ANALISIS DAN IMPLEMENTASI *COMMUNITY DETECTION* MENGGUNAKAN ALGORITMA DBSCAN PADA *TWITTER*

ANALYSIS AND IMPLEMENTATION OF COMMUNITY DETECTION USING DBSCAN ALGORITHM ON TWITTER

Lulu Alfi'a Rahma Ningsih[1], Imelda Atastina[2], Anisa Herdiani[3]

[1,2,3]Prodi S1 Teknik Informatika, Fakultas Informatika, Universitas Telkom

[1]lulualfiaaa@gmail.com, [2]imdgrupif@gmail.com, [3]anisaherdiani@gmail.com

Abstrak

***Community detection* atau *clustering* merupakan proses partisi data ke dalam kelompok-kelompok. Data yang berada dalam satu kelompok memiliki kemiripan (*similarity*) karakteristik antar satu sama lainnya dan berbeda dengan kelompok lain, seperti halnya kemiripan antar individu pada *social network*. Pada penelitian ini, *community detection* dilakukan pada *social network Twitter* berdasarkan kemiripan interaksi (*follows, mentions, reply*) antar pengguna dengan menggunakan algoritma DBSCAN dan perhitungan kualitas *cluster* dengan menggunakan *modularity*. Algoritma DBSCAN memiliki dua parameter penting, yakni *epsilon* dan *minPts* yang bernilai *random*. Dari hasil penelitian yang dilakukan, hasil terbaik didapat dari data ke-10 dengan membentuk 4 *cluster* yang menghasilkan nilai *modularity* 0,61492 dari nilai *epsilon* 0,3 dan *minPts* 4. Hal tersebut dikarenakan jumlah *cluster* yang terbentuk dari adanya keterkaitan nilai *epsilon* dan *minPts* dengan nilai kemiripan (*similarity*) dan nilai *modularity* disebabkan adanya kepadatan dari relasi dan bobot *similarity* yang besar. Sehingga untuk mendapatkan hasil *cluster* yang baik diperlukan penentuan nilai *epsilon* dan *minPts* yang tepat.**

Kata Kunci: *social network*, algoritma DBSCAN, *modularity, Twitter*

Abstract

Community detection or clustering is the process of partitioning data into groups. Data within a group have similarity characteristics between each other and are different from other groups, as do similarities between individuals on social networks. In this research, community detection is done on Twitter social network based on the following interaction (follows, mentions, reply) between users by using DBSCAN algorithm and cluster quality calculation using modularity. DBSCAN algorithm has two important parameters, namely epsilon and minPts are random value. From the results of research conducted, the best results obtained from the data to-10 by forming 4 clusters that produce the value of modularity 0.61492 of epsilon value 0.3 and minPts 4. That is because the number of clusters formed from the existence of the linkage of epsilon and minPts value with the similarity and modularity values are due to the density of the relation and the great weight of similarity. So to get a good cluster results required the determination of the exact epsilon and minPts value.

Keywords: social network, DBSCAN algorithm, modularity, Twitter

1. Pendahuluan

Social network merupakan suatu jaringan yang dapat membantu masyarakat dalam berkomunikasi satu sama lain. Perkembangan *social network* dalam masyarakat sudah sangat pesat. Selain untuk berkomunikasi, masyarakat juga bisa bertukar informasi dan memasang iklan. Salah satu *social network* yang banyak digunakan dan tengah marak di masyarakat adalah *Twitter*. Berdasarkan hasil survei yang dilakukan oleh APJII tahun 2016, penggunaan *Twitter* berkembang pesat dan merupakan 5 situs terbesar yang populer di Indonesia dimana telah memiliki pengguna mencapai 7,2 juta [1]. Sehingga *Twitter* memiliki skala yang cukup besar dalam berbagai kegiatan komunikasi untuk sebuah *social network*. Namun *Twitter* tidak hanya sekedar digunakan untuk media pertemanan, namun sering juga digunakan sebagai media bertukar informasi, memasang iklan sebuah produk maupun hal lainnya. Sehingga *Twitter* memiliki ukuran dan bentuk data yang berbeda-beda yang menarik untuk dijadikan sebuah bahan penelitian, terutama *community detection*.

