informática, convertidos para Java quando conveniente. Quando não o for, pretende-se utilizar a biblioteca JavaOctave ou outra que sirva o mesmo propósito, a fim de manter o projecto maioritariamente escrito em Java. Em último caso, utilizar-se-á apenas um script Octave.

CONCEITO INICIAL

Programa que sirva para gerir aplicações de pessoas a um determinado emprego. O lado do potencial trabalhador fornece um questionário que depois de completo adiciona os dados a um ficheiro ou base de dados simples (Exemplo: USER ID, SKILLSET ID, EDUCATION LEVEL ID, TIMESTAMP), guardado em CSV. Este ficheiro é processado pelo programa do empregador, que cria, para cada emprego diferente, um modelo de funcionário com o skillset ideal, com o nível de educação ideal, etc. O programa depois filtrará todos os empregados pelo seu índice de semelhança e opcionalmente eliminar aqueles que não possuem semelhança alguma(ou baixa semelhança).

DESCRIÇÃO BREVE DO PROGRAMA

Três programas são utilizados. Um deles será o programa utilizado pela pessoa que submete a aplicação à empresa. Este é denominado por de programa B no documento. O segundo, o programa A, é utilizado pelo empregador para gerir todas as aplicações aos empregos disponíveis. O último programa, dito programa C, serve para gerar dados simulados para utilização pelo programa A.

DESCRIÇÃO BREVE DO PROGRAMA B

Este programa contém uma interface simples, com um questionário com escolhas múltiplas. Quando o questionário é submetido, este gera dados em CSV, os quais são adicionados ao ficheiro principal, no formato seguinte:

- USER_ID, SKILLS_ID, EDUCATION_LEVEL_ID, USER_NAME, USER_BIRTHDATE, TIMESTAMP

Estes dados serão armazenados em dois ficheiros CSV. Idealmente, em vez disso, seriam armazenados numa única base de dados que use duas tabelas. Dependendo do progresso do projecto até à data de entrega, esta funcionalidade poderá ou não ser implementada.

Descrição dos campos das tabelas

- USER_ID – ID do utilizador, atribuído por ordem crescente. Na segunda tabela, os dados da linha X correspondem ao utilizador com ID X+1.
- SKILLS_ID – IDs das capacidades.
- EDUCATION_LEVEL_ID – Nível de educação (0 – Nenhum, 1 – Primário, 2 – Secundário, 3 – Licenciatura, 4 – Mestrado, 5 – Doutoramento).
- USER_NAME – Nome completo do utilizador.
- USER_BIRTHDATE – Data de nascimento do utilizador em segundos.
- TIMESTAMP – A timestamp de criação dos dados em milissegundos.

DESCRIÇÃO BREVE DO PROGRAMA A

Este programa trata do processamento dos dados obtidos. Essencialmente, lê os dados da primeira tabela, constrói uma lista dos skillsets de cada pessoa, insere os “ideais” como novas pessoas no fim da lista, calcula a distância de Jaccard entre todos os utilizadores e filtra todos os que estiverem abaixo de um determinado nível de semelhança. Depois apresentará os utilizadores mais relevantes para cada emprego, por ordem de nível de escolaridade.

A interface é simples, e consiste apenas numa opção que serve para iniciar o processamento. Eventualmente, esta interface poderá ser expandida e aumentar em complexidade, mas não faz parte do plano inicial.

O programa utiliza outra tabela que contém os empregos (objectos Job) em formato CSV:

- SKILLS_ID0,EDUCATION_LEVEL_ID0,SKILLS_ID1,EDUCATION_LEVEL_ID1,JOB_TITLE,MIN_SIM_INDEX

Os primeiros campos dizem respeito ao utilizador ideal para esse emprego, os segundos ao utilizador mínimo necessário (todos os que não cumpram os requisitos são excluídos independentemente do resto dos seus parâmetros), JOB_TITLE ao nome do emprego e MIN_SIM_INDEX ao índice de semelhança mínimo.

DESCRIÇÃO BREVE DO PROGRAMA C

Este programa gera dados aleatórios para os ficheiros utilizados pelo programa A. Este programa pede apenas o número de utilizadores que se deve criar e gera esse número de utilizadores para um ficheiro. É gerado usando um programa escrito em Java.

FUNCIONAMENTO DO PROGRAMA A

CLASSE BLOOMFILTER

A classe BloomFilter é uma adaptação da versão escrita em Octave para os guiões das aulas práticas da unidade curricular. Esta classe implementa um “Bloom Filter”. Utiliza a função de dispersão “Djb2”, tal como a versão na qual se baseia.

Esta classe contém um array que contém o bloom filter propriamente dito, denominado de bloomFilter. Esta variável é privada e só pode ser alterada pelos métodos implementados.

Para inicializar o bloom filter, deve-se criar um novo objecto do tipo BloomFilter, e fornecer-lhe como argumento o tamanho desejado. O construtor inicializa então o array com zeros.

Para obter o array, deve-se recorrer ao método getArray, que retorna o dito-cujo.

Para obter o tamanho, deve-se utilizar o método getSize, que retorna a length do array bloomFilter.

