Nama : Muhamad Achir Suci Ramadhan

NPM : 1706979354

Laporan Tugas 4 Perolehan Informasi

Program dapat dilihat pada file T4_1706979354_MuhamadAchirSuciRamadhan.pl.

Sebelum menjelaskan algoritma yang dibuat, ada beberapa hal yang ditemukan di dalam data yang diberikan:

1. Terdapat beberapa *tweet* yang duplikat, dengan detail id sebagai berikut:
   - 637929733460656129 (3 kali)
   - 633694597932146688 (2 kali)
   - 637935451114876928 (2 kali)
   - 636546118990479360 (2 kali)
   - 637580597892767744 (2 kali)
   - 641573581390921728 (2 kali)
   - 646612658242785280 (2 kali)
   - 646615593475420160 (2 kali)

   Pada program yang penulis buat, penulis menggunakan *hash* untuk memetakan *id tweet* ke *tweet* yang bersangkutan. Hal ini menyebabkan semua *tweet* yang diproses unik. Dengan demikian, *tweet* yang program penulis keluarkan pada 1706979354_output_a.txt ada sebanyak 612 buah *tweet*, kurang 9 buah dibandingkan *tweet* pada twitter_rupiah.txt yang ada sebanyak 621 buah *tweet*.

2. Terdapat kata yang ditemukan di positif.txt dan negatif.txt, yaitu “ramai”. Penulis tidak tahu apakah ada kata lain lagi yang ditemukan di keduanya atau tidak. Adanya kata “ramai” ini menyebabkan nilai sentimen *tweet* dengan id 636043729392898048 sangat bergantung pada implementasi, karena kata “ramai” menjadi satu-satunya kata yang memiliki sentimen di dalam *tweet* tersebut. Jika yang duluan dipertimbangkan adalah kata-kata positif, lalu menggunakan *elsif* untuk mengecek kata-kata negatif, maka *tweet* tersebut akan dinilai positif. Sebaliknya, jika yang dipertimbangkan terlebih dahulu adalah kata-kata negatif, lalu menggunakan *elsif* untuk mengecek kata positif, maka *tweet* tersebut akan dinilai negatif.

   Untuk mengatasi hal ini, penulis memutuskan untuk tidak menggunakan *elsif* dan membiarkan pengecekan dilakukan dua kali. Hal ini menyebabkan kata “ramai” terhitung sebagai +1 - 1 = 0, sehingga *tweet* dengan id tersebut terhitung netral.

Cara kerja program:

Cukup letakkan kode T4_1706979354_MuhamadAchirSuciRamadhan.pl pada direktori yang sama dengan file-file data yang diberikan. Setelah itu, jalankan pada terminal dengan *command* “T4_1706979354_MuhamadAchirSuciRamadhan.pl”. Program yang dibuat akan mencetak data-data duplikat, mencetak iterasi pada *k-means clustering*, dan men-*generate* file 1706979354_output_a.txt dan 1706979354_output_b.txt sesuai permintaan soal.

Penjelasan algoritma:

A. *Clustering*

Clustering yang dibuat menggunakan k-means clustering biasa dengan k = 4. *Cluster* hanya berubah pada iterasi pertama, selanjutnya sudah konvergen dengan detail:

Iterasi pertama:

Id centroid awal dan banyak elemen anggota *cluster*:

- *Cluster* 1: 637923108154814464 (270 anggota)
- *Cluster* 2: 637761919835762688 (60 anggota)
- *Cluster* 3: 635862704486379520 (142 anggota)
- *Cluster* 4: 641583206056103936 (140 anggota)

Centroid setelah pencarian *medoid* baru setiap *cluster*:

- *Cluster* 1: 637923097887117313
- *Cluster* 2: 638018713258582016
- *Cluster* 3: 643268417353416705
- *Cluster* 4: 651763752300249088

Iterasi kedua:

Id centroid awal dan banyak elemen anggota *cluster*:

- *Cluster* 1: 637923097887117313 (222 anggota)
- *Cluster* 2: 638018713258582016 (25 anggota)
- *Cluster* 3: 643268417353416705 (151 anggota)
- *Cluster* 4: 651763752300249088 (214 anggota)

Centroid setelah pencarian *medoid* baru setiap *cluster* (tidak berubah, *centroid* akhir):

- *Cluster* 1: 637923097887117313
- *Cluster* 2: 638018713258582016
- *Cluster* 3: 643268417353416705
- *Cluster* 4: 651763752300249088

Hasil lebih lengkap dapat dilihat pada file 1706979354_output_a.txt.