Community detection atau *clustering* merupakan salah satu cara untuk menemukan komunitas yang terbentuk dari interaksi yang ada pada suatu jaringan sosial yang besar. Dalam hal ini, *community detection* bertujuan untuk membagi jaringan sosial dalam kelompok-kelompok berdasarkan interaksi masing-masing. Daerah yang dimaksud berupa entitas-entitas yang memiliki hubungan dan kesamaan (*similarity*) karakteristik sehingga dapat bermanfaat sebagai penentuan dalam menargetkan skema pemasaran produk, mengetahui berita maupun info terbaru, menghitung kepopuleran suatu barang, dan mendeteksi isu yang beredar di masyarakat [2]. Sedangkan, bagi entitas-entitas yang sama sekali tidak memiliki hubungan dan kesamaan karakteristik akan dianggap sebagai *noise* dimana jika pun dieliminasi tidak akan berpengaruh terhadap kualitas kelompok-kelompok yang sudah terbentuk.

Peneliti sebelumnya menggunakan beberapa algoritma *clustering* antara lain algoritma K-L, metode *hierarchical clustering* berdasarkan pengukuran *similarity*, dan algoritma G-N berdasarkan penghapusan *edges gradually*. Masalah umum pada metode-metode tersebut adalah tidak memberikan informasi yang objektif tentang berapa banyak pengguna yang harus dibagi dalam jaringan [3]. Terdapat juga algoritma DBSCAN (*density-based spatial clustering of applications with noise*) yang biasa digunakan pada kasus *clustering*. Algoritma DBSCAN dipilih karena dapat menangani data berskala besar, mengenali *noise/outlier*, dan mampu mengidentifikasi *cluster* dengan ukuran dan bentuk yang berbeda-beda [4]. Oleh sebab itu, pada penelitian ini membahas mengenai hasil pembentukan komunitas dengan algoritma DBSCAN. Untuk mengetahui kualitas komunitas/*cluster*, peneliti menggunakan *modularity*. Penggunaan *modularity* dikarenakan *modularity* dapat menghitung kualitas tidak hanya pada komunitas tersebut, tapi juga kualitas antar komunitas.

2. Dasar Teori

2.1. *Social Network*

2.1.1. Definisi *Social Network*

Social Network merupakan peta keterhubungan antar individu dimana dapat kita amati aktifitas sosial yang terjadi didalamnya [5]. Hubungan antar individu digambarkan dengan sebuah graf, setiap individu di representasikan sebagai *node* yang dapat dihubungkan satu sama lain dengan sebuah *edge*. Grafik yang digunakan untuk mewakili data sosial memungkinkan untuk dilakukan analisis informasi yang tertanam dalam hubungan sosial [6].

2.1.2. *Twitter*

Twitter merupakan situs *social network micro-blogging* yang terus berkembang yang memungkinkan pengguna untuk mengirim dan berbagi pesan singkat hingga 140 karakter. *Twitter* memiliki beberapa fitur yang selalu digunakan secara umum oleh *pengguna*, berikut penjelasannya :

2.1.2.1. *Follow*

Follow dapat berupa *follower* (pengikut) dan *friend/followed* (yang mengikuti). *Follower* merupakan relasi dimana pengguna lain mengikuti akun Anda, sedangkan *friend/followed* merupakan relasi dimana Anda mengikuti pengguna lainnya.

2.1.2.2. *Tweet*

Tweet merupakan segala sesuatu yang pengguna *post* di *twitter*, baik berupa tulisan, gambar, maupun video. Dalam satu kali *posting* hanya dapat dilakukan maksimal dengan 140 karakter.

2.1.2.3. *ReTweet* (RT)

ReTweet merupakan pilihan dimana pengguna dapat menyebarluaskan informasi dari pengguna lain secara mudah tanpa mengubah isi *tweet* sebenarnya.

2.1.2.4. *Hash Tag* (#)

Hashtag (#) biasanya dilakukan oleh pengguna untuk memudahkan penelusuran *tweet* yang mempunyai topik tertentu. *Tweet* yang dimaksud seperti *tweet* yang saling berkaitan dan lebih spesifik dalam membahas sebuah topik.

2.1.2.5. *Mention*

Mention yaitu interaksi yang dilakukan untuk menyampaikan informasi kepada pengguna lain dalam sebuah kalimat *tweet* seseorang dengan menuliskan '@(namapengguna)'. Dalam satu *tweet* dapat menyebutkan beberapa pengguna.

