**MPEI – Trabalho Prático**

Relatório

Gonçalo Sousa 98152

João Rodrigues 103947

Data: 16/12/2024

**Descrição do Trabalho Prático**

O nosso trabalho prático passa pela utilização dos conceitos do Naïve Bayes, Bloom Filter e MinHash lecionados na cadeira de MPEI. Assim sendo, apresentamos uma aplicação dos conhecimentos adquiridos relativos a estes. O trabalho subdivide-se em 3 secções (Módulos, Testes e Aplicação Conjunta) que passaremos a descrever:

Suponhamos um sistema de visualização de vídeos, como, por exemplo, o Youtube. Neste, existe um conjunto de vídeos armazenados, acessíveis a qualquer utilizador, que lhe são recomendados consoante o algoritmo existente. O nosso trabalho ambiciona criar um programa de recomendação dos vídeos armazenados baseado no histórico de vídeos do utilizador, bem como atribuir uma classificação de categoria a novos vídeos tendo como referência os vídeos já existentes no dataset.

# **Fluxo do Sistema**

Antes de mais, o sistema tem acesso a um dataset com bastantes vídeos à priori, que contém informação relevante aos nossos módulos, como id, título, nome de canal, categorias, tags e descrições.

Assim sendo, o primeiro passo do sistema é a adição de novos vídeos a este dataset e a sua classificação. Este passo simula a introdução de novos vídeos num sistema de visualização de vídeos. Para isso, o Naïve Bayes é utilizado. Baseado nas palavras dos títulos, tags e descrições, uma vez filtradas, associadas às categorias, calcula-se a categoria mais provável para o vídeo a ser analisado.

De seguida, começamos o processo de recomendação de um vídeo ao utilizador. É analisado o histórico de vídeos visualizados por este e, utilizando um Bloom Filter, removem-se vídeos já vistos daqueles que poderão ser recomendados, garantindo a recomendação de novo conteúdo ao utilizador.

Finalmente, procuram-se vídeos similares aos que agradam ao utilizador com o MinHash. Partindo das palavras dos títulos, faz-se os shingles destes com k = 5, calculam-se as assinaturas com funções de dispersão, obtêm-se pares candidatos e encontram-se os pares similares, ou seja, os vídeos que mais se assemelham aos vistos pelo utilizador.

# **Descrição dos Módulos**

Como descrito acima, este trabalho contém os módulos Naïve Bayes, BloomFilter e MinHash, feitos no Matlab. Todo o código apresentado (fora caso explicitado na secção seguinte) está devidamente pronto e a funcionar. A descrição de cada uma das funções incluídas é a seguinte:

**Módulo Naïve Bayes**

Composto por 2 ficheiros principais (script NB\_test.m e função NB.m) e 5 auxiliares (restantes ficheiros que começam com NB).

- ***NB\_test***

Script de teste do NB. Retorna ficheiros de texto na pasta testes com a precision, recall e F1 correspondentes ao NB utilizado. Permuta sempre o dataset para apresentar resultados úteis.

- ***NB***

Função principal do Naïve Bayes que classifica documentos em categorias.

Input: Data (dataset dos vídeos), teste\_row (número de linhas de teste), treino\_row (número de linhas de treino), Cat (categorias a utilizar), flag\_tags (flag de utilização de tags), docs\_col\_n (número mínimo de ocorrência de palavras para se considerar a palavra), flag\_perm (flag de permutação dos dados)

Output: classes\_added (cell array coluna classes adicionadas), classes\_corretas (cell array coluna classes do dataset original), Cat (categorias a utilizar), flag\_tags (flag de utilização de tags), docs\_col\_n (número mínimo de ocorrência de palavras no dataset para se considerar a palavra), flag\_perm (flag de permutação dos dados).

- ***NB\_obter\_dataset\_teste***

Dado um dataset, obtém uma versão reduzida deste, removendo as linhas utilizadas.

Input: Dataset (conjunto de dados a ser utilizado e reduzido), Cat (categorias que indicam as linhas de dados relevantes), Total\_rows (número total de linhas a utilizar)

Output: dataset\_teste (conjuntos de dados com linhas relevantes ao NB), aux\_dataset (counter das linhas do dataset\_teste), classes\_corretas (conjunto das classes corretas (utilizado só nos testes))

- ***NB\_obter\_dataset\_treino***

Dado um dataset, obtém uma versão reduzida deste.

Input: Dataset (conjunto de dados a ser utilizado), Aux\_dataset (contador para auxiliar na função), Cat (categorias que indicam as linhas de dados relevantes), Total\_rows (número total de linhas a utilizar)

Output: dataset\_treino (conjuntos de dados com linhas relevantes ao NB), classes\_treino (conjunto das classes treino relativo ao dataset\_treino)

- ***NB\_obter\_palavras\_unicas***

Input: flag\_filter (flag para aplicar só o filtro ao dataset (devia ser função individual)), flag\_treino (flag para indicar se é necessário os documentos\_C2), classes\_treino (fornece as classes relativas ao dataset de treino, se flag\_treino = true), titles (coluna de títulos), descriptions (coluna das descrições), varargin (coluna das tags. Pode não existir).