Pada bagian pencarian *medoid*, digunakan rata-rata dari *Jaccard distance* suatu *tweet* ke semua *tweet* lainnya di dalam *cluster* yang sama. Tetapi, karena setiap *tweet* meninjau semua *tweet* pada *cluster* yang sama, maka pembagi dari perhitungan rata-rata tersebut pasti selalu sama untuk *cluster* yang bersangkutan, yaitu banyak anggota *cluster* tersebut. Sehingga, untuk menghindari kemungkinan *underflow* (walaupun sangat kecil kemungkinannya), penulis tidak membaginya lagi dengan banyak anggota *cluster*, penulis langsung mencari *tweet* yang memiliki jumlahan *Jaccard distance* minimal ke semua *tweet* lainnya di *cluster* tersebut.

Pada bagian perhitungan *Jaccard distance*, penulis tidak memperhitungkan banyaknya kata yang sama. Penulis menganggap setiap kata unik memiliki bobot satu. Sehingga, apabila ada suatu kata yang muncul berkali-kali di dalam *tweet*, kata

tersebut tetap hanya akan dihitung satu kali, begitu juga pada bagian perhitungan irisan dan gabungannya.

Cluster 2 memiliki ukuran yang lebih kecil dibandingkan *cluster* lainnya karena hanya *tweet-tweet* tersebut yang membahas mengenai "jokowi"- "memecat"-"puan"-"maharani".

Cluster 1 berisi *tweet-tweet* yang membahas mengenai "jokowi" - "pemerintah" - "rupiah" sehingga terkumpul pada satu *cluster*.

Cluster 3 berisi *tweet-tweet* yang umumnya membahas mengenai "jokowi" - "melemah" - "rupiah". Mirip dengan *cluster* 1, tetapi kata-kata yang digunakan memiliki kemiripan sendiri dibandingkan *tweet-tweet* pada *cluster* 1. Sehingga, *tweet-tweet* ini terkumpul pada *cluster* 3.

Cluster 4 berisi *tweet-tweet* yang lebih beragam tentang "jokowi" - "melemah" - "menguat". Seharusnya mirip dengan *cluster* 1 dan 3, tetapi kata-kata yang dipilih memang lebih berbeda dan beragam dibandingkan kedua *cluster* tersebut dan membentuk *cluster* sendiri.

B. Sentiment Analysis

Untuk menghitung nilai sentimen setiap *tweet*, digunakan formula *polarity sentiment* yang diberikan pada deskripsi soal. Tetapi, melihat daftar kata yang diberikan ada yang berimbuhan ada yang tidak, penulis berasumsi bahwa kata yang dicocokkan antara *tweet* dengan daftar kata positif dan negatif yang diberikan harus tepat sama. Dengan demikian, penulis tidak melakukan *stemming* pada *tweet* yang diberikan maupun pada daftar kata yang diberikan.

Pada perhitungan nilai sentimen ini, penulis tetap memperhitungkan banyaknya kata yang sama, jika ada kata positif yang sama muncul lebih dari satu kali di dalam *tweet* yang sama, maka bobot yang diberikan tetap sebanyak berapa kali kata tersebut muncul. Hal yang sama berlaku untuk kata negatif.

Hasil perhitungan banyak setiap sentimen pada setiap *cluster* dijelaskan pada tabel berikut:

Nomor *Cluster*	#*tweet* positif	#*tweet* negatif	#*tweet* netral
1	5	15	202
2	25	0	0
3	13	65	73
4	102	46	66

Dengan nomor *cluster* yang dimaksud merujuk pada nomor *cluster* yang sama pada soal bagian A.