2.1.2.6. *Reply*

Reply merupakan balasan oleh seorang pengguna yang menulis *tweet*. Penulisan *reply* biasanya didahului dengan '@(namapengguna)' dan dilanjutkan dengan kalimat balasan.

2.2. *Community Detection*

Community detection atau dapat disebut dengan *clustering* merupakan pengelompokan yang melakukan pemisahan kedalam sejumlah kelompok berdasarkan karakteristik tertentu. Dalam ilmu *data mining, community detection* dikelompokkan berdasarkan hubungan informasi yang ada pada data [6]. Pada penelitian ini, *community detection* digunakan untuk menemukan komunitas yang terbentuk dari interaksi yang ada pada suatu jaringan sosial. Tujuan dari pengelompokan adalah mengelompokkan data berdasarkan tingkat kesamaan satu sama lain (keterhubungan) yang berbeda dengan kelompok lainnya.

2.3. *Similarity*

2.3.1. *Follow Similarity*

Tujuan dari *follow similarity* adalah untuk menghitung kedekatan antar pengguna berdasarkan interaksi *follow*. Untuk mengetahui nilai *follow similarity* antara u_a dan u_b dirumuskan sebagai berikut : [7]

$$sim_{follow}(a,b) = \frac{C_{friend}}{\sqrt{|friend_a|}\sqrt{|friend_b|}} + \frac{C_{follower}}{\sqrt{|follower_a|}\sqrt{|follower_b|}} \quad (1)$$

Keterangan :

C_{friend} = jumlah *friend* yang sama antar 2 pengguna, yakni u_a dan u_b

$C_{follower}$ = jumlah *follower* yang sama antar 2 pengguna, yakni u_a dan u_b

$|friend_a|$ = jumlah *friend* dalam 1 pengguna (u_a)

$|friend_b|$ = jumlah *friend* dalam 1 pengguna (u_b)

$|follower_a|$ = jumlah *follower* dalam 1 pengguna (u_a)

$|follower_b|$ = jumlah *follower* dalam 1 pengguna (u_b)

2.3.2. *Mention Similarity*

Tujuan dari *mention similarity* adalah untuk menghitung kedekatan antar pengguna berdasarkan interaksi *mention*. Untuk mengetahui nilai *mention similarity* antara u_a dan u_b dirumuskan sebagai berikut : [7]

$$sim_{mention}(a,b) = \frac{C_{mention}}{\sqrt{|R_a|}\sqrt{|R_b|}} + \frac{n_{ab} + n_{ba}}{|R_a| + |R_b|} \quad (2)$$

Keterangan :

$C_{mention}$ = jumlah pengguna yang sama di*mention* oleh u_a dan u_b

n_{ab} = seberapa banyak u_a *mention* ke u_b (liat bobot)

n_{ba} = seberapa banyak u_b *mention* ke u_a (liat bobot)

$|R_a|$ = jumlah pengguna yang di*mention* u_a

$|R_b|$ = jumlah pengguna yang di*mention* u_b

2.3.3. *Reply Similarity*

Tujuan dari *reply similarity* adalah untuk menghitung kedekatan antar pengguna berdasarkan interaksi *reply*. Untuk mengetahui nilai *reply similarity* antara u_a dan u_b dirumuskan sebagai berikut : [7]

$$sim_{reply}(a,b) = \frac{C_{reply}}{\sqrt{|R_a|}\sqrt{|R_b|}} + \frac{n_{ab} + n_{ba}}{|R_a| + |R_b|} \quad (3)$$

Keterangan :

$C_{mention}$ = jumlah pengguna yang sama di*reply* oleh u_a dan u_b

n_{ab} = seberapa banyak u_a *reply* ke u_b (liat bobot)

n_{ba} = seberapa banyak u_b *reply* ke u_a (liat bobot)

$|R_a|$ = jumlah pengguna yang di*reply* u_a

$|R_b|$ = jumlah pengguna yang di*reply* u_b

2.3.4. *Total Similarity*

Dalam bagian ini menggabungkan *similarity* untuk medapatkan *total similarity*. Definisi *total similarity* antar pengguna sebagai berikut : [7]

$$sim_{total}(a,b) = \alpha\, sim_{follow}(a,b) + \beta\, sim_{mention}(a,b) + \gamma\, sim_{reply}(a,b) \quad (4)$$

Semakin tinggi nilai *similarity*, semakin dekat hubungan dua pengguna tersebut. Sedangkan untuk α, β, dan γ merupakan parameter antara 0 dan 1 untuk mengontrol bobot *similarity*, dan total dari α + β + γ = 1.