Output: documentos\_C1 (conjunto de documentos da classe C1), documentos\_C2 (conjunto de documentos da classe C2), palavras\_unicas\_C1 (conjunto de palavras unicas da classe C1), palavras\_unicas\_C2 (conjunto de palavras unicas da classe C2), Data (Dataset caso se utilize a flag\_filter)

- ***NB\_obter\_num\_ocorrencias***

Input: documentosC1 (conjunto de documentos da classe C1), documentosC2 (conjunto de documentos da classe C2), palavras\_unicasC1 (lista com palavras unicas de C1), palavras\_unicasC2 (lista com palavras unicas de C2)

Output: OCORRENCIA\_C1 (conjunto de num ocorrencias de palavras da classe C1), OCORRENCIA\_C2 (conjunto de num ocorrencias de palavras da classe C2), palavras\_unicas\_treino (lista com palavras unicas de treino)

- ***NB\_calculo\_classes***

Input: classes\_treino (lista de classes de treino), Cat (categorias que indicam as linhas de dados relevantes), OCORRENCIAS\_treinoC1 (conjunto de num ocorrencias de palavras da classe C1 nos documentos de treino), OCORRENCIAS\_treinoC2 (conjunto de num ocorrencias de palavras da classe C2 nos documentos de treino), documentos\_teste (conjunto de documentos de teste), palavras\_unicas\_teste (lista com palavras unicas de teste), palavras\_unicas\_treinoC1 (lista com palavras unicas de treino da classe C1), palavras\_unicas\_treinoC2 (lista com palavras unicas de treino da classe C2), n\_max\_word (número máximo de palavras unicas das classes)

Output: res (lista probabilidades das classes)

## · **Módulo Bloom Filter**

Utilizado para verificar a pertença de um elemento num conjunto.

N - número de elementos do Bloom Filter

M - número de membros (vídeos)

K =número de funções de dispersão

- ***BF\_initialize***

Inicializa o Bloom Filter para N elementos.

- ***BF\_add***

Adiciona um elemento ao Bloom Filter. Output: Bloom Filter com o elemento adicionado

- ***BF\_isMember***

Verifica se um elemento pertence ao Bloom Filter. Output: True caso pertença,False caso contrário.

- ***BF\_test***

Script de teste do Bloom Filter. Verifica se um conjunto de videos pertence ao his

## · **Módulo Finding Similar Items**

- ***MH\_criar\_conjunto***

Baseado num conjunto, devolve a lista dos singles desse conjunto

***- calcular\_distancia\_MinHash\_shingle***

Dado um conjunto, calcula a distância de Jaccard baseado nos shingles.

- ***MH\_distancia\_minima***

Utiliza as demais funções, devolvendo a distância mínima de Jaccard calculada.

## · **Aplicação Conjunta – main.m**

Para a execução da aplicação conjunta, foi criado um script que chama as funções dos módulos e segue o fluxo original anteriormente mencionado. É portanto um script que baseado no dataset de vídeos e o histórico do utilizador, lhe recomenda vídeos semelhantes aos que já viu.

Primeiramente, é necessário obter os datasets utilizados. Normalmente, utiliza-se a função load\_dataset para obter ambos, no entanto, devido a demorar algum tempo, vamos utilizar diretamente a função load nos ficheiros de dados guardados (Data do Dataset\_reduzido.mat e x do Dataset\_histórico.mat).

De seguida, aplica-se o Naïve Bayes no dataset Data previamente adquirido. Após a sua execução temos acesso às colunas do Data utilizadas no naïve Bayes.

O passo seguinte passa por utilizar o Bloom Filter de modo a remover vídeos já visualizados das recomendações finais.

Finalmente, aplica-se o MinHash para encontrar vídeos semelhantes aos do histórico do utilizador.

# **Resultados:**

Quanto ao Naïve Bayes, os resultados quando se considera a classe “Music” temos um F1 respeitável de aproximadamente 67%. No entanto, para os restantes o mesmo não se verifica, pelo que só temos valores bastante mais baixos

# 

# **Comentários:**

Deparamo-nos com dificuldades a nível da implementação do Naïve Bayes e do MinHash. No Naïve Bayes ocorreu o erro de não ser realizada a remoção das linhas de teste do dataset utilizado duas vezes ao longo da implementação. Este erro fazia com que as linhas de teste tivessem dados iguais aos de treino num número de linhas substancial, o que aumentava os resultados obtidos. A sua demorada deteção destas duas vezes fez com que a correção do método usado para calcular o Naïve Bayes fosse alterado bastante tarde no processo. Já no MinHash houve uma demorada compreensão do funcionamento deste por nós e a sua implementação final

Adicionalmente, houve funcionalidades que queríamos implementar que não tivemos tempo para fazer, mais especificamente, adicionar um counting bloom filter relacionado a um dataset de utilizadores para calcular visualizações de vídeos e utilizar visualizações como parâmetro no MinHash (vídeo com mais visualizações era recomendado antes de um vídeo completamente igual, mas com menor valor).

Finalmente, houve ideias que tivemos que simplesmente descartar por não funcionarem do método que antecipavamos. Um exemplo seria o de utilizar mais de duas classes para calcular o Naïve Bayes, no entanto, os resultados não eram suficientemente bons para justificar a sua existência.

Dito isto, pensamos que o trabalho se encontra de acordo com o pretendido e nos ajudou tremendamente a dominar os conceitos sobre estes módulos, se bem que não nos aprofundamos ao nível que pretendíamos.