Para adicionar um membro deve-se utilizar o método addMember, que aceita como argumento o o número de funções de dispersão e a String que se pretende adicionar. Para remover um membro ou verificar se ele existe, os mesmos argumentos são fornecidos, e os métodos são removeMember e existsMember, respectivamente. O último retorna um valor de true ou false, dependendo da não existência ou possível existência do membro.

CLASSE COUNTINGFILTER

A classe CountingFilter estende a classe BloomFilter com dois métodos que tornam o bloom filter num counting filter. Não é utilizado pelo programa.

CLASSE DJB2

A classe DjB2 contém apenas uma função (hash), que utiliza o algoritmo DjB2 para calcular a hash de uma String.

CLASSE JACCARD

Esta classe contém funções para calcular a similaridade e/ou a distância de Jaccard entre conjuntos ou listas de conjuntos. É utilizada para calcular o índice de similaridade das assinaturas de todos os conjuntos de skills.

CLASSE MINHASH

Esta classe contém funções para calcular uma assinatura de um conjunto. O tamanho da assinatura utilizado pelo programa é a média do tamanho de todos os conjuntos - 1.

CLASSE USER

Esta classe define um objecto que contém campos e métodos relacionados com cada pessoa que é lida do ficheiro. Utiliza um bloom filter e uma lista para armazenar os dados relativos às skills. Contém métodos para verificar a existência de skills no utilizador. O método hasSkill utiliza o bloom filter, o método hasSkillExact itera sobre a lista.

CLASSE JOB

Esta classe contém dois objectos Users, um chamado de “ideal” e um chamado de “mínimo”. Esta classe tem ainda o nome do emprego e o índice de semelhança mínimo. Ela contém os dados necessários para a eliminação e ordenação dos dados no programa principal.

CLASSE MAIN

Esta classe contém o programa principal, o qual corre um loop, uma vez para cada Job presente na lista de empregos (“jobs.tb”). Em primeiro lugar, primeiro elimina os utilizadores que não têm pelo menos as skills mínimas para o emprego. Para fazer isto, obtém uma lista de utilizadores que cumprem os requisitos utilizando o bloom filter de cada um e dessa lista obtém os utilizadores que cumprem os requisitos usando a lista. O algoritmo probabilístico é usado para reduzir o tamanho da lista, e o algoritmo exacto é apenas usado nesta lista reduzida (para eliminar falsos positivos). De seguida, são obtidos os utilizadores que cumprem os requisitos ideais de forma semelhante. Estes são então ordenados por ordem de relevância(número de skills, nível de educação), armazenados numa outra lista e removidos da lista de utilizadores mínimos.

De seguida procede-se ao cálculo da semelhança de utilizadores. É utilizada a classe MinHash2 para calcular as signatures da intersecção do conjunto de skills dos utilizadores restantes com o conjunto de skills ideal (para obter todos os utilizadores que cumprem os requisitos ideais), e depois a signature do utilizador ideal. Depois disto, é calculado o índice de semelhança de Jaccard para todos em relação ao ideal, e os utilizadores são ordenados por nível de semelhança, depois por número de skills e nível de educação. Os utilizadores com índice de semelhança inferior ao índice de semelhança mínimo são removidos da lista.

Por fim, os dados são escritos, para cada Job, em dois ficheiros CSV, os quais podem ser abertos com um programa como o LibreOffice Calc (O Microsoft Office Excel pode precisar de reconfiguração ou de uma linha “sep=,” no início para mostrar os dados correctamente formatados) e então analisados manualmente. Um ficheiro diz respeito aos utilizadores perfeitos e o outro aos utilizadores ordenados por nível de semelhança.

Outras classes foram incluídas, mas não são muito importantes.

FUNCIONAMENTO DO PROGRAMA B

O programa B consiste numa interface gráfica em Java, utilizando Swing (classe JobApplicationUI).

Na situação em que o projecto se enquadra, o programa B seria utilizado pela pessoa que se pretende candidatar a um emprego numa empresa. Adiciona os dados ao ficheiro table.tb ou cria-o se este não existir. Utiliza o mesmo formato escrito pelo programa C e lido pelo programa A.