2.4. Algoritma DBSCAN

Algoritma DBSCAN adalah sebuah algoritma *clustering* yang dikembangkan berdasarkan tingkat kerapatan data (*density-based*). Dimana algoritma ini menumbuhkan daerah yang memiliki kerapatan tinggi menjadi *cluster-cluster* dan menemukan *cluster-cluster* tersebut pada bentuk bebas dalam sebuah ruang *database* dengan memanfaatkan *noise* yang mewakili daerah yang kurang padat untuk memisahkan antara *cluster* satu dengan yang lainnya pada objek dalam ruang data [8]. DBSCAN memiliki 2 parameter yaitu *epsilon* (radius maksimum antar *node*) dan *minPts* (jumlah minimum *points* dalam satu *cluster*) yang nilainya ditentukan secara *random*. Dalam membentuk *cluster*, ada tiga kategori yang dikenal dalam DBSCAN yakni :

a. *Core*, merupakan label *node* yang jika jumlah *points* disekitarnya dalam radius *epsilon* dan lebih dari sama dengan *minPts.*
b. *Border*, merupakan label *node* yang *points*nya termasuk dalam tetangga dari poin *core* selama itu bukan poin *core*.. Poin *border* bisa menjadi tetangga dari beberapa poin *core.*
c. *Noise/Outlier*, merupakan label *node* dimana *points* tidak masuk kedalam *core* maupun *border*.

Dalam melakukan proses input output, algoritma DBSCAN dapat digambarkan sebagai berikut : [9]

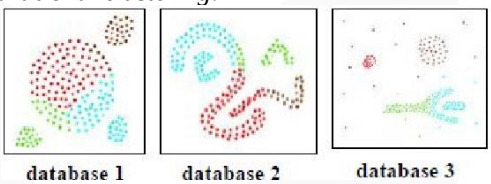
Input : Dataset berisi objek-objek *n, minPts,* dan *epsilon*

- Letakkan sebuah *edge* antara masing-masing pasang poin *core* dengan jarak *epsilon* masing-masing
- Tandai poin *noise/outlier*
- Tandai masing-masing kelompok poin *core* kedalam sebuah *cluster* yang terpisah
- Tentukan masing-masing poin *border* secara acak/sembarang ke salah satu dari *cluster-cluster* yang terhubung dengan poin-poin *core*

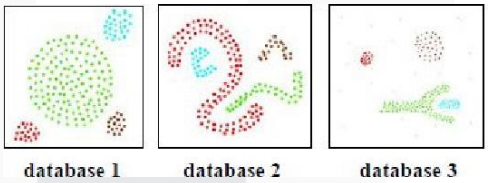
Output : Kumpulan objek-objek ter*cluster*

Kelebihan serta kekurangan dari algoritma DBSCAN sebagai berikut : [8] [10]

a. Kelebihan
   - Dapat menghasilkan *cluster* yang lebih akurat dari bentuk data yang tidak beraturan jika dibandingkan dengan *partitional clustering*.



Gambar 1 Contoh Hasil *Cluster* dengan Clarans (*Partitional Clustering*) [11]



Gambar 2 Contoh Hasil *Cluster* dengan DBSCAN (*Density-Based Clustering*) [11]

   - Dapat menangani *outlier/noise*
   - Baik untuk data dalam jumlah besar

b. Kekurangan
   - Kurang maksimal pada dataset berdimensi tinggi
   - Penentuan parameter yang cukup rumit
   - Memiliki permasalahan saat identifikasi cluster dari kepadatan yang bervariasi
   - Tidak membentuk fungsi kepadatan yang sesungguhnya, tetapi lebih ke arah poin-poin kepadatan yang saling berhubungan dan membentuk graf

