Classificação e Pesquisa de Dados

Trabalho Final

Eduardo Renani Ribeiro, 262701

1. Funcionalidade básica (70 % da nota)

Como funcionalidade básica, foi lido do arquivo de texto linha-a-linha os comentários, tokenizando os mesmos e inserindo em um árvore TRIE, junto com demais dados satélites. Além do arquivo principal com comentários, foi lido e armazenado em outra árvore TRIE um arquivo contendo Stop Words da língua inglesa.
Para isto foi utilizada, em especial, apenas a função de inserção na árvore. insert_trie();

1.1. Cálculo do escore das palavras (40%)

Para calcular o escore da palavra foi calculada a média aritmética dos comentários associados aquela palavra. Além de um campo para o escore, foi implementado um campo do tipo lista para guardar todos os n valores de escore e n comentários associados para uso futuro.

1.2. Classificação de novos comentários de filmes (20%)

O cálculo da classificação de novos comentários foi feito utilizando de um indicador de dispersão do conjunto de notas associado a uma palavra. Seguindo um raciocínio de que palavras com alta dispersão de nota tendem a ter menos influência no sentimento do comentário, foi calculado o **coeficiente de variação** do conjunto de notas de cada palavra, sendo este valor um float de 0 a 1, onde 0 é completamente homogêneo e 1 completamente heterogêneo. A partir daí, foi calculada a **média ponderada** dos escores das palavras presentes no novo comentário. Porém, utilizando de (1 - coeficiente de variação) como peso para ponderar a média.
Assim, temos que:
Para dado comentário:
C = {“great”,”piece”,”of”,”art”}
extraímos da TRIE o conjunto de notas associado a cada palavra:

Sa = {s1, s2, ..., sN}, …,Sd={s1,s2,...,sN}.

Calculamos a media aritmética de cada conjunto de notas de cada palavra:

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Ma,Mb,Mc,Md

Calculamos o desvio padrão de cada conjunto:

$$s_{n-1} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$ (n-1) por conta da correção de bessel (desvio amostral)

STa,STb,STc,STd

Calculamos o coeficiente de variação de cada conjunto:

$$c_v = \frac{\sigma}{\mu}.$$

CVa,CVb,CVc, CVd

Calculamos a média ponderada dos escores de cada palavra:

$$\bar{p} = \frac{x_1 p_1 + x_2 p_2 + \ldots + x_n p_n}{p_1 + p_2 + \ldots + p_n}$$

Mp= (1 - CVa)*(Ma)+(1 - CVb)*(Mb)+(1 - CVc)*(Mc)+(1 - CVd)*(Md) /

(1 - CVa)+(1 - CVb)+(1 - CVc)+(1 - CVd)

1.3. Identificar Extremos e Ocorrências (10%)

Para identificar os K mais positivos, K mais negativos e K mais frequentes, percorremos toda a árvore usando a função print_trie() (com parâmetro bool = 1, que significa que irá printar todos os nodos, nota média e número de ocorrências em um arquivo .txt). Após isso, será carregado em uma estrutura os dados e usado o algoritmo HeapSort para ordenação das K-chaves.

2. Adicionais (mais 50% da nota)

Para esta parte do programa, são introduzidas funções de ordenação (heapsort), leitura e escrita de arquivos e busca parcial na árvore.

2.1. Teste a partir de um arquivo de comentários (5%)

Como previsto no menu (switch-case) implementado no início da aplicação, existe um opção para escrever o nome de um arquivo de teste, o qual será lido linha-a-linha e tokenizado o comentário. Cada token gerado no while do strtok será uma palavra a ser procurada na TRIE, retornando a localização e os campos para efetuar o cálculo do coeficiente de variação e outros cálculos (já descritos) até chegar ao resultado da classificação deste comentário. As classificações são printadas na tela e ao final da iteração é printado o erro médio da predição de classificação.

2.2. Melhorar a classificação dos comentários (15 %)

Na função holdout_method(), são implementadas 3 funções de predição de nota para comentário. Uma extremamente simples, sem usar o coeficiente de variação, uma segunda também simples, mas excluindo do cálculo todas stop words e a última, mais robusta, excluindo stop words e também implementando o cálculo de média ponderada por coeficiente de variação. Ao final da iteração são informadas as 3 médias de erro de predição para a comparação.

2.3. Buscar comentários associados a palavras (10%)

Como informado no início do documento, foi implementado um campo tipo Lista* para armazenar os N comentários associados a cada palavra. Esta lista é acessada usando a função fileAssocComments() que chama search_trie() para localizar a palavra e então a função save_list() para armazenar todos comentários e suas notas em um arquivo “file_assocComments.txt”

2.4. Buscar palavras usando radicais (20 %)

Para a busca de radicais foi implementada a função search_rads() que percorre a arvore até achar o nodo que representa o radical informado. Usando esse nodo como input para a função print_trie() (funçao que printa toda a árvore), essa função irá printar toda a subárvore de raiz igual ao nodo que representa o radical.