FUNCIONAMENTO DO PROGRAMA C

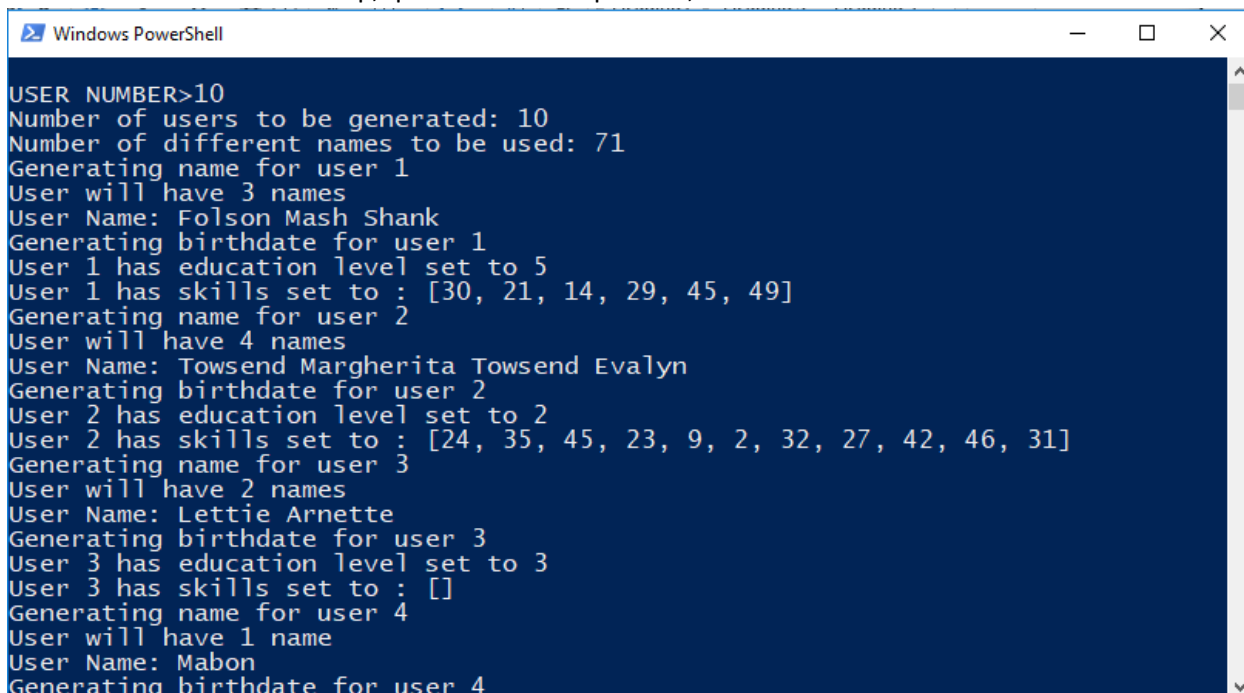
PARTE 1 – GERAR OS VALORES

O programa C serve para gerar dados aleatórios. Para atingir este fim, precisa de um argumento, que é o número de utilizadores gerados. Se não for fornecido um argumento, o programa pedirá que um valor seja inserido. Depois, lê um ficheiro “names.txt”, que deverá conter **apenas** nomes, mais especificamente um nome por cada linha, os quais são depois utilizados para gerar nomes completos, os quais variam entre 1 e 4 nomes no total. Os nomes do ficheiro foram escritos pelos desenvolvedores do projecto ou obtidos a partir de um gerador de nomes aleatório. Como apenas existe para propósito de testes, não há qualquer problema no facto dos nomes não serem coerentes e misturarem várias línguas.

É utilizado um ciclo for para gerar cada utilizador, e este é adicionado ao ArrayList de objectos User denominado de userList no fim deste. Depois do nome ser gerado, é gerada a timestamp do nascimento (em segundos), o qual varia entre 657170474 e 909631274. Não há propriamente uma razão para a escolha destes valores, uma delas é apenas a data de nascimento de ML, e a outra é cerca de 8 anos antes. Assim, as idades dos utilizadores aleatórios deve variar entre 19 e 27 anos.

Os dados são escritos para um ficheiro, “table.tb”, que contém dados em CSV relativos a todos os utilizadores.

O nível de educação é gerado de seguida, sendo um simples valor inteiro entre 0 e 5. Posteriormente, o utilizador recebe skills aleatórias, as quais variam entre 1 e 50 (havendo então 50 skills separadas). Neste programa, cada utilizador recebe um máximo de 13 skills, e pode ter 0. Esta limitação não é parte dos outros programas. No caso do programa B, mais skills podem ser escolhidas, e no caso do programa A, mais skills podem ser processadas. Tanto o limite de 13 como as 50 skills diferentes são valores arbitrários, e apenas este último se apresenta nos outros programas. Por fim, o utilizador recebe uma timestamp, que é a timestamp actual, e é adicionado à lista.



```
Windows PowerShell
USER NUMBER>10
Number of users to be generated: 10
Number of different names to be used: 71
Generating name for user 1
User will have 3 names
User Name: Folsen Mash Shank
Generating birthdate for user 1
User 1 has education level set to 5
User 1 has skills set to : [30, 21, 14, 29, 45, 49]
Generating name for user 2
User will have 4 names
User Name: Towsend Margherita Towsend Evalyn
Generating birthdate for user 2
User 2 has education level set to 2
User 2 has skills set to : [24, 35, 45, 23, 9, 2, 32, 27, 42, 46, 31]
Generating name for user 3
User will have 2 names
User Name: Lettie Arnette
Generating birthdate for user 3
User 3 has education level set to 3
User 3 has skills set to : []
Generating name for user 4
User will have 1 name
User Name: Mabon
Generating birthdate for user 4
```

Imagem 1 – Exemplo do output do programa para o terminal

PARTE 2 – ARMAZENAR OS VALORES

O programa C armazena então os valores num ficheiro “table.tb” que cria, que pode ser usado pelo programa A para obter os dados, que se encontram no formato descrito antes.