2.5. *Modularity*

Modularity merupakan salah satu alat ukur untuk menganalisis dan mengkarakteristik bentuk jaringan yang dapat digunakan sebagai acuan dalam menentukan kualitas komunitas yang ada pada jaringan tersebut. Ukuran dirancang dengan mencari kemungkinan apakah pembagian dalam membentuk cluster sudah *valid* atau tidak, dari jaringan untuk satu atau lebih yang memiliki modularitas tinggi. Berikut rumus *modularity* : [9]

$$Q = \sum_{i=1}^{N} (e_{ii} - a_i^2) = \frac{m_s}{m} - \left(\frac{d_s}{m}\right)^2 \qquad (5)$$

Keterangan :

N = jumlah *cluster*

a_i = jumlah bobot *edge* pada dan antar *cluster*/total bobot *edge*

e_{ii} = jumlah bobot *edge* pada *cluster*/total bobot *edge*

m_s = jumlah bobot *edge* pada *cluster*

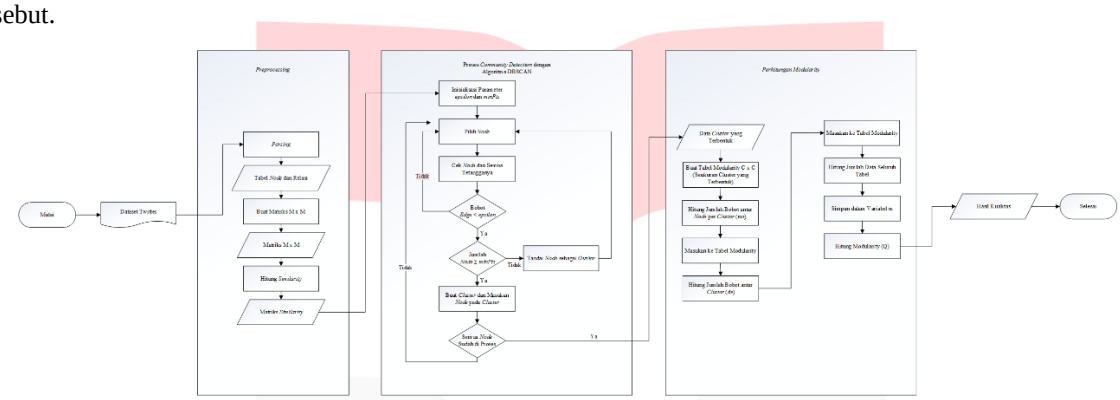
d_s = jumlah bobot *edge* pada dan antar *cluster*

m = total bobot *edge* pada graf

3. Pembahasan

3.1. Gambaran Sistem

Proses implementasi *community detection* dengan algoritma DBSCAN pada *Twitter* digambarkan pada gambar 1, dimana proses dimulai dari membentuk komunitas hingga mendapatkan nilai kualitas dari komunitas tersebut.



Gambar 3 Gambaran Sistem

3.2. Pengujian dan Analisis

3.2.1. Tujuan Pengujian

Pengujian dilakukan untuk mengetahui performansi dari algoritma DBSCAN dilihat dari jumlah komunitas yang terbentuk dari penentuan nilai *epsilon* dan *minPts* maupun besarnya data, dan nilai *modularity* dari sistem yang telah dibuat.

3.2.2. Dataset

Dataset *Twitter* yang digunakan berjumlah 10 data dengan jumlah pengguna dan relasi yang berbeda-beda tiap datanya. Dataset yang didapat terdiri dari jumlah pengguna, *follows*, *mentions*, dan *replies to*. Berikut rincian dataset yang digunakan :

Tabel 1 Rincian Dataset

Data ke-	Jumlah Pengguna	Jumlah Follows	Jumlah Mentions	Jumlah Replies to
1	90	286	55	0
2	225	260	289	1
3	238	533	132	395
4	319	786	148	79
5	1324	0	2381	19
6	1414	13853	2595	214
7	1474	14975	3241	265
8	2405	0	5160	544
9	3166	0	7947	755
10	5017	0	12785	2011

3.2.3. Metrik Pengujian

Beberapa metrik pengujian yang digunakan dalam skenario pengujian sebagai berikut :

1 Nilai *Similarity*

Menghitung *similarity* dari masing-masing interaksi antar pengguna yang akan digunakan sebagai bobot *edge* antar pengguna. Untuk kemudian akan dijadikan sebuah patokan apakah dapat memenuhi untuk menjadi *cluster* atau tidak.

