----------Relatório sobre o Trabalho Final de CPD----------

a)Integrantes:

Felipe Girardi - estudante de Ciência da Computação UFRGS(3º semestre)

Lucca Strelow Milano - estudante de Ciência da Computação UFRGS(3º semestre)

Wellington Nascente Hirsch - estudante de Ciência da Computação UFRGS(3º semestre)

b)Infraestrutura Do Programa:

Linguagem adotada: Python(3.6.5)

Método para resolução da tarefa:

Criação de classes para implementar a solução -> Tweet() - armazena o valor do sentimento de um tweet e o tweet.

Trie() - representa o nodo de uma árvore R-Trie, possuindo um char, o valor da palavra(caso seja folha), seus filhos e o nível em que está na árvore.

Word() - armazena uma palavra, seu valor de sentimento médio, número de tweets em que aparece, valor de sentimento acumulado, indice para lista de postings e uma flag.

Criação de funções para implementar a solução -> csv2trie - lê um arquivo contendo tweets e seus pesos, retornando um vetor da classe tweet(possuindo o tweet lido e seu respectivo peso).

reduceTT - simplifica um tweet visando facilitar o cálculo do valor de sentimento(utilizada na polarizeTweet e na csv2trie).

polarizeTweet - atribui valor à tweets baseado na soma do score de cada palavra contida nele, retornando um arquivo com o tweets e seu valor resultante.

reduce2radical - reduz uma palavra ao seu radical(utilizada na reduceTT).

funcionalidadeA - encontra tweets com uma palavra escolhida(funcionalidade 'a' do trabalho).

funcionalidadeB - encontra todas as palavras referentes à um radical comum escolhido(funcionalidade 'b' do trabalho).

insertTrie - realiza a inserção de uma palavra em uma árvore Trie (utilizada na csv2TRie).

searchTrie - retorna o valor(score de sentimento) de uma dada palavra na Trie (utilizada em polarizeTweet).

dataInTrie - retorna com todas palavras contidas em uma Trie.

getByPrefix - percorre o caminho de um prefixo escolhido na Trie e então chama a função dataPrefix para imprimir e guardar em um arquivo todas as palavras que têm esse prefixo.

dataPrefix - utilizada em getByPrefix para imprimir e guardar palavras que possuem um determinado prefixo.

O programa principal(main) faz uso de um loop - que para quando o usuário não desejar inserir mais tweets - e das funções criadas para solucionar a tarefa. Ele inicia lendo um arquivo de tweets polarizados informado pelo usuário, e os salva - de forma simplificada(reduceTT) - na arvore Trie, fazendo uso da função csv2Trie. Após isso, ele salva a lista de postings em um arquivo e imprime o dicionário resultante da leitura de tweets, utilizando a função dataInTrie. Posteriormente, ele polariza um arquivo composto por tweets(sem scores de sentimentos) informado pelo usuário, fazendo uso da função polarizeTweet. Em seguida, ele realiza a funcionalidade 'a' do trabalho segundo a palavra informada pelo usuário, usando a função funcionalidadeA e logo após, realiza a funcionalidade 'b' do trabalho - com o prefixo informado também peo usuário - utilizando a função funcionalidadeB. Por último, caso o usuário deseje, ele insere mais tweets de um arquivo(informado pelo usuário) no arquivo de tweets original, atualizando a arvore e as demais estruturas, continuando o loop até o usuário não desejar inserir novos tweets.

e)Estrutura de Dados:

Ambas as funcionalidades utilizam uma R-Trie, sendo que a funcionalidade 'a' também utiliza uma lista de postings.

A lista de postings foi utilizada na funcionalidade 'a' visando obter um registro que permita eficiente acesso ao tweet indicado por uma palavra no arquivo, já que procurá-la diretamente no tweet do arquivo principal seria uma tarefa muito custosa, dado o número de tweets que possam ser inseridos e seu tamanhos.

A R-Trie foi utilizada em ambas as funcionalidades pois dado o número de informações a serem armazenadas, é necessária uma estrutura cujo tempo de execução não dependa do número total

de elementos - A Trie possui essa propriedade, sendo que quanto maior seu tamanho mais eficiente torna-se o uso do espaço. A busca na trie restringe o número de elementos a serem observados e comparados a cada char analizado - percorrendo o caminho indicado por eles, diminuindo assim o tempo de procura das palavras dos tweets - provando-se ser muito útil com um grande numero de tweets e inserção - por conta de sua análise por caminho de chars. A Trie também pode associar valores às suas folhas, armazenando assim, os dados de cada palavra - outra propriedade que foi utilizada na resolução da tarefa(para armazenar os dados relacionados aos sentimentos).

Inserção na Trie -> Mesma coisa da letra d)(feita pelo Wellington)

Inserção de Postings -> Durante a inserção de uma palavra na Trie, é também feita a inserção de um índice em ordem crescente(sendo 0, o valor da primeira palavra, 1 da segunda, seguindo adiante), que servirá para representar a palvra na lista de Postings e de um posting que indicará a linha ocupada pelo tweet contendo a palavra em questão no arquivo. Com essas informações, é feito um arquivo possuindo o índice da palavra e as linhas dos tweets que a contém em cada linha.

Atualização -> A cada loop, os novos tweets são inseridos no arquivo principal e suas palavras são inseridas na Trie, atualizando os scores das palavras ja existentes e adicionando às novas.A lista de Posts também recebe os novos apontadores referentes às linhas criadas no arquivo principal após a inserção dos tweets. Com isso, a funcionalidade 'a' realiza a criação de mais um arquivo - referente a palavra que o usuário deseja procurar - e armazena nele todos os tweets referentes àquela palavra - tendo assim, com o passar das atualizações, vários arquivos de tweets referentes a palavras. A funcionalidade 'b' também realiza a criação de um arquivo - referente ao prefixo escolhido pelo usuário - e nele armazena todas as palavras derivadas a partir desse prefixo - tendo assim, ao passar das atualizações, vários arquivos de palvras referentes à prefixos.