2 Nilai Korelasi
Mengetahui korelasi antara *similarity* dengan jumlah pengguna dan relasi pada masing-masing data. Nilai 0 jika tidak berkorelasi, sedangkan nilai 1 jika berkorelasi.
3 Nilai *Modularity*
Menghitung kualitas *cluster* yang terbentuk. Perhitungan dilakukan dalam satu *cluster* dan antar *cluster*.

3.2.4. Hasil Pengujian

3.2.4.1. Hasil Pengujian Skenario 1

Skenario 1 bertujuan untuk mengetahui pengaruh interaksi antar pengguna dengan nilai *similarity* dan algoritma DBSCAN. Hasil yang didapat berupa *range* nilai *similarity* seperti tabel dibawah ini :

Tabel 2 *Range* Nilai Hasil *Similarity*

Data ke-	Jumlah Pengguna	Jumlah Relasi	*Range* Nilai *Similarity*
1	90	341	0,0937 - 1,669024
2	225	550	0,0693 - 1,1
3	238	1060	0,0127 - 2,258883
4	319	1013	0,0727 - 2,076142
5	1324	2400	0,0061 - 2,04044
6	1414	16662	0,0120 - 6
7	1474	18481	0,0135 - 2,894945
8	2405	5704	0,0119 - 2,5
9	3166	8702	0,0099 - 3,116649
10	5017	14796	0,0303 - 8

Dari tabel 2 didapat nilai korelasi antara jumlah pengguna dengan *range* nilai *similarity* sebesar 0,765 dan antara jumlah relasi dengan *range* nilai *similarity* sebesar 0,737.

3.2.4.2. Hasil Pengujian Skenario 2 dan 3

Skenario 2 bertujuan untuk mengetahui pengaruh parameter *epsilon* dan *minPts* terhadap nilai *modularity*, sedangkan skenario 3 bertujuan untuk mengetahui pengaruh banyaknya data terhadap hasil komunitas yang terbentuk. *Range* nilai *epsilon* yang digunakan mulai dari 0,1 sampai 10. Hasil yang didapat berupa nilai *modularity* maksimal yang diperoleh seperti tabel dibawah ini :

Tabel 3 Nilai *Modularity* Maksimal yang Diperoleh Berdasarkan Berbagai Nilai *epsilon* dan *minPts*

Jumlah Pengguna	Jumlah Relasi	*epsilon*	*minPts*	Jumlah *Outlier*	Jumlah *Cluster*	*Modularity*	*Running Time* (detik)
90	341	2,6	9	65	6	0,35045	168.098
225	550	2,2	4	219	3	0,48529	39.696
238	1060	3,9	10	163	5	0,38922	635.964
319	1013	2,3	9	290	4	0,59955	330.794
1324	2400	0,2	10	1253	4	0,25493	1624.853
1414	16662	0,2	10	1371	3	0,11285	116653.766
1474	18481	5,8	10	875	346	0,0090592	340027.175
2405	5704	0,9	10	2336	4	0,32624	26038.815
3166	8702	2,9	10	3077	5	0,2453	43003.026
5017	14796	0,3	4	4988	4	0,61492	46342.320

3.2.5. Analisis

3.2.5.1. Analisis Pengaruh Interaksi Terhadap Hasil *Cluster*

Pada tabel 2 dapat diperhatikan bahwa nilai korelasi yang didapat kuat karena bernilai positif dan mendekati 1. Sehingga karakteristik data yang dimiliki dengan nilai similarity yang didapat sangat berhubungan. Untuk kemudian pengaruh tinggi rendahnya nilai α, β, dan γ pada nilai similarity tidak dapat digeneralisasi untuk semua jenis data karena setiap data memiliki relasi yang bersifat unik. Relasi unik yang dimaksud merupakan bobot dari masing-masing jenis interaksi yang ada pada setiap data berbeda-beda. Kemudian dari jumlah relasi yang semakin banyak, nilai modularity yang dihasilkan belum tentu semakin besar. Hal tersebut dipengaruhi dari bobot jenis relasinya.

Nilai *similarity* sangat berpengaruh dengan *modularity*, karena nilai *similarity* tersebut digunakan sebagai patokan untuk membentuk *cluster*. Setelah *cluster-cluster* terbentuk, untuk menghitung kualitasnya dibutuhkan *modularity*. Dalam perhitungan *modularity*, nilai *similarity* akan digunakan sebagai bobot *edge*

antar *node* yang ada pada *cluster*. Sehingga yang berpengaruh terhadap kualitas *cluster* adalah nilai *similarity*.

3.2.5.2. Analisis Pengaruh Parameter *epsilon* dan *minPts* Terhadap Hasil *Cluster*

Epsilon merupakan batas maksimum antar pengguna dan *minPts* batas minimum jumlah pengguna dalam sebuah *cluster*. Nilai *epsilon* yang semakin besar tidak selalu membuat nilai *modularity* semakin tinggi, dikarenakan pengaruh nilai *similarity* dengan nilai *epsilon*. Dari berbagai nilai *minPts*, nilai *epsilon* yang signifikan untuk mendapatkan nilai modularity maksimum terdapat pada *range* di tabel 4.

Tabel 4 *Range epsilon*

Jumlah Pengguna	*Range epsilon*
90	≥ 2,6
225	≥ 2
238	≥ 3,9
319	≥ 2,3
1324	0,2
1414	0,2
1474	≥ 5,8
2405	0,9 – 1
3166	2,9
5017	≥ 0,3

Pengaruh nilai *similarity* dengan nilai *epsilon* pun juga dapat membuat kemunculan *outlier/noise* seperti pada tabel 3. Ketika *node* yang diproses tidak memenuhi parameter *epsilon* maupun *minPts* dimana hubungan *node* tersebut dengan tetangganya melebihi nilai *epsilon* ataupun jumlah tetangganya kurang dari *minPts.*

Pada tabel 3, terdapat juga data mengenai nilai *modularity* maksimal yang diperoleh berdasarkan berbagai nilai *epsilon* dan *minPts*. Data yang memiliki jumlah pengguna 1474 dengan jumlah relasi 18481 menghasilkan jumlah *cluster* yang terbanyak dibandingkan dengan data lainnya yakni 346 *cluster*. Hal tersebut dikarenakan jumlah relasi pada data tersebut lebih banyak dan sesuai dengan prinsip kerja DBSCAN dimana jika *node* yang dalam tahap pengecekan memenuhi kedua parameter *epsilon* dan *minPts* maka dapat membentuk *cluster*. Sehingga dengan jumlah pengguna/*node* dan jumlah relasi yang padat dari data tersebut memungkinkan menghasilkan *cluster* yang lebih tinggi.

Namun untuk nilai *modularity* yang didapat lebih kecil dibandingkan dengan data lainnya. Penyebabnya adalah *cluster-cluster* yang terbentuk tidak sepadat *cluster* yang terbentuk dari data lainnya walaupun bila dilihat dari data secara keseluruhan relasinya begitu padat.

3.2.5.3. Analisis Pengaruh Besarnya Data

Pengaruh proses lebih lama atau tidaknya berdasarkan tabel 3, tidak hanya jumlah pengguna namun jumlah relasi juga, karena bila relasi padat dan bobot besar, nilai *similarity* akan tinggi sehingga proses yang dilakukan akan semakin lama. Sehingga jumlah pengguna yang banyak dan relasi padat dengan bobot *similarity* yang tidak tinggi dapat lebih cepat prosesnya dibandingkan dengan yang bobot *similarity*nya tinggi.

Kemudian untuk menentukan nilai *modularity*nya, nilai *epsilon* dan *minPts* yang menentukan. Nilai *modularity* yang dihasilkan dari masing-masing data berbeda. Pada tabel 3, data yang memiliki nilai *modularity* terbesar dibandingkan dengan data lainnya bukanlah yang memiliki jumlah relasi banyak, namun *cluster* yang terbentuk dari data tersebut dengan memiliki kepadatan relasi yang padat dan bobot *similarity* yang tinggilah yang menyebabkan nilai *modularity* yang dihasilkan lebih besar.

4. Kesimpulan dan Saran

4.1. Kesimpulan

Berdasarkan pengujian dan analisis hasil uji pada sistem yang telah dibangun, maka didapat kesimpulan sebagai berikut :

1 Nilai *similarity* dan nilai *epsilon* serta *minPts* sangat berpengaruh satu sama lain. *Cluster* akan terbentuk bila nilai *similarity* masih dalam *range* nilai *epsilon* (kurang dari *epsilon*) dan jumlah *node* memenuhi *minPts* (lebih dari sama dengan *minPts*). Nilai *similarity* yang besar dan padat akan menghasilkan nilai *modularity* yang lebih baik.

2 Penentuan nilai *epsilon* dan *minPts* dalam pembentukan *cluster* harus tepat agar nilai *modularity* yang didapat pun bernilai tinggi. Dari pengujian yang dilakukan, kondisi-kondisi tertentu yang dipengaruhi oleh *epsilon* dan *minPts* seperti :
   a. Bila nilai *epsilon* tetap/sama dan nilai *minPts* semakin besar, maka *outlier* yang terdeteksi dan *cluster* yang terbentuk berkurang.
   b. Bila nilai *minPts* tetap/sama dan nilai *epsilon* semakin besar, maka *outlier* yang terdeteksi semakin kecil dan *cluster* yang terbentuk bertambah.

   Nilai *epsilon* yang semakin besar tidak selalu membuat nilai *modularity* semakin tinggi karena adanya pengaruh nilai *similarity* terhadap nilai *epsilon.*
3 Performansi algoritma DBSCAN dilihat dari nilai *modularity* berdasarkan pengujian yang telah dilakukan. Bernilai bagus jika data yang dimiliki besar namun memiliki bobot dan relasi yang padat, walaupun akan mengalami proses yang lama.

4.2. Saran

Adapun saran untuk mengembangkan penelitian selanjutnya mengenai *community detection* adalah :

1 Penggunaan data selain *Twitter* dan data dengan level pengguna lebih dari ribuan untuk melakukan percobaan, seperti *Youtube, Facebook, Instagram* dan lainnya.
2 Untuk mendapatkan nilai *epsilon* yang optimal dapat menggunakan *sorted k-dist graph*.
3 Untuk menentukan kualitas dari komunitas, coba gunakan selain *modularity* seperti *clustering coefficient, graph density* ataupun perhitungan lainnya.

Daftar Pustaka

[1] P. Indonesia dan A. Indonesia, “Infografis Penetrasi dan Perilaku Pengguna Internet Indonesia,” APJII, Jakarta, 2016.

[2] I. A. Nur, Community Detection Menggunakan Genetic Algorithm dalam Social Network Twitter, Bandung: Universitas Telkom, 2016.

[3] K. Bing, Z. Lihua dan L. Weiyi, “Improved Modularity Based on Girvan-Newman Modularity,” dalam *International Conference on Intelligent Systems Design and Engineering Application*, China, 2012.

[4] M. T. Furqon dan L. Muflikhah, “Clustering The Potential Risk of Tsunami Using Density-Based Spatial Clustering of Applications with Noise (DBSCAN),” *Environmental Engineering & Sustainable Technology*, vol. 03, no. 01, pp. 1-8, 2016.

[5] L. Ding dan P. Shi, “Social Network Analysis Application in Bulletin Board Systems,” dalam *International Conference on Intelligence Science and Information Engineering*, China, 2011.

[6] C. N. Utami, “Analisis dan Implementasi Community Detection Menggunakan Algoritma Girvan and Newman dalam Social Network,” IT Telkom, Bandung, 2014.

[7] Y. Zhang, Y. Wu dan Q. Yang, “Community Discovery in Twitter Based on User Interests,” *Journal of Computational Information Systems*, vol. 8, no. 3, pp. 991-1000, 2012.

[8] N. I. Sari, I. Srimuddawamah, L. P. Novita dan M. S. Munif, “Makalah DBSCAN,” https://id.scribd.com/, Malang, 2014.

[9] Y. M. ElBarawy, R. F. Mohamed dan N. I. Ghali, “Improving Social Network Community Detection Using DBSCAN Algorithm,” dalam *World Symposium on Computer Applications & Research (WSCAR)*, Sousse, Tunisia, 2014.

[10] N. M. A. Santika Devi, I. K. G. Darma Putra dan I. M. Sukarsa, “Implementasi Metode Clustering DBSCAN pada Proses Pengambilan Keputusan,” *Lontar Komputer*, vol. 06, no. 03, pp. 655-661, 2015.

[11] K. Rahmat, B. A. Putro dan S. , Implementasi Density Based Spatial Clustering Application with Noise (DBSCAN) Dalam Perkiraan Terjadinya Banjir di Bandung, Bandung: Universitas Telkom, 2011.